ERROR CONTROL FOR H.264/AVC VIDEO STREAMING

by

Seyed Mohsen Amiri
B.Sc., Isfahan University of Technology, Iran, 2004
M.Sc., Sharif University of Technology, Iran, 2006

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF APPLIED SCIENCE
in the School
of
Engineering Science

APPROVAL

Name:	Seyed Mohsen Amiri
Degree:	Master of Applied Science
Title of Thesis:	Error Control For H.264/AVC Video Streaming
Examining Committee:	Dr. Atousa HajShirMohammadi Chair
	Dr. Ivan V. Bajić, Senior Supervisor
	Dr. Jie Liang Supervisor
	Dr. Mohamed M. Hefeeda SFU Examiner
Date Approved:	Aug 19, 2009

Abstract

In recent years, the development of network and multimedia technologies has increased the demand for video delivery over various types of networks, such as the Internet or wireless networks. However, video delivery over packet-based networks poses several challenges, such as packet loss protection, limited bandwidth, and limitation on the maximum allowable delay. This thesis focuses on developing different error control techniques for video streaming over various networks to combat these challenges.

First, we focused on developing a new two-stage encoder structure, which enables motion-compensated predictive encoders to have real-time Reference Picture Selection (RPS). In the second part of this thesis, we developed a novel noncausal whole-frame error concealment algorithm. Finally, we studied error control for multicast video transmission. We developed a subset selection scheme based on Type-II hybrid ARQ/FEC, which chooses video packets to be protected based on their influence on the decoded video quality.

Executive Summary

This thesis focuses on developing different error control techniques for video streaming over various networks to combat these challenges. First, we focused on developing a new two-stage encoder structure, which enables motion-compensated predictive encoders to have real-time Reference Picture Selection (RPS). RPS is a strong error resilience technique. Servers usually do not choose RPS as their error resilience feature, because it is highly computationally demanding. In this thesis, the proposed two-stage encoder achieves significant speed-up at the streaming time at the expense of additional storage space needed.

Using channel coding such as ARQ or FEC can reduce channel errors, but can not completely eliminate them. Therefore, every video decoder needs to have an error concealment module to mitigate the effect of channel errors. In the second part of this thesis, we developed a novel noncausal whole-frame error concealment algorithm. The proposed algorithm has the ability to extract the information in the preceding and succeeding frames, and use this information to conceal the lost frame.

Finally, we studied error control for multicast video transmission. We developed a subset selection scheme based on Type-II hybrid ARQ/FEC, which chooses video packets to be protected based on their influence on the decoded video quality. This approach enables better utilization of parity packets in cases where there is not enough bandwidth to transmit sufficiently many parity packets to correct all losses at all users.

To my parents and my beloved wife

"Would you that splang of Existance spend

About the Secret-quick about it, Friend!

A hair perhaps devides the False from True

And upon what , prithee, does life depend "

— *The Keepr,* OMAR KHAYYÁM, *1123AD*

Contents

Approval **ii**

Abstract **iii**

Executive Summary **iv**

Dedication **v**

Quotation **vi**

Contents **vii**

List of Tables **xi**

List of Figures **xiv**

Preface **xix**

1 Background **1**

1.1 Video Compression Concepts 3

1.2 H.264/AVC Overview 5

1.2.1 Features 6

1.3 Error Resilience 8

1.3.1 Error Resilience Tools in H.264/AVC 9
1.3.2 Reference Picture Selection (RPS) 11
1.4 Error Concealment 14
1.4.1 Whole Frame Concealment 15
1.5 Channel Coding 16
1.5.1 Forward Error Correction (FEC) 16
1.5.2 Automatic Repeat reQuest (ARQ) 19
1.5.3 Hybrid ARQ/FEC 20
1.6 Contributions of the Thesis 20

2 Two-Stage Encoder for Fast RPS **23**
2.1 Overview of the Proposed Two-Stage Encoder 24
2.2 Performance modeling of the two-stage encoder 29
2.3 Performance Evaluation of the Two-stage Encoder 33
2.3.1 Characterization of the two-stage encoder 34
2.3.2 Performance over a lossy network 35
2.3.3 Encoding speed 36

3 Noncausal Whole-frame Concealment **43**
3.1 Whole-frame Concealment 44
3.1.1 The Method Proposed in [14] 44
3.2 Noncausal Whole-frame Concealment 46
3.2.1 Noncausal optical flow 46
3.2.2 Using Optical Flow in the Proposed Error Concealment Algorithm . 49
3.2.3 Spatiotemporal boundary matching 50
3.3 Performance Evaluation of the Noncausal Whole-Frame concealment 52

4 End to End System Performance Evaluation 60

4.1 Experimental results 60

4.2 Conclusions 65

5 Channel Coding for Video Multicast 67

5.1 Conventional Error Control Techniques for Multicasting 68

5.1.1 FEC 69

5.1.2 ARQ 69

5.1.3 Hybrid ARQ/FEC type-II 69

5.1.4 RPS 71

5.1.5 Rate Distortion Optimization (RaDiO) and Collision Distortion Optimization (CoDiO) 72

5.2 Subset Selection in Hybrid ARQ/FEC for Video Multicast 73

5.3 Subset Selection in Type-II Hybrid ARQ/FEC for Video Multicast with One Subset 74

5.3.1 Full Search for the Best Subset 74

5.3.2 Suboptimal Search Based on Simulated Annealing 82

5.4 Experimental Results 82

5.5 Analytical Results 87

5.5.1 Error Distribution 88

5.5.2 Subset Selection Effects on the Residual Packet Loss Rate without Feedback 91

5.5.3 Subset Selection Effects on the Receivers Distortion without Feedback 96

5.6 Subset Selection in Type-II Hybrid ARQ/FEC for Video Multicast with Multiple subsets 96

6 Conclusion and Future Work 99

A Performance Modeling of the Two-stage Encoder 102

A.1 Modeling of the motion vector estimation error 102

A.2 Modeling of the difference between two motion vector fields 106

B User Guide 110

B.1 Files on the DVD 110

Bibliography 114

Index 120

List of Tables

2.1 PSNR in dB of the JM 12.4 encoder (JM), our two-stage encoder (2-ST), and their difference (Δ) when the losses happen independently, with rate control turned on. Encoded bit rate is shown in the top row of the table. 41

2.2 PSNR in dB of the JM 12.4 encoder (JM), our two-stage encoder (2-ST), and their difference (Δ) over a Gilbert channel with $L_B = 2.5$, and with rate control turned on. Encoded bit rate is shown in the top row of the table. . . 41

2.3 PSNR in dB of the JM 12.4 encoder (JM), our two-stage encoder (2-ST), and their difference (Δ) over measured packet traces from [60], with rate control turned on. Encoded bit rate is shown in the top row of the table. 42

2.4 Motion search methods in five configurations of the JM encoder. 42

2.5 Average encoding time per frame (in milliseconds) for various JM configurations (JM-1 through JM-5), and our two-stage encoder (two-stage). At at the first stage, the encoder uses JM-1 for motion estimation 42

3.1 PSNR performance in dB of different concealment algorithms when RTT = 60 ms. The gain of the proposed method over the one in [14] is also shown. . 58

3.2 PSNR performance in dB of different concealment algorithms when RTT = 120 ms. The gain of the proposed method over the one in [14] is also shown. . 58

3.3 Average time required to conceal one frame, rounded up to the nearest millisecond. 59

4.1 PSNR in dB of system **A** (JM 12.4 encoder (JM)+Belfiore's concealment), system **B** (JM 12.4 encoder (JM)+our noncausal error concealment), system **C** (our two-stage encoder (two-stage)+Belfiore's concealment), and system **D** (our two-stage encoder (two-stage)+our noncausal error concealment) when the losses happen independently, with rate control turned on. Encoded bit rate is shown in the top row of the table. 62

4.2 PSNR in dB of system **A** (JM 12.4 encoder (JM)+Belfiore's concealment), system **B** (JM 12.4 encoder (JM)+our noncausal error concealment), system **C** (our two-stage encoder (two-stage)+Belfiore's concealment), and system **D** (our two-stage encoder (two-stage)+our noncausal error concealment) over a Gilbert channel with $L_B = 2.5$, and with rate control turned on. Encoded bit rate is shown in the top row of the table. 63

4.3 PSNR in dB of system **A** (JM 12.4 encoder (JM)+Belfiore's concealment), system **B** (JM 12.4 encoder (JM)+our noncausal error concealment), system **C** (our two-stage encoder (two-stage)+Belfiore's concealment), and system **D** (our two-stage encoder (two-stage)+our noncausal error concealment) over measured packet traces from [60], with rate control turned on. Encoded bit rate is shown in the top row of the table. 63

4.4 PSNR difference of different systems with system **A** in dB when the losses happen independently, with rate control turned on. Encoded bit rate is shown in the top row of the table. System **A** is (JM 12.4 encoder (JM)+Belfiore's concealment), system **B** is (JM 12.4 encoder (JM)+our noncausal error concealment), system **C** is (our two-stage encoder (two-stage)+Belfiore's concealment), and system **D** is (our two-stage encoder (two-stage)+our noncausal error concealment) 64

4.5 PSNR difference of different systems with system **A** in dB over a Gilbert channel with $L_B = 2.5$, with rate control turned on. Encoded bit rate is shown in the top row of the table. System **A** is (JM 12.4 encoder (JM)+Belfiore's concealment), system **B** is (JM 12.4 encoder (JM)+our noncausal error concealment), system **C** is (our two-stage encoder (two-stage)+Belfiore's concealment), and system **D** is (our two-stage encoder (two-stage)+our noncausal error concealment) 64

4.6 PSNR difference of different systems with system **A** in dB over measured packet traces from [60], with rate control turned on. Encoded bit rate is shown in the top row of the table. System **A** is (JM 12.4 encoder (JM)+Belfiore's concealment), system **B** is (JM 12.4 encoder (JM)+our noncausal error concealment), system **C** is (our two-stage encoder (two-stage)+Belfiore's concealment), and system **D** is (our two-stage encoder (two-stage)+our noncausal error concealment) 65

4.7 Comparison between computation complexity of different systems. 66

5.1 CPU usage by the two subset selection algorithms. 87

List of Figures

1.1 Abstraction of a Video Communication System 2

1.2 Standard video coder based on motion compensation. 4

1.3 Structure of the H.264/AVC 6

1.4 RPS-ACK 13

1.5 RPS-NACK 14

1.6 Binary symmetric channel (BSC). A transmitted 0 may be detected as 0 or as 1, or vice versa 17

1.7 Erasure Channel. One transmitted symbol may be received correctly or may be lost 17

1.8 (n, k, L) convolutional encoder 18

1.9 Packet structure for error protection using FEC and CRC 19

2.1 Standard video coder based on motion compensation. 24

2.2 The proposed two-stage encoder. 26

2.3 Estimation of motion vector using different references. To estimate the solid motion vector, Frame $\hat{Y}_1$ is used as the reference. And, to estimate the dashed motion vector, Frame $\hat{Y}_2$ is used as the reference which causes motion vector displacement 30

2.4 The difference between motion vectors when using two differently quantized reference frames. QP_1 is the quantization parameter for one of the references and QP_2 is the quantization parameter of the other one. Measured data are directly measured in the encoder and Estimated data are calculated using (A.18). 31

2.5 Estimated $\Delta R_{2,1}$(extra bits/pixels). Measured data represents measured values of $\Delta R_{2,1}$ when a two-stage encoder is used compare to the standard encoder, and Estimated data are calculated using (2.7). QP_1 is the quantization parameter for the first stage and QP_2 is the quantization parameter for the second stage. 33

2.6 Quality degradation when QP in the first stage (QP1) is different from the QP in the second stage. (a) *Bus*, (b) *Crew*, (c) *Foreman*, and (d) *Soccer*. . . . 39

2.7 Bit-rate increase when QP in the first stage (QP1) is different from the QP in the second stage. (a) *Bus*, (b) *Crew*, (c) *Foreman*, and (d)*Soccer*. 40

3.1 Causal MV recovery: original MV from frame $t-3$ to frame $t-1$ shown by solid line, recovered MV from frame $t-1$ to frame t shown by dashed line. . 47

3.2 Anti-causal MV recovery: original MV from frame $t+1$ to frame $t+2$ shown by solid line, recovered MVs for the two lost frames, t and $t+1$, shown by dashed line. 48

3.3 Noncausal MV recovery: original MV from frame $t-1$ to frame $t+2$ shown by solid line, recovered MV from frame $t-1$ to frame t shown by dashed line. 50

3.4 Boundary matching. 52

3.5 Frame 144 of *Soccer*. (a) Loss-free decoded frame, (b) Reconstructed by frame copy (frame 144 is lost) (c) Reconstructed by the method from [14] (frame 144 is lost), and (d) Reconstructed by the proposed method (frame 144 is lost). 55

3.6 Frame 145 of *Soccer*. (a) Error free decoded frame, (b) Reconstructed by frame copy (frame 144 is lost), (c) Reconstructed by the method from [14] (frame 144 is lost), and (d) Reconstructed by the proposed method (frame 144 is lost). 56

3.7 The difference between original version of Frame 144 and its erroneous version. (a) Reconstructed by frame copy (frame 144 is lost), (b) Reconstructed by the method from [14] (frame 144 is lost), and (c) Reconstructed by the proposed method (frame 144 is lost). 57

3.8 The difference between original version of Frame 145 and its erroneous version. (a) Reconstructed by frame copy (frame 144 is lost), (b) Reconstructed by the method from [14] (frame 144 is lost), and (c) Reconstructed by the proposed method (frame 144 is lost). 57

5.1 Probability of retransmission 70

5.2 Comparison of ARQ and Hybrid ARQ/FEC type-II.($N = 12$) 72

5.3 Motivation for subset selection in Type-II hybrid ARQ/FEC. 74

5.4 Different frames have different importance for the decoded video quality. . . . 77

5.5 Creating subgroups of receiver for gathering feedback from receivers when the server deals with a large group of clients 78

5.6 Lost frames indicated by darker shading. 79

5.7 ARQ may choose to retransmit any of the lost frames, say the ninth frame, since it cannot distinguish their importance. In this particular case, there is no benefit even to user #3, because the ninth frame depends on the fifth frame, which is also lost. 80

5.8 RaDiO and CoDiO schedulers retransmit the fifth frame of the GOP, which leads to the recovery of 14 frames across the users. In this case, only users #2 and #3 benefit from the retransmission. 81

5.9 The optimal subset for parity generation is composed of the first, fifth, and eleventh frame of the GOP. Reception of the parity packet at all users would lead to the recovery of 26 frames. 81

5.10 Performance comparison of different error control schemes as a function of packet loss probability, with one error control packet per GOP. 84

5.11 Performance comparison of different error control schemes as a function of packet loss probability, with two error control packets per GOP. 85

5.12 Performance comparison of different error control schemes as a function of the number of users, with one error control packet per GOP. 86

5.13 Average PSNR in dB for the *Foreman* sequence with one error control packet per GOP. 87

5.14 Gilbert Model. p_{BG} be the probability of going to a G state from a B state, and p_{GB} is the probability of going to a B state from a G state 89

5.15 Error distribution for different packet loss patterns, when the packet loss rate is 10%. 90

5.16 A simple subset protection for pure FEC. In this scheme n_u information packet are left unprotected, and the remaining information packets are protected with n_c parity packets 91

5.17 The residual packet loss probability in the *subset selection without feedback* scheme for channel **model 1** 93

5.18 The achieved coding gain in the *subset selection without feedback* scheme for channel **model 1** 94

5.19 The probability of having m_c lost packets among n_c parity packets can be found using $\pi(m+1, n+1)$ when the last protected packed is received correctly (top image) or using $\bar{\pi}(m+1, n+1)$ when the last protected packed is lost (bottom image) 95

5.20 Motivation for subset selection in Type-II hybrid ARQ/FEC with multiple subsets. 97

6.1 Hybrid structure for streaming server. In default, this structure uses pre-compressed video stream. But whenever a frame uses an invalid reference it switched to two-stage encoder for fast RPS. 101

A.1 Two equivalent linear space invariant (LSI) systems that can produce $\psi(x, y)$ 108

Preface

First I would like to thank Dr. Ivan Bajic, for providing me the great opportunity of working with him and learning a lot from him. I greatly enjoyed and learned from his structural methodology in research. I would like to thank Dr. Jie Liang and Dr. Mohamed M. Hefeeda for reading my thesis and helping me to enhance it. I would like to thank Dr. Atousa Haj Shirmohammadi for reading my thesis and chairing the defense.

I would like to thank my fellow graduate students in multimedia communication research group at Simon Fraser University.

I would also thank my parents and my beloved wife for their great support and providing a wonderful condition for me to continue my graduate study.

Chapter 1

Background

In recent years, the development of different types of networks, such as the Internet, wireless and mobile networks, has created new applications for new and popular communication facilities. Some of the most interesting applications are based on multimedia and video communications. Over the past decade some of these technologies appeared and became very popular, such as YouTube [66]. YouTube is a video sharing website, which has been launched on February 2005, and just one year later was bought by Google for 1.65 billion dollars [2]. Based on the Alexa reports [1], it has the third highest traffic rank among all websites in the Internet, and about 18 percent of the Internet users visit YouTube at least once a day.

Compression is an essential component of digital video in the current technology world, because uncompressed digital video requires a huge amount of storage to be stored on the disks, and huge amount of bandwidth to be transmitted from one point to another. As an example, transmitting one second of digital CIF video with 30 frames per second requires 36.9 Mbits/sec bandwidth approximately, and for digital HD video, it requires more than 620 Mbits/sec. These numbers become more comprehensible when we compare them with the common size of storage and also with the available bandwidths of common communication

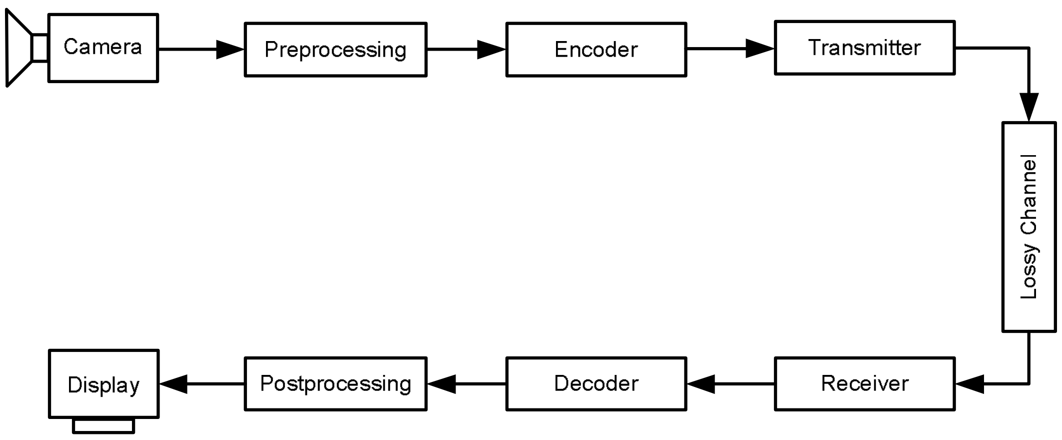


Figure 1.1: Abstraction of a Video Communication System

systems. As an example, a 500 Gbyte hard disk can store about thirty hours of uncompressed CIF video, or less than two hours of uncompressed HD video. As another example, by using an *IEEE 802.11n* (300 Mbits/sec) wireless network, it takes about 7.4 minutes to download one hour of uncompressed CIF video, and about 2 hours is required to download one hour of uncompressed HD video.

Considering the importance of the compression for video technologies, in this chapter we first talk about the basics of video coding in section 1.1, then we introduce H.264/AVC standard as the state of the art video compression standard in section 1.2. It should be noted that H.264/AVC is used for video compression in this thesis.

Furthermore, most of popular channels were not specialized for video transmission. Figure 1.1 shows an abstraction of a packet-based video transmission system. The main challenge for the system in Figure 1.1 is that packet losses may happen in the lossy channel. Almost every channel has errors or packet losses, and compressed video is very sensitive to packet loss. To solve the problem of packet errors or packet losses, the application can use Forward Error Correction (FEC) [59] or Automatic Repeat reQuest (ARQ) [54] such as in TCP protocol in IP networks . Conventional downloading applications mainly use ARQ to provide very robust file transfer (e.g. File Transfer Protocol (FTP)) [54]. Even though downloading protocols can offer robust data transfer, they suffer from two serious problems when used by a multimedia application; their receivers may need to have large buffers, and users need to wait for a long time before starting playback.

To solve problems with packet losses in a video communication system, several techniques have been proposed [54]. These error control techniques are specialized for streaming of video over lossy channels, and they consider video streaming systems' requirements. Generally, we can divide these techniques into three different categories; Error Resilience (section 1.3), Error Concealment (section 1.4), and Channel Coding Techniques (section 1.5).

1.1 Video Compression Concepts

Data compression is the process of reducing the number of bits required for representation of the data by removing its redundancy. It can be either lossless or lossy. In lossless data compression, the original data can be perfectly recovered using the compressed data. However, in most cases lossless data compression can not achieve suitable compression ratios for video applications. Since Human Vision System (HVS) can not perceive fine details of the video, lossy video compression with higher compression ratios can be applied in most video applications [47]. Generally, there are three different types of redundancy in raw video, namely statistical redundancy, spatial redundancy, and temporal redundancy. Statistical redundancy is the hidden redundancy in the pixel histograms, and can be easily removed by entropy encoding (lossless encoding). Spatial redundancy is the hidden redundancy between the neighboring pixels and usually can be removed by spatial transformations such as Karhunen-Loeve Transform (KLT), or Discrete Cosine Transform (DCT). Statistical redundancy and spatial redundancy are in common between still images and videos. The last type of redundancy is temporal redundancy. Usually, there is a strong similarity and large redundancy between neighboring video frames. The basic idea behind video compression algorithms is to take advantage of motion estimation and motion compensation to remove

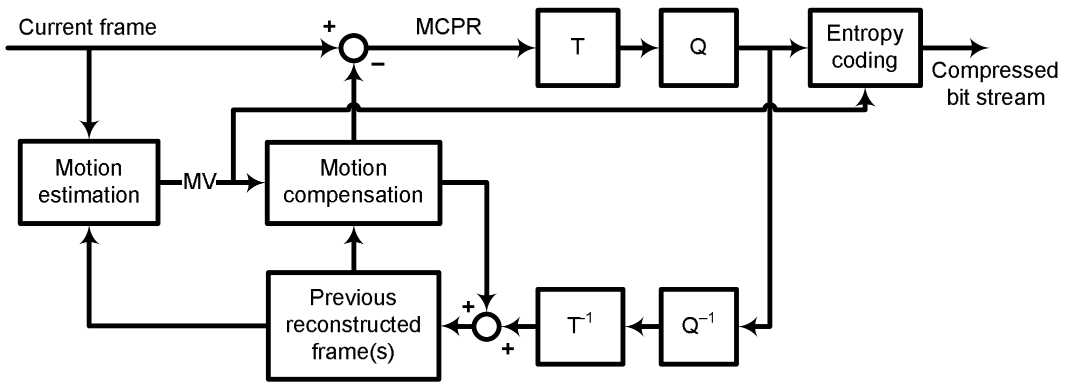


Figure 1.2: Standard video coder based on motion compensation.

this redundancy [59].

Figure 1.2 demonstrates an effective and widely used video encoder structure, which has been adopted in most video encoding standards. The demonstrated structure is very successful in removing all three types of redundancy mentioned above.

In this encoder, the *motion compensation* module predicts input frames using their neighboring frames and corresponding motion vectors, to remove temporal redundancy. These motion vectors are estimated by motion estimation module in the encoder. Since the prediction may not be exactly the same as the original frame, the encoder subtracts the predicted frame from the original frame to find its *Motion Compensated Prediction Residual* (MCPR), and encodes the resulting MCPR.

The encoder performs a transform such as *Discrete Cosine Transform* (DCT) to remove the spatial redundancy of the video signal. Then, to achieve a higher compression ratio, quantizes the transformed MCPR coefficients, then performs entropy coding to the transformed and quantized MCPR coefficients. The encoder can control its output's bitrate by adjusting a quantizer .

On the decoder side, the decoder reconstructs received frames by using their motion vectors, MCPR and the previously received reference frames. The reference frames in the decoder are usually different from their original version in the encoder, because they have been quantized before transmission. If the encoder uses one version of a reference frame to estimate a frame and the decoder uses a different version to reconstruct it, the output of the decoder will encounter a cumulative error known as drift. To prevent this problem, there is always a feedback path in the encoder, which simulates the decoder operations. The feedback path performs dequantization, inverse transform, reconstructs the encoder frame, and stores in the reconstructed frame buffers. Subsequently, stored frames in the *previous reconstructed frames buffer* will be used by motion estimation and motion compensation modules.

1.2 H.264/AVC Overview

H.264/AVC is the latest coding standard from *ITU-T Video Coding Expert Group* (VCEG) and the *ISO/IEC Moving Picture Expert Group* (MPEG) [47]. The VCEG and the MPEG developed H.264/AVC to present a new digital video coding standard with a superior compression performance. Also, H264/AVC is a *network-friendly* video representation for conversational applications such as video telephony or video conferencing, and non-conversational applications such as storage and broadcasting [63].

Many video coding standards had been proposed prior to H.264/AVC, such as MPEG-1, H.261, H.262 (MPEG-2), H.263, H.263+, and MPEG-4 (Part 2). However, growth of different networks, such as the Internet, wireless networks, and increasing popularity of video applications provide a demand to increase the coding efficiency while maintaining high quality. Finally in late 2001, the VCEG, and the MPEG-ISO/IES formed a joint video team (JVT). In March 2003, JVT published the first draft of H.264/AVC [56, 63].

The published standard by the JVT only describes the syntax of the encoded bitstream, and enforces every H.264/AVC decoder to understand this bitstream syntax. This sort of explanation of the standard allows developers to adapt their implementations to their application, and explore trade-off between compression performance and required resources.

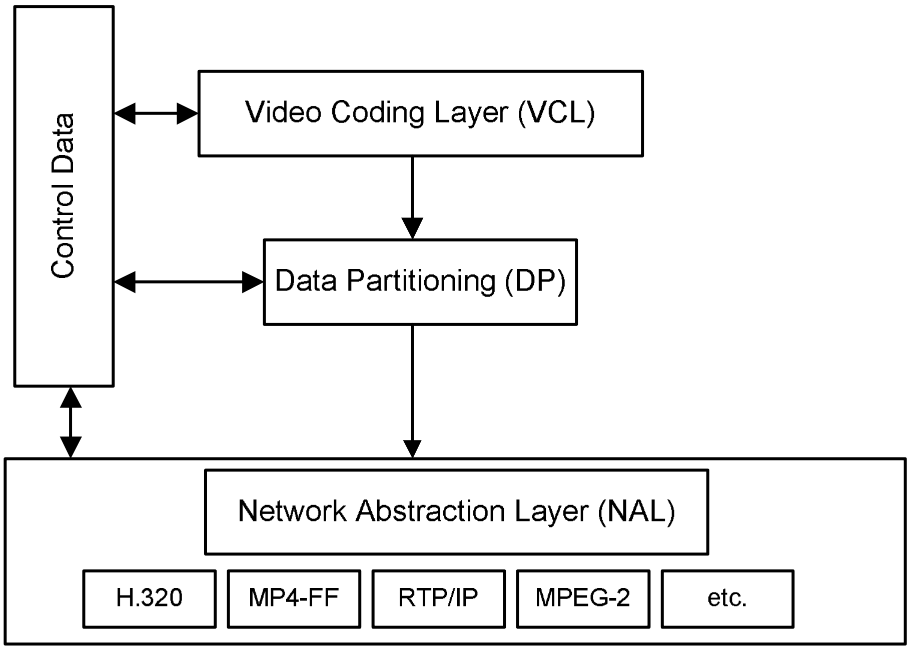


Figure 1.3: Structure of the H.264/AVC

1.2.1 Features

H.264/AVC has been designed for a variety of application such as

- Broadcasting over cable, satellite, and the Internet.
- Interactive or serial storage on CD and DVD.
- Conversational services over the Internet, LAN, and wireless mobile networks.

Managing this variety of applications requires a large degree of adaptability in the codec. To deal with this need, H.264/AVC has two separate parts. The first part is Video Coding Layer (VCL), which deals with compression/decompression of the video content, and the second part is Network Abstraction Layer (NAL), which formats VCL data for delivery over a variety of transportation layers or storage technologies (Figure 1.3) [63].

In H.264/AVC, the Network Abstraction Layer can map the VCL data into different transport layer formats such as:

- RTP/IP for video streaming over IP networks,
- File format such as ISO MP4 for storing on disks,
- H.32X for conversational services,
- and MPEG-2 system format for digital TV broadcasting.

The NAL in H.264/AVC encapsulates the VCL data in NAL units. NAL units contains an integer number of bytes. The first byte of each NAL unit works as the header of the NAL unit, and remaining bytes are the payload. Some output formats such as MPEG-2, use NAL unit as a byte-stream, and some other transporters such as internet protocol (IP) or real time transport protocol (RTP) systems encapsulate NAL units in their own packet formats. The next important duty for the NAL is transmission of encoding parameters to the decoder such as video format, entropy coding type, instantaneous decoding refresh (IDR) flag, end of sequence flag, etc.

Comparing to the prior video coding standards, H264/AVC offers a better compression performance, because of some improvements on its prediction structure. Some of the most important of them are listed bellow:

- Hierarchal variable block-size motion compensation with support of small block size, which are 16x16, 16x8, 8x16, 8x8, 8x4, 4x8, and 4x4 for image luma (and 8x8, 8x4, 4x8, 4x4, 4x2, 2x4, and 2x2 for image chroma),
- Subpixel motion vector accuracy up to 1/4 pixel,
- Multiple reference picture motion compensation,
- Decoupling of referencing order from displaying order,
- Allowing encoder to use B-frames for prediction,
- Weighted prediction,

- Directional spatial prediction for Intra coding,
- Loop Filter,
- Context adaptive entropy coding: H.264 supports two modes, Context Adaptive Binary Arithmetic Coding (CABAC), and Context Adaptive Variable-Length Coding (CAVLC).

Also, comparing to prior standards, H.264/AVC has more robustness to loss and errors, and offers more flexibility with different networks and communication infrastructures. These advantages are mainly because of the following new options of H.264/AVC.

- Parameter set structure,
- Flexible Macroblock Ordering (FMO),
- Arbitrary Slice Ordering (ASO),
- Redundant pictures,
- Data partitioning,
- SP/SI synchronization/switching pictures,
- Flexible slice size.

Going through the details of H.264 in not the focus of this thesis. Interested readers can refer to [47] [59], and [63] for comprehensive details of H.264/AVC.

1.3 Error Resilience

Generally in error resilience techniques, the encoder tries to increase the robustness of the encoded bitstream against the error prone communication channel. In contrast to the channel coding techniques, in an error resilience technique, the encoder does not directly add redundancy such as parity packets, or retransmitted packets. Usually in an error resilient encoder, by changing its prediction process, the encoder removes less redundancy, which helps the decoder to mitigate some effects of errors or losses by using embedded redundancy in the bitstream. Depending on the existence of the feedback channel, two types of error resilience can be developed. When the video communication system does not have access to the feedback channel, the encoder can use intra-refresh, possibly with loss-aware rate-distortion optimization to decide on intra-refresh strategy [20, 70]. On the other hand, when the feedback channel is available, the system can use Reference Picture Selection (RPS) to prevent error propagation.

1.3.1 Error Resilience Tools in H.264/AVC

Unlike some early video coding standards, H.264/AVC provides powerful error resilience tools, which help the video communication system to work in error prone environments. Since most earlier video coding standards such as MPEG-1, and MPEG-2 were developed to store video on physical storage devices, developers only considered some simple error resilience features for them. However, modern video coding standards are also designed for video transmission over communication channels. The focus of this part of the thesis is on describing some important embedded tools of H.264/AVC, which allow us to have error resilient H.264/AVC bitstreams.

Slice Coding

Slice is a set of MacroBlocks (MB), which are grouped together and encoded independently of other slices of the current frame. Since slices are encoded independently, when a slice is lost, the decoder is still able to decode the affected frame partially. Also, using multiple slices allows the decoder to remain synchronized with the encoder, if at least one slice per frame is received correctly. However, using multiple slices degrades the performance of spatial redundancy removal in the encoder.

Flexible Macroblock Ordering (FMO)

In some earlier video coding standards, MBs were encoded in a raster scan order, which starts from the top-left of the frame and continues to the bottom-right side. Therefore, if a slice is lost, a large connected area needs to be concealed in the decoder. Error concealment techniques in the decoder use the information of the neighboring MBs to conceal the erroneous area. Hence, concealment of a large connected area may be difficult. Using an appropriate FMO can spatially interleave MBs to increase the performance of error concealment algorithms in the decoder. Similarly to multiple slices, using FMO can decrease the performance of spatial redundancy removal at the encoder [16, 61].

Data Partitioning (DP)

Different parts of the H.264/AVC bitstream have different importance for the decoder. For example, the decoder can not use encoded motion vectors and residuals without headers, nor can it use coded residuals when motion vectors are not present. Therefore in H.264, encoder can divide the coded bitstream in up to three parts: headers, coded motion vectors, and coded residuals. The transmitter can protect these three parts differently. For example, the transmitter tries its best to protect headers, because, the decoder can keep its synchronization with the encoder only when it has headers available. The transmitter protects coded motion vector with the next lower degree of protection, and protects coded residuals with the lowest protection level.

Redundant Slices

The H.264/AVC encoder may generate and transmit redundant slices. If the original version of the slice is lost, the decoder has the chance to receive its redundant version, and keep the frame quality at a higher lever. Redundant Slices are more useful when there is Region Of Interest (ROI) in the video and the encoder wants to provide a better error resilience for that region [9].

Flexible Reference Frame Concept

In contrast to the earlier standards, in H.264, the encoder can use any arbitrary combination of the previously encoded frames as reference frames (the only limitation is the available memory in the decoder for reference buffering). This ability helps the encoder to exploit more temporal redundancy, but does not directly increase the error resiliency of the bitstream. However, this ability enables other error resilience algorithms (e.g. Reference Picture Selection) to have a comprehensive control over the usage of the reference frames in motion compensated prediction.

Instantaneous Decoder Refresh (IDR)

When a packet is lost, the error can easily propagate through future frames. The simplest solution is that the encoder uses an Instantaneous Decoder Refresh frame and encodes the next frame as an Intra frame, to stop error propagation. Since the encoder can not remove temporal redundancy of the intra coded frames, they need more bandwidth and reduce the compression efficiency significantly. Additionally, Intra frame are more sensitive in error prone environments, because they need more packets, or longer packets. In practice, longer packets are more sensitive to channel errors.

1.3.2 Reference Picture Selection (RPS)

When a feedback channel is available, the receiver can inform the transmitter about the "received" or "not received" packets. Therefore, the transmitter can take an action, and reduce error effects. One solution is using retransmission-based methods such as ARQ, but these techniques may lead to a very long latency, which is not appropriate for video streaming. Another solution is “Instantaneous Decoder Refresh,” as described above. IDR can stop error propagation very fast and does not provide long latency as ARQ does. Intra coding can not exploit temporal redundancy between frames, therefore it will impair coding efficiency. Also, I-frames are longer than P-frames and B-frames; therefore they are more sensitive to channel errors.

In RPS, the encoder uses the feedback information to manage the usage of reference frames in motion compensation. In contrast to ARQ and IDR, because RPS can remove some temporal redundancy, RPS can stop error propagation without a significant loss of coding efficiency, and does not add any extra latency. Simulations of RPS have demonstrated its effectiveness and advantages against FEC, ARQ, and intra-refresh [29, 38, 40, 70]. Two main types of RPS have been proposed, RPS-ACK and RPS-NACK.

RPS-ACK

The first type of RPS is RPS-ACK, in which the decoder sends a positive acknowledgement (ACK) to the encoder for each correctly received frame, so the encoder uses only those frames that have been flagged as correctly received for motion compensation. This is illustrated in Figure 1.4. In Figure 1.4, the acknowledgment for receiving frame t is received before encoding of frame $t + 2$, therefore encoder uses frame t for encoding of frame $t + 2$. Similar thing happens for frames $t+3$ and $t+4$, because the acknowledgments are received on time. Now, assume that frame $t + 3$ has been lost during transmission, therefore the encoder does not receive any acknowledgment for frame $t + 3$, when it wants to encode frame $t + 5$. Thus, the encoder uses frame $t + 2$ to encode frame $t + 5$. In this example, the decoded video only has distorted frame $t + 3$, and errors will not propagate further.

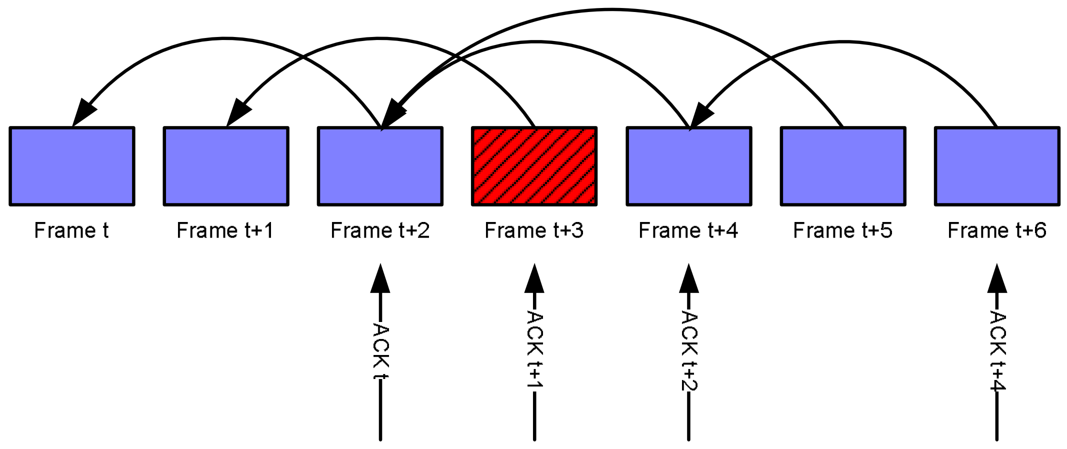


Figure 1.4: RPS-ACK

RPS-NACK

The second type of RPS is RPS-NACK, in which the decoder sends a negative acknowledgment (NACK) for each damaged or lost frame. The encoder then avoids using those frames for motion compensation. This is illustrated in Figure 1.5. Here, the encoder uses frame t to encode frame $t+1$. The encoder behaves similarly for encoding $t+2$, $t+3$, and $t+4$. Since frame $t+3$ has been lost, the decoder can not use it to decode frame $t+4$, so both frames $t+3$ and $t+4$ will be distorted. But the encoder receives the negative acknowledgement about frame $t+3$ before starting to encode frame $t+5$, therefore, uses frame $t+2$ for encoding frame $t+5$ and stops error propagation at this point.

In RPS-ACK, the encoder never uses an erroneous frame as a reference, hence there is no error propagation. However, compression efficiency is reduced because the encoder often uses temporally distant reference frames for motion compensation. Typically, the number of correctly received frames is much higher than the number of damaged/lost frames, therefore the number of ACKs in RPS-ACK is much higher than the number of NACKs in RPS-NACK. Hence, RPS-NACK is more efficient than RPS-ACK in term of bandwidth consumption, both on the forward and backward channel.

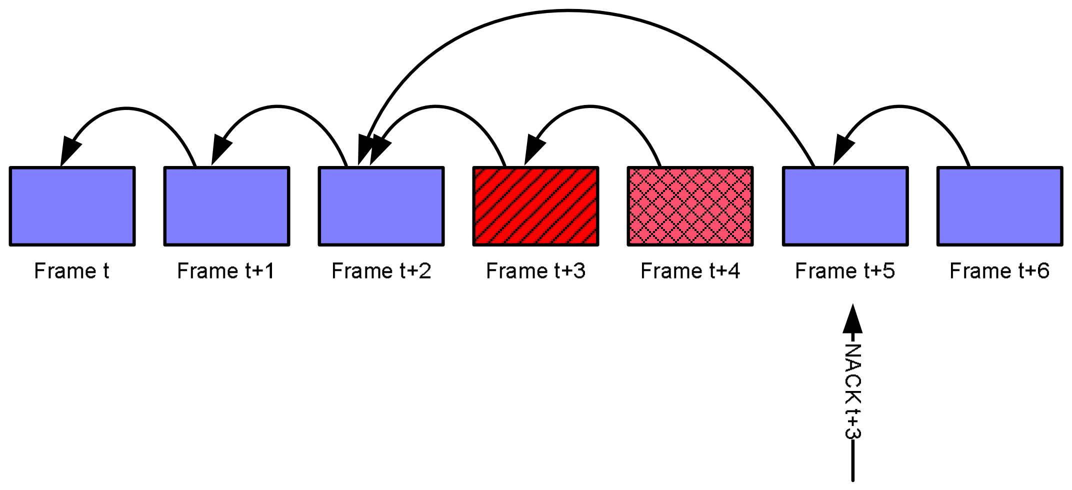


Figure 1.5: RPS-NACK

1.4 Error Concealment

Coding efficiency always suffers from using channel coding techniques and error resilience methods, because they add redundancy, or do not remove some parts of the existing redundancy from the encoded bitstream. In contrast, without suffering from degradation of coding efficiency, error concealment techniques hide error and loss effects at the decoder side using the remaining redundancy in the received bit-stream. The redundancy may exist in temporal domain [34], spatial domain [46], or spatiotemporal domain [17]. Consequently, three different categories of error concealment have been developed.

1. **Spatial Error Concealment** Spatial concealment techniques usually use the spatial smoothness property of the video signal and attempt to use the information from neighboring MBs to interpolate the corrupted pixels. Sharp edges and complex textures disrupt the smoothness of the video signal. In such cases, an edge preserving smoothness criterion is needed to interpolate the corrupted pixels [26].

2. **Temporal Error Concealment** Temporal concealment algorithms usually reestimate motion vectors (MVs) of the corrupted macroblocks (MBs) to be used in motion

compensation.

3. **Spatiotemporal Error Concealment** Spatiotemporal error concealment techniques use both remaining temporal redundancy and spatial redundancy in the bitstream to recover damaged parts of a video.

1.4.1 Whole Frame Concealment

In low bit rate coding of low resolution (e.g., QCIF) video, each encoded frame typically fits within a single network packet, so the loss of a packet during transmission of compressed bit-stream may cause the loss of a whole video frame [8]. Even in high bit rate scenarios, traffic congestion causes burst errors which may also lead to whole-frame loss. When a whole-frame loss happens, the error concealment algorithms mentioned above are of limited use, because most of them use information from neighboring MBs to recover the motion vector information in temporal concealment, or to interpolate the corrupted pixels in spatial concealment. Therefore, to deal with a whole-frame loss, the concealment algorithm needs to use a different strategy. In [14], Belfiore et al. proposed a novel whole-frame concealment method, which is based on multi-frame optical flow estimation. This algorithm estimates the motion of the lost frame by extending the motion vectors of the last decoded frame and projecting that frame onto the lost frame. In [14], it was reported that this algorithm outperforms the simple *frame* copy method in the JM H.264 decoder by several dBs in PSNR. However, it is too complex to run in real time, because it uses several median filtering operations in the pixel domain. To solve the high complexity problem of the method from [14], in [8] Baccichet et al. proposed a block-based whole-frame concealment which can work in real time at the expense of losing some quality.

1.5 Channel Coding

Shannon's channel coding theorem states that if the data entropy is lower than the channel capacity, then a channel code can be found to guarantee arbitrary small error rate [52]. Two important classes of channel coding schemes have been proposed. The first class is usually called Forward Error Correction (FEC) algorithms, and the second one Automatic Repeat reQuest (ARQ) based algorithms [41, 53].

1.5.1 Forward Error Correction (FEC)

In Forward Error Correction (FEC) algorithms, the encoder generates some redundancy and transmits it along with the data. Based to the amount of redundancy, the decoder can recover lost or corrupted data up to a threshold. Since the loss rate may change in time, the encoder continually needs to adjust its FEC rate. This adjustment is often problematic, since a reliable channel state information is generally not available, so FEC often tends to be either over-designed or under-designed, as noted in [7]. Combination of FEC with long interleaving can, to some extent, mitigate the variable loss rate problem, but the latency caused by the interleaver makes this kind of FEC unsuitable for interactive video services [41].

The error in communication systems usually appears as erroneous detected bits. The simplest model for this type of error is memoryless binary symmetric channel. Figure 1.6 illustrates a memoryless binary symmetric channel with error probability of p. In this channel, an error happens when a 0 is transmitted but detected as 1, or vice versa.

Another famous channel model is the erasure channel. In erasure channel, there are two possibilities for a transmitted symbol; a symbol may be received correctly (with probability $1-p_L$) or it may be not received at all (with probability p_L). The second situation is usually called erasure. Figure 1.7 illustrates erasure channel.

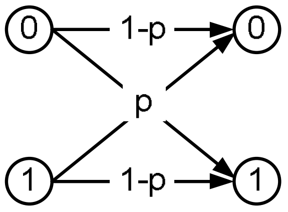


Figure 1.6: Binary symmetric channel (BSC). A transmitted 0 may be detected as 0 or as 1, or vice versa

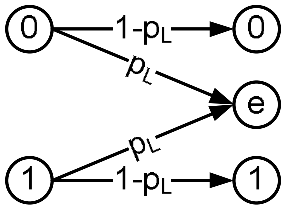


Figure 1.7: Erasure Channel. One transmitted symbol may be received correctly or may be lost

Convolutional Codes

Convolutional codes are one of the most famous forward error correction codes which were first introduced by Elias [22] in 1955. The main application of convolutional codes is bit error correction in the receiver. These codes became popular after a fast and easy way to implement the decoder was proposed by Viterbi in [41, 57]. Figure 1.8 illustrates a (n, k, L) convolutional encoder, which can be implemented with n modulo-2 adders at the output, and no more than Lk bits shift register. In this implementation, the encoder at each step gets k bits and returns n encoded bits at its output.

Reed-Solomon (RS) Codes

Reed-Solomon codes are very popular in competing with erasures. In a $RS\,(n, k)$ code , the encoder adds $n - k$ redundant symbols to the data, and the decoder can recover the original data if it receives k or more symbols. The basic idea of Reed-Solomon codes is over-sampling a polynomial. Assume that we have a k-degree polynomial, we are able to

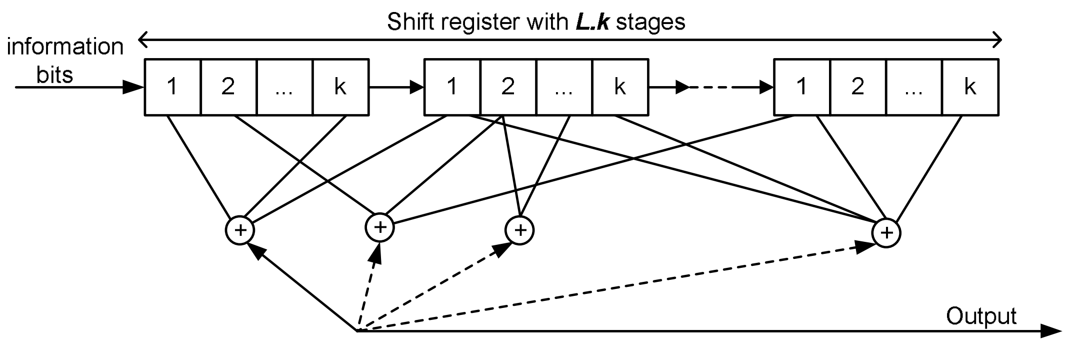


Figure 1.8: (n, k, L) convolutional encoder

recover the polynomial if we have at least k points on the polynomial. In a Reed-Solomon code, the encoder chooses $n > k$ point on a polynomial over a finite field which contains k input points. Then the decoder can recover original points using any k out of n points. Because the performance of Reed-Solomon Codes with short code word length can become very close to the channel theorem limits, they are very popular in combating with packet erasure/loss in communication systems.

Cyclic redundancy check

Communication systems use Error Detection codes to detect bit error at the receivers. Cyclic Redundancy Check (CRC) codes are popular class of error detection codes. One of the main applications of Error Detection codes is converting a channel with bit errors to a channel with erasures. Dealing with erasures is much simpler than dealing with bit errors in many communication systems, and devices usually prefer to only deal with accurate date. But the bit error is the common error that happens in a communication link.

To convert a channel with bit error to an erasure channel, communication systems can use the following scheme. The encoder adds some error detection redundancy to the data for enabling the decoder to check the correctness of the data after decoding (e.g. using CRC). Then, adds some error correction redundancy to enable the decoder to correct possible bit errors. Finally, all this data is sent to the receiver as a packet (Figure 1.9).

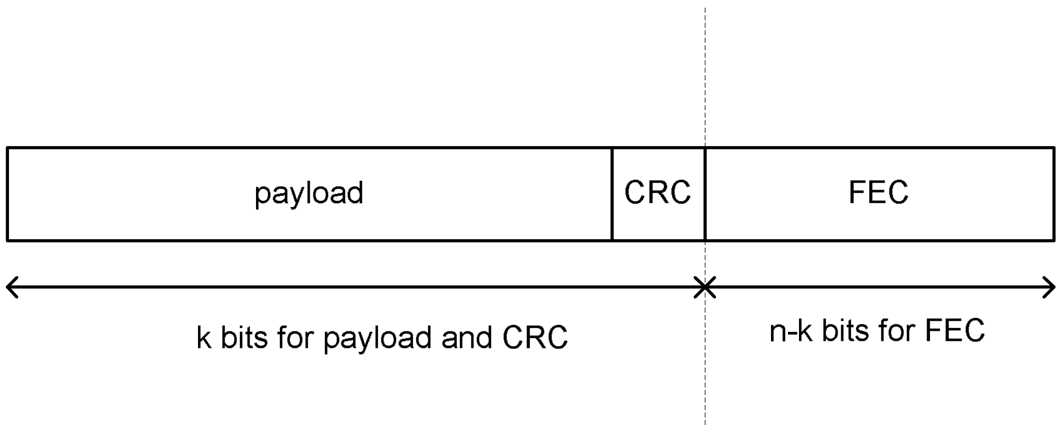


Figure 1.9: Packet structure for error protection using FEC and CRC

At the receiver side, the decoder tries to correct possible errors in the data and CRC. Then, it checks the CRC to make sure that everything is correct. If the CRC determines that there are some bit errors in the received data, the receiver drops the whole packet and signals a packet loss. Figure 1.9 shows the packet structure for this scheme.

1.5.2 Automatic Repeat reQuest (ARQ)

When transmission in a communications system is two-way, the error correction can be achieved using error detection in the receiver and retransmission by the transmitter, called Automatic Repeat request (ARQ). In an ARQ system, when an error is detected in the receiver, a request is sent for the transmitter to repeat the message, and it continues until the message is received correctly. Therefore, performing ARQ does not require any channel information and only requires a two-way channel. Another important advantage of using ARQ over FEC is that the error detection and retransmission is much simpler than encoding/decoding in FEC-based schemes. Conversely, several error-detections and retransmissions may happen until the message is received correctly, which can add up to a long communication delay [41, 58].

1.5.3 Hybrid ARQ/FEC

As we compared ARQ and FEC, each of them has some advantages and some drawbacks. By properly combining of FEC and ARQ, a communication system can use the advantages of both techniques and overcome their drawbacks. Two popular types of this combination have been proposed, called *Type-I Hybrid ARQ/FEC* and *Type-II Hybrid ARQ/FEC* [11, 41].

Type-I Hybrid ARQ/FEC

One hybrid strategy is using FEC as a subsystem which is contained in an ARQ based system. The function of FEC portion is to protect the data for the most frequent errors. Therefore, ARQ is less likely to become a cause for throughput decrease. When an infrequent error happens, the ARQ portion can retransmit the required part of the data. Using ARQ on top of FEC in this scheme prevents over-designing of the FEC part and saves the required bandwidth.

Type-II Hybrid ARQ/FEC

Another hybrid strategy for combining ARQ and FEC involves the transmission of parity packets for the lost or erroneous data. If the receiver detects error or loss in the received data, it can request for parity until the data is received correctly. In unicast scenarios, when there is only one receiver, the performance of this scheme becomes very similar to pure ARQ. The advantage of Type-II Hybrid ARQ/FEC only appears when there is more than one receiver. In chapter 5, we will discuss Type-II Hybrid ARQ/FEC in more detail.

1.6 Contributions of the Thesis

In this thesis, we focus on three major aspects of error control for a video communication system. In chapter 2, we propose a new encoder structure named two-stage encoder for implementing the fast reference picture selection feature. First, in section 2.1, we propose the new structure, then in section 2.2 we model the performance of the proposed encoder structure, and finally in section 2.3 we examine the performance of the two-stage encoder different test conditions. The results indicate that the proposed encoder has compression performance comparable to the JM 12.4 encoder, while achieving significant encoding speed-up at streaming time.

In chapter 3, we propose a new whole-frame concealment algorithm, named noncausal whole-frame concealment. In this framework, the decoder has the ability to extract the information in the preceding and succeeding frames, and use this information to conceal the lost frame. In section 3.1.1, we describe a state-of-the-art whole-frame concealment method from [14], and in section 3.2, we introduce our noncausal error concealment in detail. First, we describe our modification on the optical flow estimation to have the noncausal optical flow, and then we employ spatiotemporal boundary matching algorithm. and finally in section 3.3, we examine the performance of the proposed noncausal error concealment algorithm. The results indicate that the proposed method outperforms the state-of-the-art method in both objective and subjective visual performance, all at a fraction of the computational cost.

In chapter 4, we combine the encoder from chapter 2 and the decoder from chapter 3, and evaluate the performance of the combined system. We compare the combined system with three other video transmission systems in term of their resulting PSNR, time complexity at the server side, and time complexity at the client side. The results indicate the the proposed system is the only system which has a low complexity at both the encoder side and the decoder side. In addition, it has the second rank among other systems, when we compare them in terms of their rate-distortion performance.

In chapter 5, we propose a new coding scheme for video multicasting, named subset selection in Hybrid ARQ/FEC. In section 5.2 we introduce the proposed subset selection algorithm, and in section 5.3 we introduce the optimum full search algorithm and a sub-optimal simulated annealing algorithm for selecting an appropriate subset. In section 5.4, we compare the performance of the proposed subset selection algorithm with some other channel coding schemes. And then in section 5.5, we study the performance of the subset selection algorithm in analytical point of view. The results indicate that subset selection can bring an improvement of 1-1.5 dB in decoded video PSNR compared to a state-of-the-art error control scheme based on rate-distortion optimization.

In chapter 6, we sum up contribution in the thesis and give some suggestion to improve the proposed methods. Also, the thesis has two appendixes. In appendix A, we derive an analytical approach to estimate the accuracy of precomputed motion vector in the two-stage encoder. And in appendix B, we provide a user manual for the code developed during this research.

Chapter 2

Two-Stage Encoder for Fast Reference Picture Selection

Error propagation is a serious problem affecting transmission of predictively-coded video. Even a single erroneous bit can cause error propagation through the whole video, and significantly degrade the video quality at the output. Errors cause received frames at the decoder to be different from the ones that were used in the encoder for motion compensation. Once an erroneous frame gets into the decoder's motion compensation loop, the errors propagate to future frames. This way, an error in one frame may propagate to many future frames, degrading the quality of the decoded video.

Reference Picture Selection (RPS) has been proven as a strong error resilience tool for video streaming applications. But, RPS has rarely been used in commercial video streaming systems. The main reason for rare usage of RPS in commercial applications is that RPS requires a high computational power at streaming time. In this chapter we introduce a new structure for the encoder, which enables fast reference picture selection suitable for real-time implementation. we proposed this structure in [5].

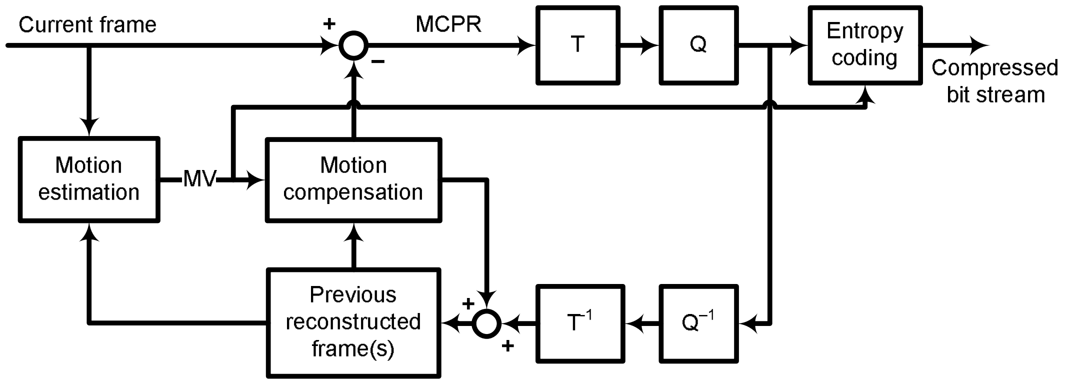


Figure 2.1: Standard video coder based on motion compensation.

2.1 Overview of the Proposed Two-Stage Encoder

Figure 2.1 shows the structure of a standard video coder based on motion-compensated prediction [58]. Incoming video frame (called "current frame") is subject to motion estimation with respect to previous reconstructed frame(s). Using the resulting motion vectors (MVs), a prediction of the current frame is made. The difference between the current frame and the prediction (called motion-compensated prediction residual, MCPR) is then subject to transform, quantization, and entropy coding along with MVs, to create the compressed bit stream. In addition, quantized MCPR is dequantized, inversely transformed, and added to the prediction to create the reconstructed version of the current frame, which is then stored in the frame buffer for future motion estimation and compensation.

Motion estimation can account for up to 80% of the processing time and memory usage in H.264/AVC video coding [39]. However, in typical streaming applications such as video-on-demand, video content is available ahead of streaming time. This gives us the opportunity to perform motion estimation ahead of time, leaving motion compensation, quantization, and entropy coding for real-time operation during the streaming session. Hence, in our proposed two-stage encoder, we reduce real-time processing requirements of the video encoder by decoupling motion estimation from motion compensation.

Let f be the video frame rate and N be the number of previous reconstructed frames stored in the encoder buffer to be used as references for encoding the current frame. Because H.264/AVC encoder uses multi-frame references, the value of N can be greater than one. Since RPS-enabled encoder works based on ACK or NACK messages from the decoder, there is a strong relationship between a suitable value of N and the Round Trip Time (RTT). Hence, parameter N should be chosen depending on the maximum expected RTT.

For a given RTT, the delay (in terms of frames) between sending a frame, and receiving feedback related to that frame, is $n = \lceil RTT \times f \rceil$. Hence, the encoder can only perform RPS when $n < N$. If $n \geq N$, RPS-ACK encoder will reduce to an intra-frame encoder, while the RPS-NACK encoder will insert an I-frame whenever it receives a NACK message, which reduces compression performance significantly. Therefore, the value of N should be chosen to be at least $\lceil RTT_{max} \times f \rceil$. For example, if $RTT_{max} = 110$ ms $= 0.11$ s, and the frame rate $f = 30$ fps, then $N \geq \lceil RTT_{max} \times f \rceil = 4$ frames.

Figure 2.2 shows the structure of the proposed two-stage encoder. In the first stage, the switch connects the motion estimation module to the motion compensation module, and the entropy coder is turned off. The encoder estimates motion vectors (MVs) between each frame and its N previous frames for each coding mode (16×16, 16×8, 8×16, 8×8, 8×4, 4×8, and 4×4 blocks) and stores the computed motion vectors on the storage device. To reduce the amount of memory on the storage, the encoder removes all the modes which use more than one reference for each macro-block and then, among the remaining modes, for each macro-block and for each reference, chooses the one which minimizes the macro-block rate-distortion Lagrangian cost function (2.1). In (2.1), l is the macro-block index, m_l is the mode for the l-th macro-block, J_l is the macro-block rate-distortion Lagrangian cost, D_l is the distortion of the l-th macro-block, R_l is the rate of the l-th macro-block and λ is the Lagrange multiplier. In many H.264/AVC encoders, the Lagrange multiplier λ is defined as

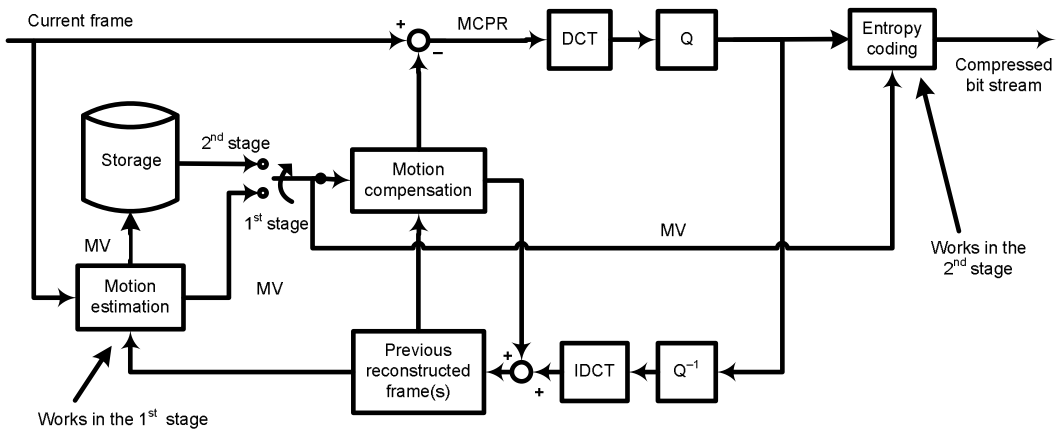


Figure 2.2: The proposed two-stage encoder.

Algorithm 1 First stage of the proposed two-stage encoder (for one macro-block)

```
for all frames($r$) used as reference do
  $J_{r,min} \leftarrow \infty$
  for all modes($m$) that only use frame $r$ as the reference do
    find associated motion vectors $(mv_{m,r})$ and Lagrangian cost function
    $(J(r, m, mv_{m,r}))$
    if $J(r, m, mv_{m,r}) < J_{r,min}$ then
      $J_{r,min} \leftarrow J(r, m, mv_{m,r})$
      $MV \leftarrow mv_{m,r}$
      $MODE \leftarrow m$
    end if
  end for
  $STORE(MV, MODE)$ \\ store motion vectors and mode on the storage device
end for
```

a function of QP (2.2) [62]. The procedure discussed above is implemented as Algorithm 1.

$$J_l(m_l) = D_l(m_l) + \lambda R_l(m_l) \tag{2.1}$$

$$\lambda = 0.85 \cdot 2^{\frac{QP-12}{3}} \tag{2.2}$$

The second stage is the standard motion compensation loop without motion estimation. Now, the switch in Figure 2.2 connects the motion compensation module to the storage. In this case, the motion estimation module is turned off, and the entropy coder is turned on.

Depending on the feedback from the decoder, the encoder chooses the appropriate reference frame for each macro-block from the reference picture buffer, and uses the associated (pre-computed) motion vectors and coding modes to perform motion compensation (Algorithm 2). *Only this second stage needs to run at streaming time.* As demonstrated in the results section, by avoiding motion estimation, the second stage of the two-stage encoder has a much lower complexity than the standard encoder, and is more suitable for real-time operation.

In RPS-ACK, the decoder sends an ACK message for each received frame to the encoder. Initially, all frames are marked as $INVALID$ references at the encoder. When the encoder receives an ACK about a particular frame, it marks that frame as a $VALID$ reference. On the other hand, in RPS-NACK, all frames are initially marked as $VALID$ references. The decoder sends a NACK for each erroneous frame to encoder. When the encoder receives a NACK about a particular frame, it uses Algorithm 3 to mark that frame, and subsequent frames that depend on it, as $INVALID$. Motion compensation in Algorithm 2 then uses only $VALID$ frames as references.

Note that the proposed two-stage encoder generates the compressed video bitstream only in the second stage, while streaming. Prior to streaming, video is stored in the raw (uncompressed) format at the server along with the motion vectors generated in the first stage. Hence, reducing the complexity of the second stage, which needs to run in real time during the streaming session, will cost us some extra storage space at the server. To estimate this extra cost, we can use the following simplified analysis. Suppose that, on the average, one motion vector is assigned to every 8×8 block of pixels in each frame (in reality, larger or smaller block sizes may be used; some blocks are coded in the I-mode without any associated motion information). If the motion search range is ± 127 (an overestimate in most cases), then x- and y-component of a motion vector each take 8 bits (one byte) to be stored. Hence, if the encoder uses $N = 4$ previous frames as references, we get $N \times 2 = 8$ extra bytes for

Algorithm 2 Second stage of the proposed two-stage encoder (for one macro-block)

```
$J_{Inter} \leftarrow \infty$
**for all** frames($r$) used as reference **do**
  **if** $CheckValidity(r) == VALID$ **then**
    $READ\,(MV, MODE)$ \\ read motion vectors from the storage
    compute Lagrangian cost function $J\,(r, m, mv_{m,r})$
    **if** $J\,(r, m, mv_{m,r}) < J_{Inter}$ **then**
      $MODE_{Inter} \leftarrow MODE$
      $MV_{Inter} \leftarrow MV$
      $J_{Inter} \leftarrow J\,(r, m, mv_{m,r})$
    **end if**
  **end if**
**end for**
Find the best Intra-coding mode and its Lagrangian $J_{Intra}$
**if** $J_{Intra} \geq J_{Inter}$ **then**
  Encode the macro-block as a P-block using $MODE_{Inter}$ and $MV_{Inter}$
**else**
  Encode the macro-block as a I-block using $MODE_{Intra}$.
**end if**
```

each group of 64 pixels, which represents an increase of $8/64 = 12.5\%$ for gray-scale video, or $8/96 \approx 8.3\%$ for YUV 4:2:0 video. So the increase in storage requirements is approximately $8\% - 12\%$ with respect to raw video size. Whether the ability to do fast RPS justifies this increase in size, and the fact that the video is not compressed until the streaming begins, depends on the particular application and the associated business model. Availability of cheap storage makes this trade-off very attractive.

Another point that needs to be considered is the issue of quantization of reference frames. Note that the standard encoder of Figure 2.1 estimates the motion between the reconstructed (i.e., quantized) previous frame, and the original (i.e., unquantized) current frame. Since the quantization parameter QP may vary, one may wonder what kind of effect would this have on the resulting MVs, and subsequently on the compression performance. It turns out that different QP values generally lead to different MVs. In subsections 2.2 and 2.3, we study the effect of QP values used in the first stage on the compression performance achieved in the second stage of our two-stage encoder.

Algorithm 3 Encoder receives a feedback in RPS-NACK (frame r has been lost)

```
mark frame $r$ as an $INVALID$ reference.
for $k \leftarrow r+1$ to the last encoded frame do
  if frame $k \neq$ I-frame then
    for all frames $r'$ used as reference of frame $k$ do
      if frame $r'$ is marked as $INVALID$ then
        mark frame $k$ as $INVALID$
      end if
    end for
  end if
end for
```

2.2 Performance modeling of the two-stage encoder

In the second stage of the proposed two-stage encoder, the encoder uses pre-computed motion vectors in the motion compensation module. It is important to analyze the effect of using such motion vectors on the performance of the encoder. Using equation (A.18) (Appendix A) enables us to estimate the covariance matrix of the difference between two sets of motion vectors. To find $\Sigma_{\Delta\vec{mv}_{\hat{Y}_1,\hat{Y}_2}}$ using (A.18), we only need to have two differently quantized versions of the reference frame and the current frame as shown in Figure 2.3 . If these frames are not available in the encoder, one can use a model to estimate the required parameters. To examine the validity of (A.18), we used a modified two-stage encoder to encode several video sequences with different qualities by using differently quantized references. Meanwhile, the encoder stored all different motion vectors to calculate the probability density function of the *difference of motion vectors* estimated using differently quantized reference frames. Figure 2.4 makes a comparison between results generated with the model (A.18), and real data measurement results. QP_1 is the quantization parameter for one of the references and QP_2 is the quantization parameter of the other one. Measured data are directly measured in the encoder and estimated data are calculated using (A.18). In Figure 2.3, when the quantization parameter in the first stage QP_1 is greater than the

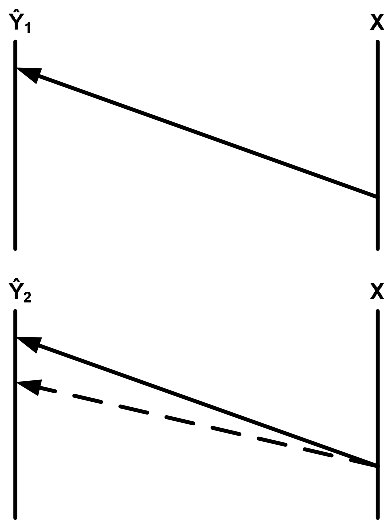


Figure 2.3: Estimation of motion vector using different references. To estimate the solid motion vector, Frame $\hat{Y}_1$ is used as the reference. And, to estimate the dashed motion vector, Frame $\hat{Y}_2$ is used as the reference which causes motion vector displacement

quantization parameter in the second stage QP_2, the model can estimate the motion displacement with 10% error. But, when QP_2 is greater than QP_1 the model can not follow the changes as accurate as previous case because of the approximations that we consider to drive an analytically close form model.

The main difference between the two-stage encoder and other encoders is in using possibly less accurate precomputed motion vectors, therefore the next step is analyzing the performance of using inaccurate motion vectors in the performance of an encoder. Based on the Appendix, we can find the covariance matrix of the error in precomputed motion vectors using (A.18).

When an encoder uses inaccurate motion vectors, the motion compensation module loses its efficiency and generates a residual with higher variance. In [27], Girod showed that the power spectral density of the residuals after motion compensation using inaccurate motion vectors can be estimated by (2.3) and (2.4).

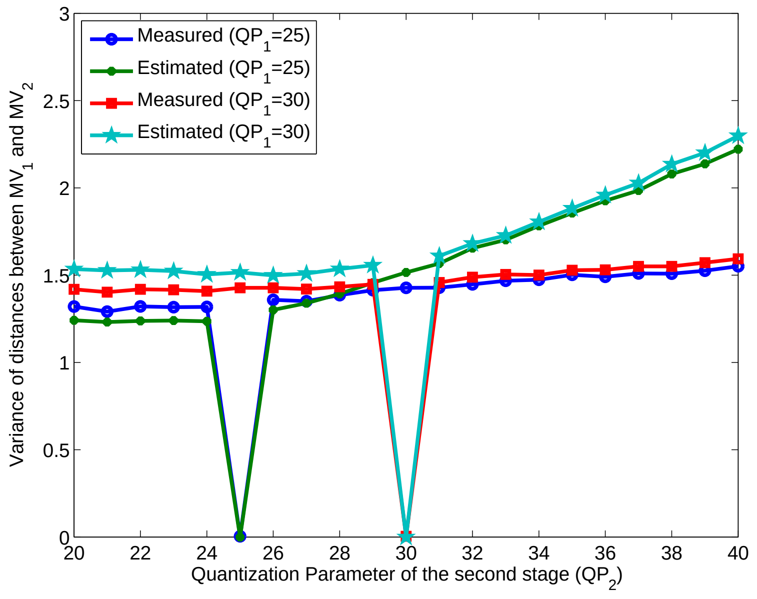


Figure 2.4: The difference between motion vectors when using two differently quantized reference frames. QP_1 is the quantization parameter for one of the references and QP_2 is the quantization parameter of the other one. Measured data are directly measured in the encoder and Estimated data are calculated using (A.18).

$$\begin{aligned}\Phi_{ee_1}\left(\omega_x,\omega_y\right) = \quad & \Phi_{ss}\left(\omega_x,\omega_y\right)\left(1+\left|F\left(\omega_x,\omega_y\right)\right|^2\right.\\ & \left.-2\Re\left\{F\left(\omega_x,\omega_y\right)P_1\left(\omega_x,\omega_y\right)\right\}\right)\\ & +\Phi_{nn}\left(\omega_x,\omega_y\right)\left|F\left(\omega_x,\omega_y\right)\right|^2\end{aligned} \tag{2.3}$$

$$\begin{aligned}\Phi_{ee_2}\left(\omega_x,\omega_y\right) = \quad & \Phi_{ss}\left(\omega_x,\omega_y\right)\left(1+\left|F\left(\omega_x,\omega_y\right)\right|^2\right.\\ & \left.-2\Re\left\{F\left(\omega_x,\omega_y\right)P_2\left(\omega_x,\omega_y\right)\right\}\right)\\ & +\Phi_{nn}\left(\omega_x,\omega_y\right)\left|F\left(\omega_x,\omega_y\right)\right|^2\end{aligned} \tag{2.4}$$

In (2.3) and (2.4), $\Phi_{ee_1}\left(\omega_x,\omega_y\right)$ and $\Phi_{ee_2}\left(\omega_x,\omega_y\right)$ represent the power spectral density (PSD) of the remaining residuals after the motion compensation. $F\left(\omega_x,\omega_y\right)$ is the Fourier transform of the loop filter which is used for deblocking. $P\left(\omega_x,\omega_y\right)$ is the Fourier transform of the error probability distribution in motion vectors which are used during motion compensation.

The power spectral density (PSD) of a frame $\Phi_{ss}(\omega_x, \omega_y)$ can be modeled by (2.5), an isotropic spatial power spectrum [15]. And, $\Phi_{nn}(\omega_x, \omega_y)$ the power spectral density (PSD) of the noise can be modeled as a typical white noise with a flat PSD [28].

$$\Phi_{ss}(\omega_x, \omega_y) = \frac{2\pi\sigma_s^2}{\omega_0^2}\left(1 + \frac{\omega_x^2 + \omega_y^2}{\omega_0^2}\right) \tag{2.5}$$

Consequently, the coder needs more bits to keep the quality constant. In [44], Noll and *et al* showed that when the frame and its motion compensated residual can be modeled as a Gaussian Wide-Sense Stationary (WSS) signals, the maximum achievable bit-rate saving by motion compensation (in bits/pixel) for the encoder can be represented by (2.6).

$$\Delta R = \frac{1}{8\pi^2}\int_{-\pi}^{\pi}\int_{-\pi}^{\pi}\log_2\left(\frac{\Phi_{ee}(\omega_x, \omega_y)}{\Phi_{ss}(\omega_x, \omega_y)}\right)\omega_x.\omega_y \tag{2.6}$$

In (2.6), $\Phi_{ee}(\omega_x, \omega_y)$ and $\Phi_{ss}(\omega_x, \omega_y)$ are the power spectral densities of the motion compensated residual, and original frame respectively. The bit-rate at the output of the encoder is usually different from the suggested value by (2.6), especially because (2.6) does not consider the required bits for encoding and transmitting motion vectors. In this analysis, we want to compare the performance of two encoders. The first one uses estimated motion vector and has $\Phi_{ee1}(\omega_x, \omega_y)$ as the PSD of the MC residual and the second one uses the precomputed motion vetoers and has $\Phi_{ee2}(\omega_x, \omega_y)$ as the PSD of the MC residual. If we assume residuals are optimally encoded and achieve maximum bit-rate saving whit respect to intra coding, the difference of their bit-rate, $\Delta R_{2,1}$ is shown by (2.7).

$$\begin{aligned}\Delta R_{2,1} &= \frac{1}{8\pi^2}\int_{-\pi}^{\pi}\int_{-\pi}^{\pi}\log_2\left(\frac{\Phi_{ee2}(\omega_x, \omega_y)}{\Phi_{ss}(\omega_x, \omega_y)}\right)\omega_x.\omega_y - \frac{1}{8\pi^2}\int_{-\pi}^{\pi}\int_{-\pi}^{\pi}\log_2\left(\frac{\Phi_{ee1}(\omega_x, \omega_y)}{\Phi_{ss}(\omega_x, \omega_y)}\right)\omega_x.\omega_y \\ &= \frac{1}{8\pi^2}\int_{-\pi}^{\pi}\int_{-\pi}^{\pi}\log_2\left(\frac{\Phi_{ee2}(\omega_x, \omega_y)}{\Phi_{ee1}(\omega_x, \omega_y)}\right)\omega_x.\omega_y\end{aligned} \tag{2.7}$$

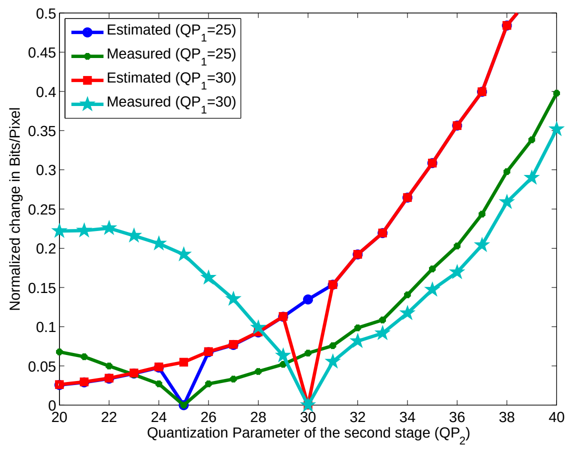


Figure 2.5: Estimated $\Delta R_{2,1}$(extra bits/pixels). Measured data represents measured values of $\Delta R_{2,1}$ when a two-stage encoder is used compare to the standard encoder, and Estimated data are calculated using (2.7). QP_1 is the quantization parameter for the first stage and QP_2 is the quantization parameter for the second stage.

$\Delta R_{2,1}$ represents extra bit/pixels required to keep the quality constant when a server uses two-stage encoder instead of the standard encoder. Figure 2.5 illustrates the measured $\Delta R_{2,1}$ and estimated $\Delta R_{2,1}$. To generate Figure 2.5, we used the estimated motion inaccuracy values from Figure 2.4 (using (A.18)), plugged them in (2.3) and (2.4) to find Φ_{ee_1} and Φ_{ee_1}. Then, we use (2.7) to generate Figure 2.5. These values do not represent estimation of $\Delta R_{2,1}$, when the residuals are not optimally encoded. Using non-optimal encoder for the residuals and error in (A.18) cause to have a mismatching between estimated and measured results in 2.5.

2.3 Performance Evaluation of the Two-stage Encoder

Both stages of our two-stage encoder have been implemented using the JM 12.4 reference software. The necessary modifications were made to store MVs on the hard disk and turn off the entropy coding module in the first stage. Then, we use these pre-computed MVs instead of motion estimation and turn on the entropy coding module in the second stage.

Comparison of RPS against other error control techniques is available elsewhere in literature, for example [70] and references therein. Hence, our main objective here is to compare the compression performance and complexity of our two-stage encoder versus the standard JM 12.4 reference encoder within the RPS framework. We perform this comparison on several standard test sequences - *Bus*, *Crew*, *Foreman*, and *Soccer* - all in YUV 4:2:0 format, QCIF resolution, at 15 fps.

In subsection 2.3.1, we validate the performance of the two-stage encoder and investigate the effects of using different QP values in the two stages. Next, in subsection 2.3.2 we test the performance of the two-stage encoder with RPS over a lossy network. Finally, in subsection 2.3.3 we illustrate the execution speed of the second stage of the two-stage encoder by comparing its execution time to the JM 12.4 encoder.

2.3.1 Characterization of the two-stage encoder

To characterize the two-stage encoder, we perform the following tests. For the first stage of the encoder, we turn off the rate control, and set the quantization parameter manually to QP $\in \{0,\ 25,\ 30,\ 35,\ 40\}$, where QP $= 0$ corresponds to using the original (unquantized) previous frames for motion estimation and compensation. Using these QP values, we estimate the MVs between each frame and its previous $N = 4$ frames. For each QP value, the output of this stage is the set of MVs, which we store on the hard disk.

In the second stage, we keep the rate control off, and choose various values of QP between 26 and 42 to cover a range of QP values that are typically used in encoding. For each of the QP values in the second stage, we read from the hard disk the MVs corresponding to one QP value from the first stage, and use those MVs for encoding. In this test, we are testing coding efficiency rather than lossy transmission performance, so we run our tests in an error-free environment. In Figure 2.6, we show the PSNR difference between the the video produced by the JM encoder (which uses the same QP value as the second stage of our encoder), and our two-stage encoder which, in the second stage, uses the QP value indicated on the horizontal axis in the graph. Note that when the QP values from the first stage and the second stage match, the two-stage encoder produces essentially the same video quality (indeed, the same bitstream) as the JM encoder. When there is a mismatch in the two QP values, the performance degrades. Hence, for VBR video (with fixed QP), the two-stage encoder can achieve the same performance as the JM encoder, as long as the same QP value is used in both stages. However, for CBR video, QP values change according to the rate control algorithm employed. In order to achieve the same performance as the JM encoder, we would need to generate and store MVs for all QP values that the JM encoder may use, which would require additional disk space.

Mismatch in the QP values in the two encoder stages may also increase the resulting bit rate. Figure 2.7 shows the resulting bit rate of the two-stage encoder, normalized to the bit rate produced by the JM 12.4 encoder. As in Figure 2.6, we see that the two-stage encoder produces the same bit-rate as the JM encoder as long as the QP values used in the two stages are the same. Overall, the results show that *the two-stage encoder can have the same compression performance as the standard H.264/AVC encoder* if the second encoding stage has access to the same MVs as the standard encoder. In the CBR case, this means that the first encoding stage needs to produce and store MVs for the QP values that are likely to be used by the standard encoder under the chosen rate control policy.

2.3.2 Performance over a lossy network

To test the performance of the two-stage encoder over a lossy network, we assume that each frame is sent in a single RTP packet. We use three commonly used loss models: independent, identically distributed (IID) loss pattern, two-state Markov (Gilbert) packet loss model with average burst length equal to 2.5, and measured packet traces from [60], which are commonly used in testing video transmission performance. The average packet loss rates (PLRs) were $0.03, 0.05, 0.10$, and 0.20. The encoder uses RPS-NACK (since it is more efficient than RPS-ACK) to decide on the reference frames to be used for motion-compensated prediction. Acknowledgements are assumed to be delivered correctly, but with a variable RTT generated as a normal random variable with a mean of 120 ms and standard deviation of 2 ms. In this test, both the JM encoder and the second stage of our two-stage encoder use rate control, with bit rate $\in \{100,\ 150,\ 200\}$ kbps. The encoded frame sizes were between 800 Bytes and 1600 Bytes. The second stage of the two-stage encoder uses MVs produced by using the same rate as in the first stage. We ran 300 simulation runs for the loss rate of 0.03, 200 simulation runs for the loss rate of 0.05, and 100 runs for the loss rates of 0.10 and 0.20.

The average PSNR results in dB are shown in Table 2.1 for the IID loss pattern, Table 2.2 for the Gilbert loss pattern with average burst length of 2.5, and Table 2.3 for measured packet traces from [60]. In Tables 2.1 - 2.3, Δ represents the PSNR of the two-stage encoder minus the PSNR of the JM decoder. The performance gap between the JM encoder and our two-stage encoder is no more than 0.6 dB in these lossy transmission scenarios with rate control. Perhaps surprisingly, at higher loss rates, the two-stage encoder sometimes provides a higher PSNR than the JM encoder, despite the fact that it has fewer MVs and coding modes to choose from in the second stage. This is because the the rate-distortion optimization in the JM encoder is optimal only for error-free transmission, as demonstrated in [68].

2.3.3 Encoding speed

H.264 is not only well known for its excellent compression performance, but also for its high complexity. The complexity of H.264 encoder poses significant challenges for real-time implementation. Therefore, complexity reduction of H.264 has been the focus of many research works. Since motion estimation is the most time consuming part of H.264 coding [39], several methods have been proposed to speed up motion estimation, such as fast motion search methods [69, 19], VLSI-based implementations [39], and implementations on SIMD[1]/MIMD[2] processors.

Fast motion search methods are commonly used for speeding up video encoding. Some of these methods use heuristic motion search instead of exhaustive search to find the best motion vector candidate [69, 19]. Another category of fast motion estimation methods takes the advantage of the motion correlation between adjacent macro-blocks [3] to reduce the motion search range. The proposed method in [69] has been included in JM12.4 and the results are demonstrated in Table 2.5 for comparison it with the proposed two-stage encoder. It should be noted that the required time for the first stage of the two-stage encoder depends on the used motion vector estimation algorithm, and the required time is very close to required time for the JM12.4 encoder in Table 2.5.

Several research works show the advantage of using VLSI implementations for motion estimation, such as parallel structures or high bandwidth pipelines between processor units and memory. But VLSI implementations are usually too expensive to become common solution. Using SIMD/MIMD processors is also an expensive solution, because SIMD/MIMD processors cost more and consume more power than SISD processors. The advantage of the proposed two-stage encoder is that it can achieve fast performance on a cheap general purpose processor, without relying on the more expensive SIMD/MIMD processors or VLSI solutions.

Although real-time H.264/AVC encoders such as x.264 are available today, we choose JM encoder for our tests because JM it is the “official” reference encoder for H.264/AVC, and our goal here is not to compare JM encoder against x.264 encoder, but rather to quantify the complexity reduction achieved by decoupling motion estimation from motion compensation,

[1]Single Instruction, Multiple Data [21]

[2]Multiple Instruction, Multiple Data [21]

which we have done within the “official” JM framework.

To test the encoding speed, we compare the time it takes to encode the *Foreman* sequence by the JM 12.4 encoder, and the second stage of our two-stage encoder. Both were compiled using the Microsoft Visual Studio 2005 C++ compiler, and tested on the Intel Core 2 machine with 2.13 GHz CPU and 2.00 GB of RAM. The JM encoder was tested in five configurations with various motion estimation algorithms listed in Table 2.4. These configurations range from the most complex one, involving full search motion estimation (JM-1), to the least complex one based on Enhanced Predictive Zonal Search (EPZS, JM-5). Encoding times are listed in Table 2.5 for $N \in \{2, 4, 8\}$ frames in the reference picture buffer. Encoding time for the two-stage encoder includes reading the pre-computed MVs from the hard disk. Note that neither encoder is optimized for the particular processor used in this test; in fact, both are based on the general-purpose JM 12.4 software, which is rather slow, so the encoding times listed in the table are somewhat pessimistic. As shown, the second stage of our two-stage encoder achieves approximately a 9-fold speed-up compared to the JM encoder for $N = 4$ with full search motion estimation, and even higher speed-up for larger N.

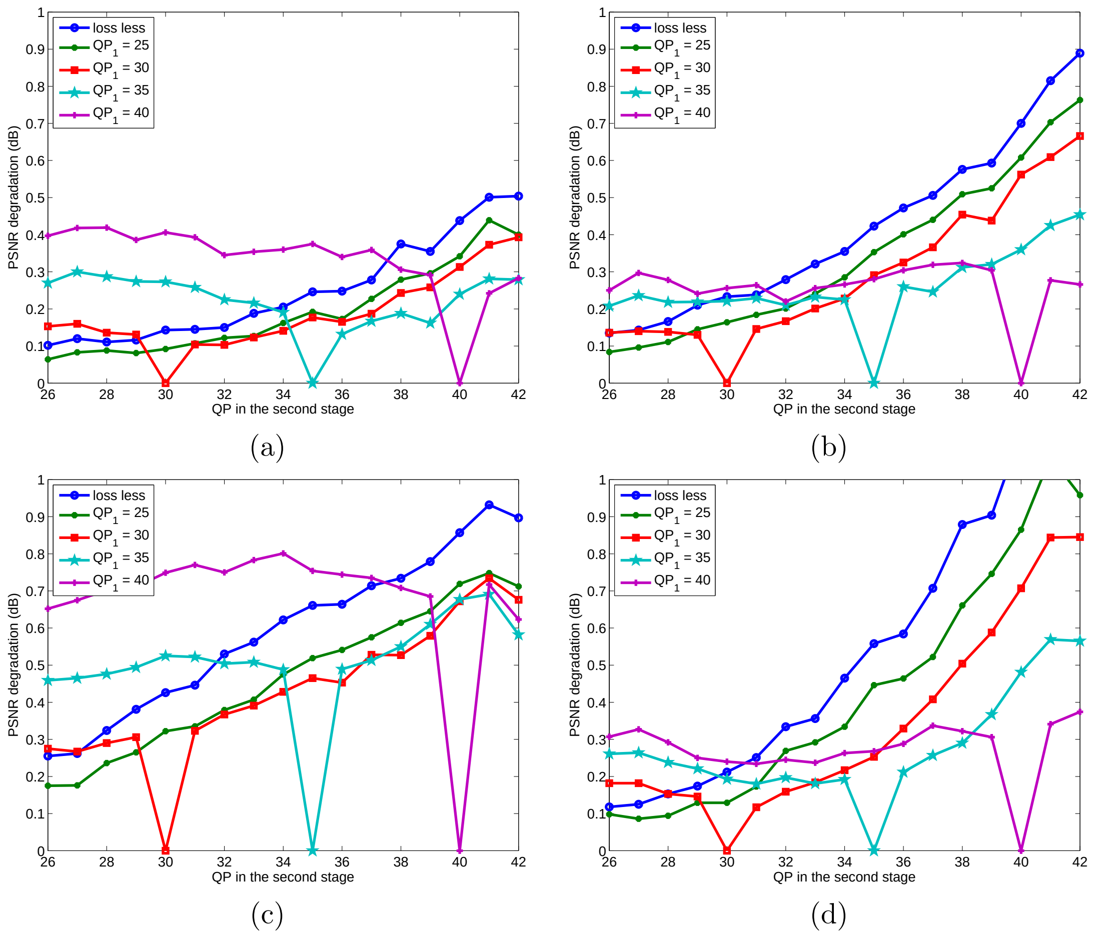
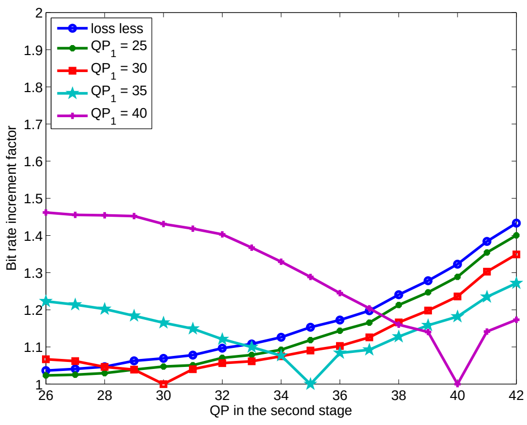
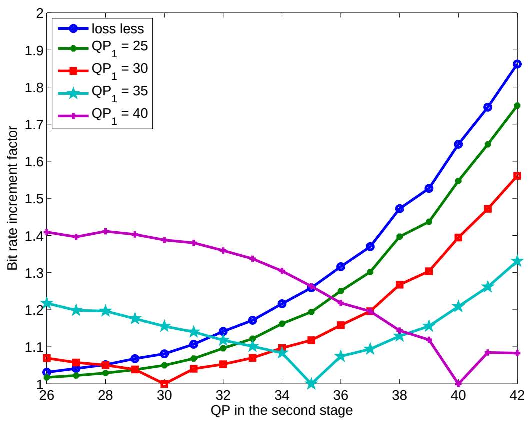


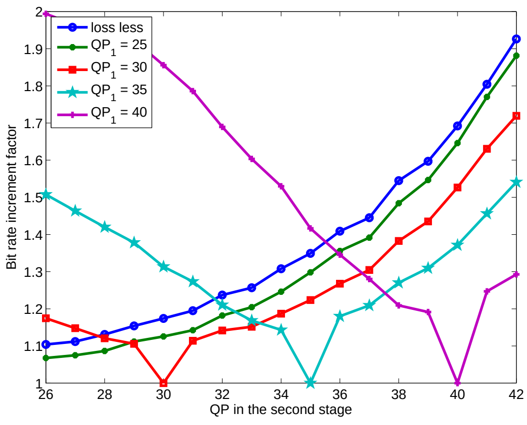
Figure 2.6: Quality degradation when QP in the first stage (QP1) is different from the QP in the second stage. (a) *Bus*, (b) *Crew*, (c) *Foreman*, and (d) *Soccer*.



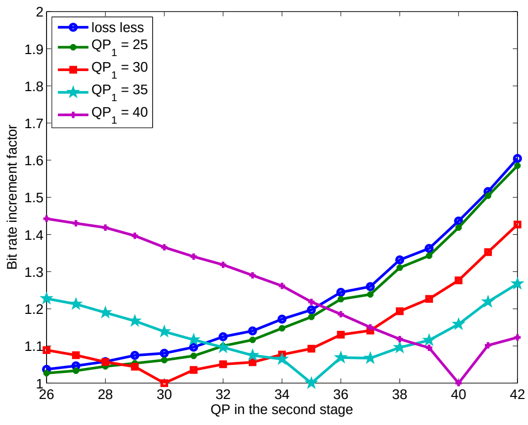
(a)



(b)



(c)



(d)

Figure 2.7: Bit-rate increase when QP in the first stage (QP1) is different from the QP in the second stage. (a) *Bus*, (b) *Crew*, (c) *Foreman*, and (d)*Soccer*.

Sequence	**PLR**	**100kbs**			**150kbs**			**200kbs**		
		JM	**2-ST**	Δ	**JM**	**2-ST**	Δ	**JM**	**2-ST**	Δ
	0.03	34.77	34.36	−0.41	36.83	36.51	−0.32	37.96	37.75	−0.22
Foreman	0.05	33.81	33.50	−0.32	35.63	35.38	−0.25	37.21	36.99	−0.22
	0.10	32.70	32.52	−0.17	34.46	34.30	−0.16	35.68	35.53	−0.15
	0.20	28.84	28.95	+0.11	29.77	29.74	−0.03	30.55	30.60	−0.05
	0.03	32.07	31.73	−0.34	33.87	33.70	−0.17	35.24	35.11	−0.13
Crew	0.05	31.58	31.26	−0.32	33.31	33.12	−0.19	34.47	34.35	−0.11
	0.10	30.84	30.60	−0.25	32.48	32.37	−0.12	33.53	33.40	−0.13
	0.20	28.31	28.32	−0.02	29.49	29.59	+0.11	30.34	30.22	−0.12
	0.03	27.72	27.44	−0.29	29.83	29.61	−0.22	31.51	31.33	−0.18
Bus	0.05	27.00	26.74	−0.26	29.02	28.84	−0.18	30.44	30.32	−0.12
	0.10	26.39	26.16	−0.23	28.14	28.06	−0.08	29.50	29.49	−0.01
	0.20	20.74	20.73	−0.01	19.42	19.47	+0.05	19.73	19.85	+0.12
	0.03	33.43	33.12	−0.31	35.57	35.36	−0.21	37.24	37.03	−0.20
Soccer	0.05	33.27	32.09	−0.36	35.41	35.16	−0.25	37.08	36.85	−0.24
	0.10	32.88	32.44	−0.44	35.05	34.74	−0.31	36.72	36.44	−0.29
	0.20	32.41	31.87	−0.55	34.64	34.26	−0.38	36.32	35.98	−0.34

Table 2.1: PSNR in dB of the JM 12.4 encoder (JM), our two-stage encoder (2-ST), and their difference (Δ) when the losses happen independently, with rate control turned on. Encoded bit rate is shown in the top row of the table.

Sequence	**PLR**	**100kbs**			**150kbs**			**200kbs**		
		JM	**2-ST**	Δ	**JM**	**2-ST**	Δ	**JM**	**2-ST**	Δ
	0.03	34.57	34.37	−0.20	35.88	35.72	−0.16	37.35	37.18	−0.17
Foreman	0.05	33.57	33.70	+0.13	34.44	34.51	+0.07	35.89	36.01	+0.12
	0.10	29.66	29.72	+0.06	31.78	32.19	+0.41	33.37	33.65	+0.28
	0.20	28.95	29.16	+0.21	30.32	30.86	+0.54	31.61	32.03	+0.42
	0.03	32.20	31.90	−0.30	33.93	33.80	−0.13	35.29	35.16	−0.13
Crew	0.05	31.43	31.17	−0.26	32.96	32.77	−0.19	34.32	34.25	−0.07
	0.10	29.33	29.15	−0.18	30.23	30.09	−0.13	31.45	31.42	−0.03
	0.20	28.57	28.47	−0.11	29.51	29.40	−0.11	30.46	30.44	−0.01
	0.03	28.00	27.73	−0.28	30.06	29.92	−0.14	31.73	31.54	−0.19
Bus	0.05	26.99	26.80	−0.19	28.96	28.87	−0.09	30.43	30.31	−0.11
	0.10	24.50	24.34	−0.16	26.49	26.60	+0.11	28.11	28.00	−0.11
	0.20	21.06	21.25	+0.19	21.96	22.28	+0.33	21.34	21.87	+0.53
	0.03	33.52	33.30	−0.22	35.65	35.52	−0.13	37.33	37.19	−0.14
Soccer	0.05	33.41	33.13	−0.29	35.55	35.37	−0.18	37.23	37.05	−0.18
	0.10	33.12	32.72	−0.40	35.29	35.02	−0.27	36.97	36.73	−0.24
	0.20	32.72	32.19	−0.53	34.92	34.56	−0.36	36.61	36.29	−0.33

Table 2.2: PSNR in dB of the JM 12.4 encoder (JM), our two-stage encoder (2-ST), and their difference (Δ) over a Gilbert channel with $L_B = 2.5$, and with rate control turned on. Encoded bit rate is shown in the top row of the table.

Sequence	**PLR**	**100kbs**			**150kbs**			**200kbs**		
		JM	**2-ST**	Δ	**JM**	**2-ST**	Δ	**JM**	**2-ST**	Δ
	0.03	35.00	34.66	−0.34	36.95	36.71	−0.24	38.44	38.20	−0.24
Foreman	0.05	34.46	34.09	−0.37	36.37	36.04	−0.33	37.76	37.59	−0.17
	0.10	28.76	29.04	+0.29	30.10	30.35	+0.25	30.79	31.12	+0.33
	0.20	28.77	28.61	−0.16	30.02	29.89	−0.13	30.88	30.74	−0.14
	0.03	32.33	32.09	−0.24	34.15	33.98	−0.18	35.44	35.33	−0.11
Crew	0.05	31.89	31.62	−0.27	33.59	33.51	−0.08	34.97	34.85	−0.13
	0.10	28.77	28.61	−0.16	30.02	29.89	−0.13	30.88	30.74	−0.14
	0.20	28.21	28.12	−0.09	29.33	29.29	−0.04	30.15	30.12	−0.02
	0.03	28.11	27.83	−0.28	30.19	30.01	−0.19	31.77	31.64	−0.14
Bus	0.05	27.58	27.29	−0.29	29.63	29.47	−0.16	31.16	31.00	−0.16
	0.10	23.94	23.99	+0.05	25.33	25.52	+0.18	26.34	26.40	+0.05
	0.20	28.06	28.36	+0.30	29.31	29.50	+0.20	30.06	30.30	+0.24
	0.03	33.39	33.08	−0.32	35.53	35.31	−0.21	37.20	36.99	−0.21
Soccer	0.05	33.47	33.20	−0.27	35.60	35.42	−0.18	37.28	37.10	−0.18
	0.10	32.83	32.52	−0.31	35.01	34.68	−0.33	36.68	36.38	−0.30
	0.20	32.32	32.07	−0.25	34.56	34.27	−0.28	36.22	35.98	−0.24

Table 2.3: PSNR in dB of the JM 12.4 encoder (JM), our two-stage encoder (2-ST), and their difference (Δ) over measured packet traces from [60], with rate control turned on. Encoded bit rate is shown in the top row of the table.

JM configuration	Motion search
JM-1	Full Search
JM-2	Fast Full Search
JM-3	UMHex
JM-4	Simplified Hexagon Search
JM-5	EPZS

Table 2.4: Motion search methods in five configurations of the JM encoder.

Encoder	$N = 2$	$N = 4$	$N = 8$
JM-1	173.86	336.55	688.51
JM-2	192.22	355.83	691.61
JM-3	68.37	106.78	188.56
JM-4	57.80	88.38	156.22
JM-5	63.24	92.13	150.75
2-stage (2^{nd} stage)	33.54	37.10	42.17

Table 2.5: Average encoding time per frame (in milliseconds) for various JM configurations (JM-1 through JM-5), and our two-stage encoder (two-stage). At at the first stage, the encoder uses JM-1 for motion estimation

Chapter 3

Noncausal Whole-frame Concealment

In contrast to all other error control techniques, error concealment algorithms have a unique advantage, which is that their use does not cause degradation of coding efficiency. Many of them contemplate on the available information at the decoder. The decoder can use the remaining redundancy in the received bit-stream. The redundancy may exist in temporal domain [34], spatial domain [46], or spatiotemporal domain [17]. Temporal concealment algorithms usually reestimate motion vectors (MVs) of the corrupted macroblocks (MBs) to be used in motion compensation. In contrast, spatial concealment techniques usually use the spatial smoothness property of the video signal and attempt to use the information form neighboring MBs to interpolate the corrupted pixels. Sharp edges and complex textures disrupt the smoothness of the video signal. In such cases, an edge preserving smoothness criterion is needed to interpolate the corrupted pixels [26]. In this chapter, we describes the Noncausal Whole-frame Concealment technique that we proposed in [6]

3.1 Whole-frame Concealment

In low bit rate coding of low resolution (e.g., QCIF) video, each encoded frame typically fits within a single network packet, therefore the loss of a packet during transmission of compressed bit-stream may cause the loss of a whole video frame [8]. Even in high bit rate scenarios, traffic congestion causes burst errors, which can again lead to whole-frame loss.

When a whole-frame loss happens, the error concealment algorithms mentioned above are of limited use, because most of them use information from neighboring MBs to recover the motion vector information in temporal concealment, or to interpolate the corrupted pixels in spatial concealment. So, to deal with a whole-frame loss, the concealment algorithm needs to use a different strategy. In [14], Belfiore et al. proposed a novel whole-frame concealment method, which is based on multi-frame optical flow estimation. This algorithm estimates the motion of the lost frame by extending the motion vectors of the last decoded frame and projecting that frame onto the lost frame. In [14], it was reported that this algorithm outperforms the simple "frame copy" method in the JM H.264 decoder by several dBs in PSNR. However, it is too complex to run in real time, because it uses several median filtering operations in the pixel domain. To solve the high complexity problem of the method from [14], Baccichet et al. in [8] proposed a block-based whole-frame concealment which can work in real time at the expense of losing some quality.

3.1.1 The Method Proposed in [14]

The algorithm proposed in [14] is based on the optical flow theory. Belfiore et al. assumed that motion of the objects in one frame does not change seriously in the consecutive frames. Therefore, the algorithm can project these objects to the lost frame and use the projected motion vectors for concealment.

To have a more detailed explanation for the algorithm, assume that frame t has been lost and frame $t-1$ is correctly decoded. The proposed algorithm will accomplish the following steps to conceal frame t

1. The first step is estimation of the optical flow. Instead of working with blocks and macroblocks, this algorithm works with pixels. In this stage, a motion vector for each pixel in frame $t-1$ is computed in order to construct a forward motion vector field. Then these motion vectors are rescaled according to their temporal distance, and their accuracy is reduced from quarter pixel to half pixel.

2. It is possible to have some holes (areas without assigned motion vector) in the forward motion vector field of the previous step, because some pixels in frame $t-1$ might be encoded in intra prediction. To fill these points the algorithm applies a 7x7 median filter. Then, the algorithm applies a 15x15 median filter to make the forward motion vector field smoother.

3. After construction of forward motion vector field, every pixel in the frame $t-1$ is moved to temporary buffer for frame t. Since motion vectors have half-pixel accuracy, the size of the temporary buffer is twice of the original buffer, and every 2x2 block of the pixels in the buffer corresponds to a pixel in the final recovered frame. For every pixel in the temporary buffer, it is possible to be pointed by several point in the frame $t-1$. The algorithm proposed in [14] simply calculates the average of these points and replaces the average value.

4. There is a possibility to still have some empty point in the temporary buffer. The algorithm recursively fill them using a 9x9 median filter.

5. At the final step, the resulted frame in the temporary buffer is downsampled by factor 2 and to prevent aliasing the frame is smoothed by 2x2 averaging filter.

3.2 Noncausal Whole-frame Concealment

Most existing concealment algorithms use causal processing. To the best of our knowledge, there has been only a few works that take advantage of the receiver-side buffer to enhance the concealment process by noncausal processing [12, 13]. The proposed algorithm works sequentially in time-forward direction, but it uses noncausal information during error concealment. To explain the algorithm more precisely, we divide it into two steps. The first step uses the Optical Flow principle [58], and assumes that every object in the video has a constant velocity (similar to what is assumed in [14]) in order to synthesize a preliminary version of the concealed frame. Subsequently, the second step uses a boundary-matching algorithm to re-estimate the motion vectors of the remaining empty areas. The following two subsections describe the two steps of the proposed algorithm.

3.2.1 Noncausal optical flow

The first step of the algorithm uses the optical flow to estimate the motion vectors of the lost frame. If the frame at time t (called frame t for short) has been lost, the algorithm has three different sources of information to estimate its motion vectors.

Causal source:

Frame $t-1$ can act as a causal source of motion vector information. With the assumption of constant velocity for each pixel in frame $t-1$, we can find the location of each pixel in the current frame t by extending and scaling its motion vector from frame $t-1$ to frame t, except for the intra-coded pixels in frame $t-1$. MV recovery from the causal source is illustrated in Figure 3.1 and summarized in Algorithm 4. In Figure 3.1, we assume that frame t is lost while frame $t-1$ is received. The illustrated MV in frame $t-1$ uses frame $t-3$ as the reference. Since the temporal distance between frame $t-3$ and frame $t-1$ is twice the temporal distance between frame $t-1$ and frame t, the MV is extended forward

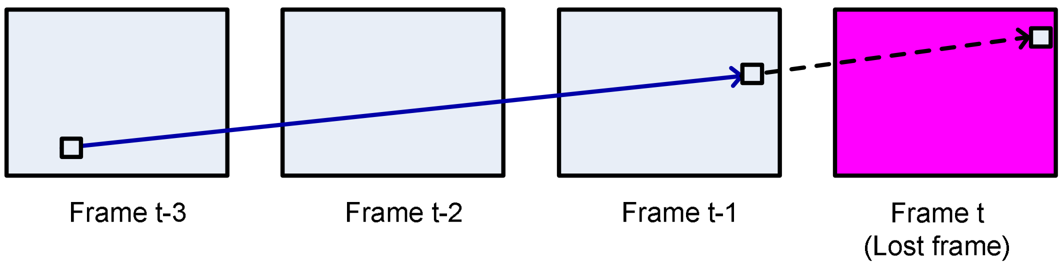


Figure 3.1: Causal MV recovery: original MV from frame $t-3$ to frame $t-1$ shown by solid line, recovered MV from frame $t-1$ to frame t shown by dashed line.

and scaled by 1/2.

Algorithm 4 Causal MV recovery

```
if frame $t$ is lost then
  Initialize $\sharp candidate_{causal}(p_t, t) \leftarrow 0$
  $t_n \leftarrow t - 1$
  for all pixels $p_{t_n}$ in frame $t_n$ do
    if $p_{t_n}$ belongs to a P-coded block then
      $scale$ = Temporal distance between $p_{t_n}$ and its reference
      $(d_x, d_y) = \frac{1}{scale}\left(MV_x\left(x, y, t_n\right), MV_y\left(x, y, t_n\right)\right)$
      $f\left(x - d_x, y - d_y, t\right) = f\left(x, y, t-1\right)$
      $\sharp candidate_{causal}(p_t, t)$ ++
    end if
  end for
  for all pixels $p_t$ in frame $t$ do
    if $\sharp candidate_{causal}(p_t, t) == 1$ then
      SET ($MV_{causal}(p_t, t)$ is VALID)
    end if
  end for
end if
```

Anti-causal source:

The first correctly received frame after the lost frame, once decoded, may suffer from error propagation if the lost frame is used as its reference in the motion compensation loop. However, its MVs can be used as an anti-causal source for MV recovery. Our algorithm extends

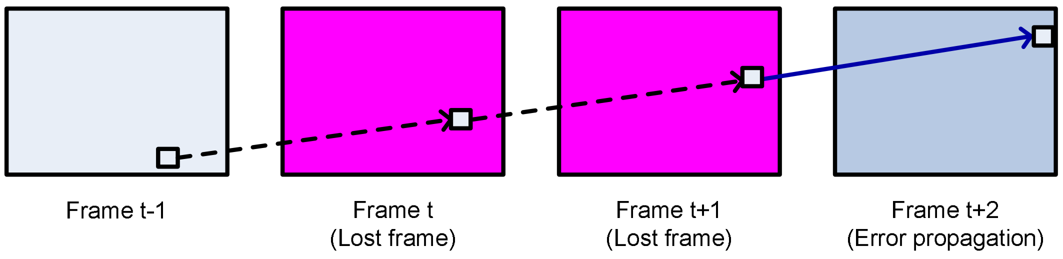


Figure 3.2: Anti-causal MV recovery: original MV from frame $t+1$ to frame $t+2$ shown by solid line, recovered MVs for the two lost frames, t and $t+1$, shown by dashed line.

the MVs of the next correctly received frame backwards, with appropriate scaling if necessary. MV recovery from the anti-causal source is illustrated in Figure 3.2 and summarized in Algorithm 5. In Figure 3.2, we assume that frames t and $t+1$ are lost, while frame $t+2$ is correctly received. The illustrated MV in frame $t+2$ uses frame $t+1$ as the reference. In this case, our algorithm decides to extend this MV in the time-reversed direction. The recovered MV for frame t will be the portion of the extended MV that lies between frames $t-1$ and t, while the recovered MV for frame $t+1$ will be the portion of the extended MV that lies between frames t and $t+1$.

Noncausal source:

If the video communication system uses RPS-NACK (Figure 1.5), the proposed algorithm can use the MVs of the reset frame (noncausal source of MV information) to recover the MVs of the lost frame. Noncausal MV recovery is illustrated in Figure 3.3, where we assume that frame t is lost, frame $t+1$ is correctly received (but may suffer from error propagation once decoded), and frame $t+2$ is the RPS-NACK reset frame. The illustrated MV in the reset frame uses frame $t-1$ as the reference, which stops error propagation. Our algorithm uses the portion of this MV that lies between frames $t-1$ and t as the recovered MV for frame t. The procedure is summarized in Algorithm 6.

Algorithm 5 Anti-causal MV recovery

```
if frame $t$ is lost then
  Initialize $\sharp candidate_{anti-causal}(p_t, t) \leftarrow 0$
  $t_n \leftarrow$ Find_Next_Correctly_Received_Frame($t$)
  for all pixels $p_{t_n}$ in frame $t_n$ do
    if $p_{t_n}$ belongs to a P-coded block then
      $scale$ = Temporal distance between $p_{t_n}$ and its reference
      $(d_x, d_y) = \frac{t_n - t}{scale}\left(MV_x\left(x, y, t_n\right), MV_y\left(x, y, t_n\right)\right)$
      $f\left(x - d_x, y - d_y, t\right) =$
        $f\left(x - \frac{dx}{t_n - t}, y - \frac{dy}{t_n - t}, t - 1\right)$
      $\sharp candidate_{anti-causal}(p_t, t)$ ++
    end if
  end for
  for all pixels $p_t$ in frame $t$ do
    if $\sharp candidate_{anti-causal}(p_t, t) == 1$ then
      SET ($MV_{anti-causal}(p_t, t)$ is VALID)
    end if
  end for
end if
```

3.2.2 Using Optical Flow in the Proposed Error Concealment Algorithm

After collecting all candidate MVs from the three sources (causal, anti-causal, or noncausal), for each pixel in the lost frame there are three possible outcomes: having no MV candidates, having only one MV candidate, or having more than one MV candidate. Occlusions and intra-coding may cause a pixel to have no MV candidates, or multiple MV candidates from the same source (for example, multiple causal MV candidates). For the pixels with a single MV candidate, the causal source usually gives the best performance due to its temporal proximity to the lost frame, while the noncausal source is the least accurate due to the fact that its MV is stretched over a large temporal distance. Therefore, we first fill in all pixels with only one causal MV candidate. Among the remaining pixels, we then fill in all those pixels with only one anti-causal MV candidate. Finally, among the remainning pixels, we fill in all those pixels with only one noncausal MV candidate.

After performing the tasks outlined above, we remove the outlier MVs from the MV

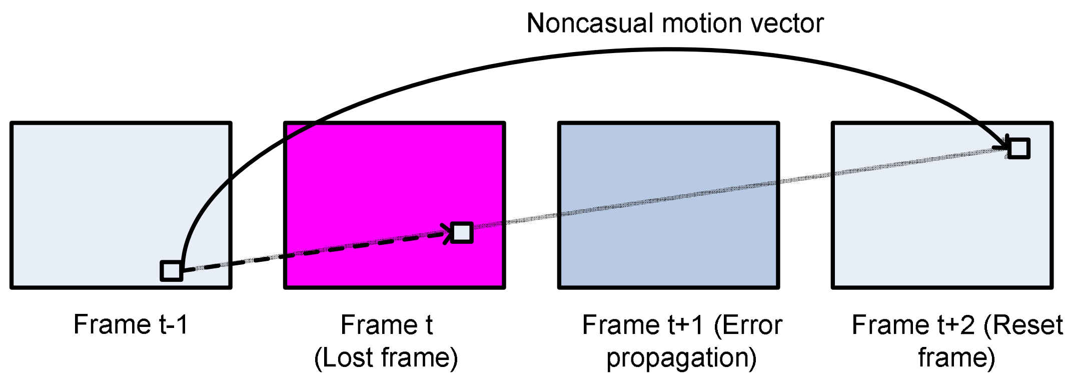


Figure 3.3: Noncausal MV recovery: original MV from frame $t-1$ to frame $t+2$ shown by solid line, recovered MV from frame $t-1$ to frame t shown by dashed line.

field in the following way. Each pixel that was assigned a MV in one of the previous steps is labeled as VALID, while the remaining pixels are labeled INVALID. We put a 5×5 mask around each pixel and count all its VALID neighbors in the mask. If a pixel has less than 5 VALID neighbors in a 5×5 neighborhood, its MV is suspected of being noisy and it is removed, while the pixel is labeled INVALID. At this point, the preliminary version of the lost frame is created. INVALID pixels constitute empty areas in this preliminary frame.

3.2.3 Spatiotemporal boundary matching

MV recovery by boundary matching [17] has proven to be a powerful spatiotemporal technique in video error concealment. Boundary matching uses two important properties of video signals: temporal correlation and spatial correlation. Strong correlation usually exists between temporally neighboring frames. Therefore, the empty areas in the preliminary version of the lost frame can be replaced by areas of the same shape from a previous frame, or from a correctly decoded future frame. For the i-th empty area, we extract its boundary B_i in order to find the best match for it in one of the neighboring frames. We use the sum of absolute differences (SAD) of boundary pixels as the distortion function, and find the MV corresponding to the minimum SAD, as in [17]. The procedure is illustrated in Fig 3.4, and

Algorithm 6 Noncausal MV recovery

```
if frame $t$ is lost then
  Initialize $\sharp candidate_{noncausal}(p_t, t) \leftarrow 0$
  $t_n \leftarrow$ Find_RPS-NACK_Reset_Frame($t$)
  if $t_n$ is available then
    for all pixels $p_{t_n}$ in frame $t_n$ do
      if $p_{t_n}$ belongs to a P-coded block then
        $scale$ = Temporal distance between $p_{t_n}$ and its reference
        $(d_x, d_y) = \frac{t_n - t}{scale}\left(MV_x(x, y, t_n), MV_y(x, y, t_n)\right)$
        $f(x - d_x, y - d_y, t) =$
          $f\left(x - \frac{dx}{t_n - t}, y - \frac{dy}{t_n - t}, t - 1\right)$
        $\sharp candidate_{noncausal}(p_t, t)$ ++
      end if
    end for
    for all pixels $p_t$ in frame $t$ do
      if $\sharp candidate_{noncausal}(p_t, t) == 1$ then
        SET ($MV_{noncausal}(p_t, t)$ is VALID)
      end if
    end for
  end if
end if
```

summarized in equations (3.1) and (3.2). In words, among the K previous frames and the RPS-NACK reset frame (if available), we choose the frame t_0 and the corresponding MV that yield the lowest SAD for boundary B_i. In (3.1), D is the Round Trip Time (RTT) in frames, so $t + D$ is the index of the RPS-NACK reset frame.

$$\left(t_0, \vec{MV}\right) = \underbrace{argmin}_{t_0 = t-1, \cdots, t-K, t+D} SAD\left(t_0, \vec{MV}\right), \tag{3.1}$$

$$SAD\left(t_0, \vec{MV}\right) = \sum_{\vec{n} \in B_i} \left| f(\vec{n}, t) - f\left(\vec{n} - \vec{MV}, t_0\right) \right|. \tag{3.2}$$

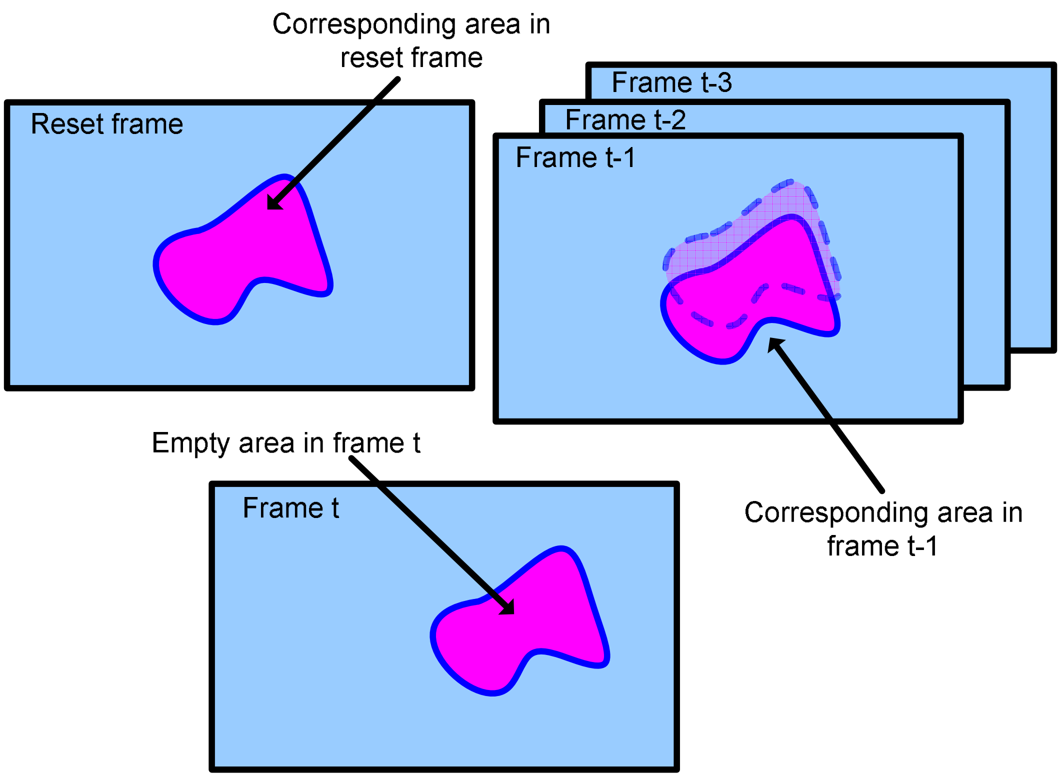


Figure 3.4: Boundary matching.

3.3 Performance Evaluation of the Noncausal Whole-Frame concealment

The proposed noncausal whole-frame concealment algorithm has been implemented in the JM12.4 H.264/AVC decoder, which has the ability to interact with the encoder in a RPS-NACK framework. To characterize the performance of the proposed algorithm, we use four standard video sequences, namely *Soccer*, *Foreman*, *Crew*, and *Bus*. All the sequences are QCIF resolution at 15 fps. They were encoded using full search motion estimation with 16×16 search window and four reference frames, at 100, 150, and 200kbps.

To gather the experimental results in this section, we do not assume any specific traffic or packet loss model. We simply drop each frame in turn from the bit-stream, and conceal it using the state-of-the-art method from [14] and our proposed method. We report the average Peak Signal-to-Noise Ratio (PSNR) in dB of all affected frames for each frame loss.

The performance of error concealment depends on RTT for several reasons. First, when RTT increases, it takes longer for the encoder to become aware of the loss through a NACK message, so the number of affected frames between the correctly received frame and the RPS-NACK reset frame increases. Second, in the case of our proposed method, the accuracy of the noncausal MV candidate is degraded when RTT increases, because the assumption of constant velocity over large temporal distances becomes less valid.

We tested the performance of the proposed algorithm with two RTTs - 60 ms and 120 ms. The average PSNR performance of our proposed method and the one from [14] is reported in Table 3.1 for RTT = 60 ms and Table 3.2 for RTT = 120 ms. Data in Table 3.1 and 3.2 represent the average PSNR of concealed frames, and those frames that use the concealed frames as a reference, compared to the original frames in dB. We also report the gain of our method compared to the method from [14]. The results show PSNR gains of up to about 1.2 dB.

Figures 3.5 and 3.6 show visual comparison of the concealed frames. In Figure 3.5, frame 144 of *Soccer* is lost. The proposed method offers a better recovery of the lost frame compared to the method from [14], especially in the high-motion areas in the foreground where the soccer player is located. This example shows that the proposed algorithm provides better visual quality.

In Table 3.3, we show the average required time to conceal one frame using the two concealment methods. The results in Table 3.3 have been obtained by running the modified JM decoder, which has been compiled using the Microsoft Visual Studio 2005 C++ compiler, and have been obtained on Intel Core 2 machine with 2.13 GHz CPU and 2 GB of RAM. These data show that the proposed algorithm runs much faster than the method from [14].

To have a real time decoder, when the frame rate is 15 fps, the decoder has about 66.7

ms on the average to decode a received frame or to conceal a lost frame. Based on the measurements we did on the machine described above, decoding a QCIF frame by the JM decoder needs 14.69 ms. Therefore, the remaining time may be used for concealment of lost frames. Let p be the packet loss probability, and assume the decoder has a long enough buffer to smooth out its playback. Then the decoder can run in real time in the steady-state only if inequality (3.3) is satisfied [42]. In inequality (3.3), p is probability of packet (frame) loss, t_c is the average required time to conceal a frame, t_d is the average time required to decode a frame, and F is the frame rate.

$$pt_c + (1-p)t_d \leq \frac{1}{F}. \tag{3.3}$$

Based on(3.3) and the data from Table 3.3, we can find the maximum value of p which allows the decoder to work in real time in the steady state. The value of p is limited above by 0.1684 when the decoder uses our proposed method to conceal the lost frames. In contrast, the maximum value for p when the decoder uses the method from [14] is 0.0012, which is much more restrictive, and smaller than the common packet loss probabilities found in real communication systems.

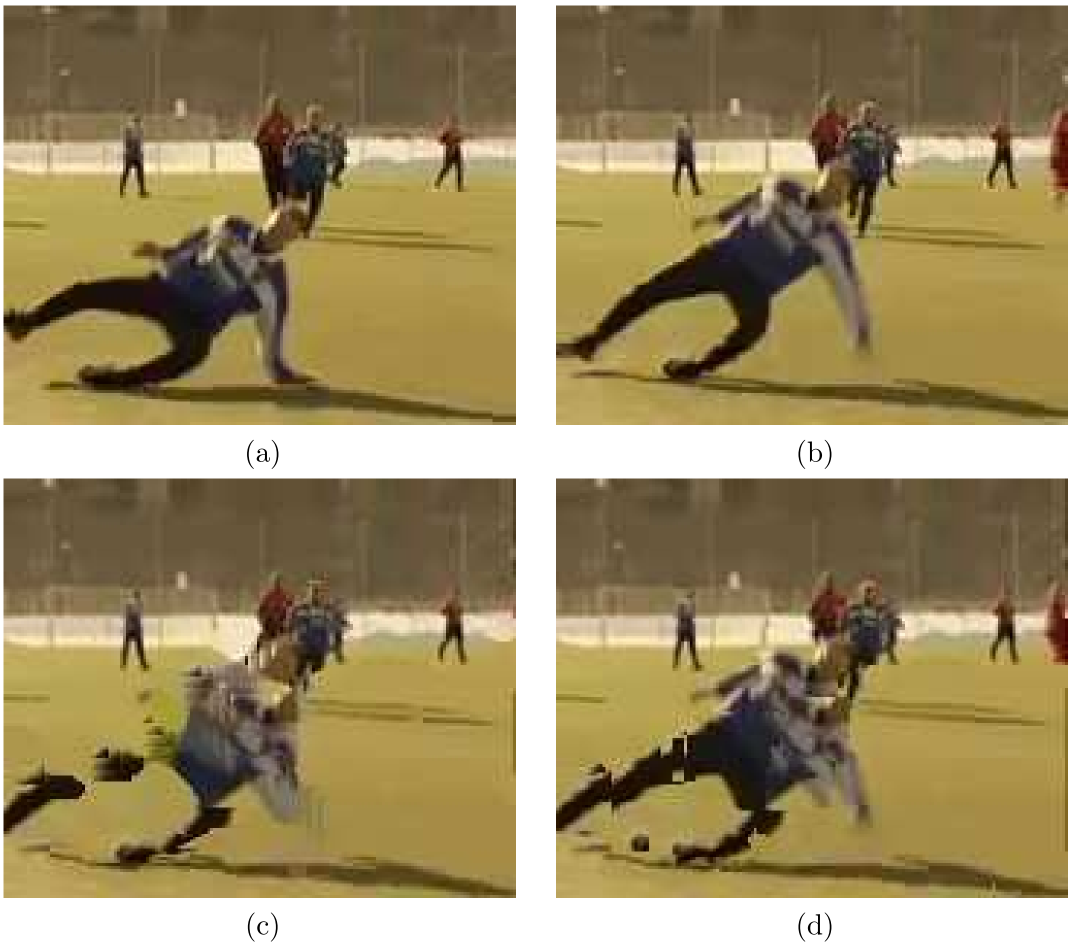


Figure 3.5: Frame 144 of *Soccer*. (a) Loss-free decoded frame, (b) Reconstructed by frame copy (frame 144 is lost) (c) Reconstructed by the method from [14] (frame 144 is lost), and (d) Reconstructed by the proposed method (frame 144 is lost).

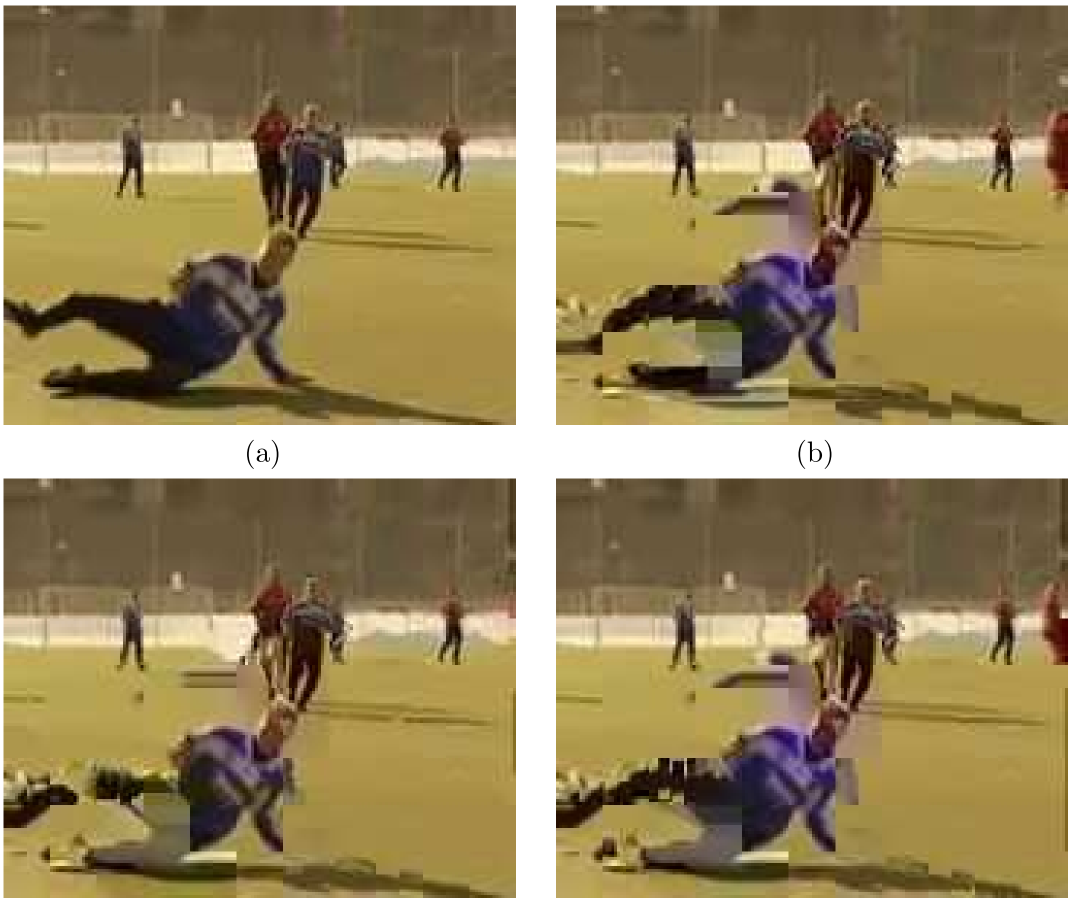
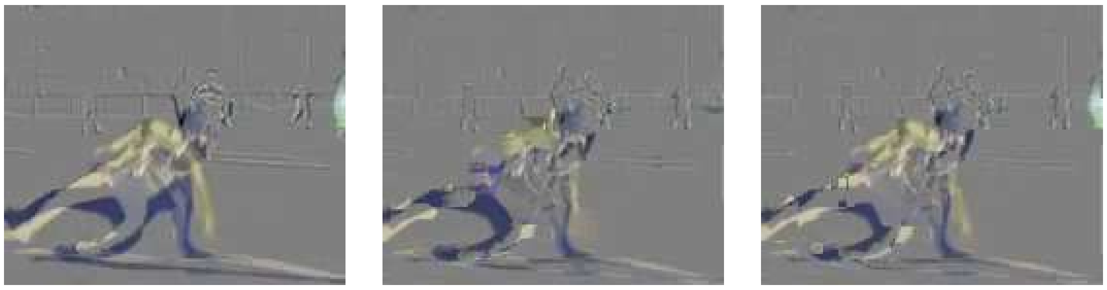
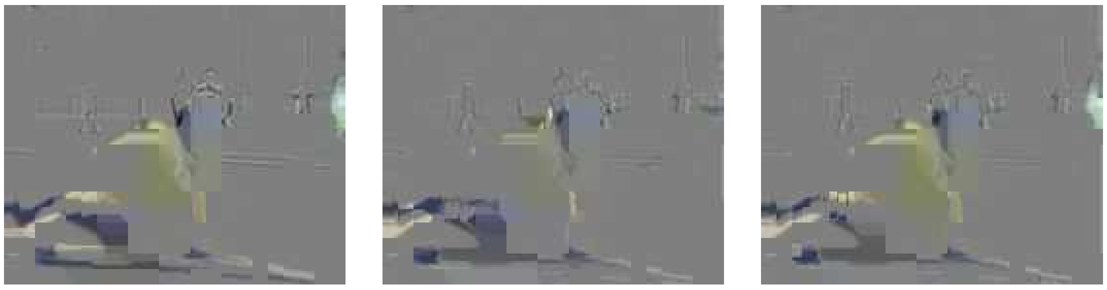

(a) (b) (c) (d)

Figure 3.6: Frame 145 of *Soccer*. (a) Error free decoded frame, (b) Reconstructed by frame copy (frame 144 is lost), (c) Reconstructed by the method from [14] (frame 144 is lost), and (d) Reconstructed by the proposed method (frame 144 is lost).



(a) (b) (c)

Figure 3.7: The difference between original version of Frame 144 and its erroneous version. (a) Reconstructed by frame copy (frame 144 is lost), (b) Reconstructed by the method from [14] (frame 144 is lost), and (c) Reconstructed by the proposed method (frame 144 is lost).



(a) (b) (c)

Figure 3.8: The difference between original version of Frame 145 and its erroneous version. (a) Reconstructed by frame copy (frame 144 is lost), (b) Reconstructed by the method from [14] (frame 144 is lost), and (c) Reconstructed by the proposed method (frame 144 is lost).

Sequence	**Bit rate**	**Method from [14]**	**Proposed**	**Gain**
Soccer	100 kbs	24.66	25.25	+0.59
	150 kbs	25.41	26.06	+0.64
	200 kbs	26.03	26.59	+0.55
Foreman	100 kbs	28.79	28.88	+0.08
	150 kbs	29.66	29.78	+0.11
	200 kbs	30.31	30.41	+0.10
Crew	100 kbs	25.77	26.19	+0.32
	150 kbs	25.88	26.43	+0.36
	200 kbs	25.89	26.54	+0.45
Bus	100 kbs	23.08	24.10	+1.02
	150 kbs	23.51	24.57	+1.06
	200 kbs	23.72	24.89	+1.17

Table 3.1: PSNR performance in dB of different concealment algorithms when RTT = 60 ms. The gain of the proposed method over the one in [14] is also shown.

Sequence	**Bit rate**	**Method from [14]**	**Proposed**	**Gain**
Soccer	100 kbs	21.68	22.12	+0.45
	150 kbs	21.87	22.27	+0.40
	200 kbs	21.80	22.29	+0.48
Foreman	100 kbs	26.28	26.40	+0.12
	150 kbs	26.50	26.67	+0.17
	200 kbs	26.62	26.76	+0.14
Crew	100 kbs	26.13	26.57	+0.29
	150 kbs	26.42	26.86	+0.32
	200 kbs	26.52	27.01	+0.39
Bus	100 kbs	23.21	24.19	+0.98
	150 kbs	23.66	24.70	+1.05
	200 kbs	23.88	25.06	+1.18

Table 3.2: PSNR performance in dB of different concealment algorithms when RTT = 120 ms. The gain of the proposed method over the one in [14] is also shown.

Concealment method	Time (t_c)
Method from [14]	43988 ms
Proposed	322 ms

Table 3.3: Average time required to conceal one frame, rounded up to the nearest millisecond.

Chapter 4

End to End System Performance Evaluation

In chapter 2 and chapter 3 we proposed, respectively, a new structure for H.264 encoder which has fast RPS and a noncausal error concealment algorithm for H.264 decoder. The performance evaluation of two-stage encoder shows that it is fast enough for run in real time, but the video quality is lower then the JM encoder. On the other hand, the proposed error concealment works better in recovering the damaged frames and provides better qualities than the competing method from [14].

In a complete video communication system, it necessary to enhance robustness of the bitstream in the encoder using error resilience techniques, and combat the remaining errors in the decoder using an error concealment algorithm. Therefore, in this chapter, we evaluate joint performance of both proposed algorithms to examine their efficiency for real video communication systems.

4.1 Experimental results

We use four video communication systems in our simulations.

- **A**: **A** is the reference system and contains a JM encoder which is modified to run reference picture selection (RPS), and a JM decoder which uses the Belfiore's algorithm in the error concealment module [14].
- **B**: **B** contains a JM encoder which is modified to run reference picture selection (RPS), which uses the proposed noncausal error concealment algorithm.
- **C**: **C** contains the proposed two-stage encoder, which has fast RPS, and a JM decoder which uses the Belfiore's algorithm in the error concealment module [14].
- **D**: **D** contains the proposed two-stage encoder, which has fast RPS, and a JM decoder which uses the proposed noncausal error concealment algorithm.

To evaluate these systems, we simulated their performance using the following conditions:

- We use four different video sequences (*Foreman, Crew, Bus, and Soccer*) with different properties to cover different situations that a system may encounter.
- We use a range of practical packet loss rages ($PLR \in \{3\%, 5\%, 10\%, 20\%\}$) in a video communication system.
- Also, we use three different packet loss patterns, namely
  1. Independent packet losses,
  2. Packet losses with average burst loss length of 2.5 packets,
  3. Traces of real packet loss patterns over the Internet [60].

We run the simulations for each condition 100 times. PSNR averages are reported in Tables 4.1, 4.2, and 4.3. For each test condition, four numbers are reported: one value is for the average PSNR of the received video in system **A**, one value is for the average PSNR of the received video in the system **B**, one value is for the average PSNR of the received video in the system **C**, and one value is for the average PSNR of the received video in the

Sequence	**PLR**	**100kbs**				**150kbs**				**200kbs**			
		A	**B**	**C**	**D**	**A**	**B**	**C**	**D**	**A**	**B**	**C**	**D**
Foreman	0.03	35.84	36.02	35.03	35.26	38.01	38.21	37.38	37.61	39.65	39.87	39.02	39.33
	0.05	35.55	35.74	34.70	34.88	37.76	37.94	36.97	37.28	39.41	39.63	38.70	39.03
	0.10	34.96	35.14	33.83	34.13	37.23	37.43	36.34	36.64	38.90	39.10	38.21	38.43
	0.20	34.24	34.44	33.11	33.33	36.61	36.80	35.67	35.93	38.33	38.53	37.43	37.76
Crew	0.03	32.75	32.91	32.08	32.38	34.70	34.88	34.18	34.44	36.18	36.37	35.67	35.97
	0.05	32.57	32.75	31.98	32.15	34.53	34.73	34.02	34.25	36.03	36.21	35.55	35.80
	0.10	32.17	32.35	31.52	31.69	34.19	34.38	33.61	33.86	35.71	35.88	35.20	35.46
	0.20	31.68	31.85	30.84	31.14	33.76	33.95	33.14	33.40	35.31	35.49	34.78	35.04
Bus	0.03	28.81	28.97	28.25	28.49	30.92	31.10	30.50	30.69	32.65	32.81	32.26	32.42
	0.05	28.53	28.70	27.89	28.16	30.54	30.85	30.11	30.38	32.43	32.60	31.86	32.12
	0.10	27.94	28.08	27.25	27.41	29.90	30.23	29.36	29.65	31.85	32.02	31.12	31.42
	0.20	27.20	27.33	26.27	26.54	28.81	29.04	27.90	28.45	31.14	31.31	30.33	30.62
Soccer	0.03	33.43	33.61	32.86	33.12	35.57	35.77	35.04	35.36	37.24	37.44	36.76	37.04
	0.05	33.26	33.43	32.66	32.90	35.41	35.61	34.97	35.16	37.08	37.25	36.64	36.85
	0.10	32.88	33.06	32.21	32.44	35.05	35.24	34.49	34.74	36.72	36.94	36.19	36.44
	0.20	32.41	32.57	31.56	31.87	34.64	34.82	34.04	34.26	36.32	36.53	35.66	35.98

Table 4.1: PSNR in dB of system **A** (JM 12.4 encoder (JM)+Belfiore's concealment), system **B** (JM 12.4 encoder (JM)+our noncausal error concealment), system **C** (our two-stage encoder (two-stage)+Belfiore's concealment), and system **D** (our two-stage encoder (two-stage)+our noncausal error concealment) when the losses happen independently, with rate control turned on. Encoded bit rate is shown in the top row of the table.

system **D**. Also, to emphasize on the PSNR difference of the received videos, Tables 4.4, 4.5, and 4.6 report the PSNR difference of system **B**, system **C**, and system **D** with the reference system (system **A**).

Since the JM encoder has slightly better rate-distortion performance, we are expecting that systems which use JM encoder provide better PSNR compared to their corresponding systems with two-stage encoder. Also, since the proposed noncausal error concealment algorithm provides better qualities compared to Belfiore's algorithm, we are expecting that systems using proposed noncausal error concealment in their decoders provide better PSNR compared to their corresponding systems with Belfiore's algorithm.

The reported results confirm both of the expectations and we can sort systems from the best quality to the worst quality as follows:

- system **B**
- system **A**
- system **D**

Sequence	PLR	100kbs				150kbs				200kbs			
		A	B	C	D	A	B	C	D	A	B	C	D
Foreman	0.03	36.02	36.21	35.29	35.59	38.17	38.38	37.58	37.88	39.79	39.99	39.27	39.57
	0.05	35.80	36.00	34.98	35.20	37.98	38.17	37.31	37.57	39.62	39.83	38.98	39.29
	0.10	35.39	35.60	34.46	34.66	37.62	37.82	36.92	37.11	39.28	39.51	38.60	38.87
	0.20	34.72	34.90	33.57	33.85	37.04	37.26	36.17	36.39	38.72	38.92	37.91	38.21
Crew	0.03	32.85	33.02	32.37	32.59	34.78	34.96	34.31	34.60	36.27	36.45	35.84	36.11
	0.05	32.74	32.91	32.20	32.38	34.68	34.84	34.20	34.44	36.17	36.37	35.79	35.97
	0.10	32.44	32.60	31.78	32.01	34.43	34.60	33.90	34.13	35.93	36.12	35.43	35.70
	0.20	32.02	32.19	31.30	31.51	34.06	34.23	33.51	33.72	35.58	35.76	35.03	35.33
Bus	0.03	28.90	29.05	28.52	28.73	31.04	31.16	29.89	30.90	32.78	32.94	32.31	32.62
	0.05	28.71	28.85	28.25	28.43	30.86	30.99	29.60	30.62	32.59	32.74	32.12	32.35
	0.10	28.31	28.44	27.73	27.90	30.47	30.61	29.06	30.12	32.20	32.37	31.56	31.86
	0.20	27.70	27.84	26.91	27.11	29.86	30.03	28.44	29.39	31.60	31.76	30.90	31.15
Soccer	0.03	33.52	33.70	33.12	33.30	35.65	35.83	35.20	35.52	37.33	37.50	36.97	37.19
	0.05	33.41	33.59	32.87	33.11	35.55	35.73	35.06	35.35	37.23	37.43	36.81	37.04
	0.10	33.12	33.32	32.57	32.72	35.29	35.47	34.82	35.02	36.97	37.18	36.52	36.73
	0.20	32.72	32.89	32.01	32.19	34.92	35.09	34.40	34.56	36.61	36.79	35.97	36.29

Table 4.2: PSNR in dB of system **A** (JM 12.4 encoder (JM)+Belfiore's concealment), system **B** (JM 12.4 encoder (JM)+our noncausal error concealment), system **C** (our two-stage encoder (two-stage)+Belfiore's concealment), and system **D** (our two-stage encoder (two-stage)+our noncausal error concealment) over a Gilbert channel with $L_B = 2.5$, and with rate control turned on. Encoded bit rate is shown in the top row of the table.

Sequence	PLR	100kbs				150kbs				200kbs			
		A	B	C	D	A	B	C	D	A	B	C	D
Foreman	0.03	35.80	35.98	34.91	35.20	37.98	38.18	37.32	37.56	39.61	39.82	38.94	39.28
	0.05	35.94	36.14	35.21	35.41	38.10	38.32	37.44	37.74	39.72	39.91	39.18	39.44
	0.10	34.85	35.04	33.73	34.01	37.15	37.35	36.34	36.53	38.82	39.00	38.06	38.33
	0.20	33.99	34.19	32.82	33.09	36.40	36.58	35.38	35.71	38.11	38.31	37.33	37.55
Crew	0.03	32.74	32.92	32.09	32.37	34.68	34.86	34.18	34.42	36.17	36.35	35.67	35.96
	0.05	32.82	33.01	32.23	32.50	34.75	34.93	34.23	34.53	36.24	36.42	35.83	36.05
	0.10	32.12	32.30	31.35	31.58	34.10	34.30	33.53	33.76	35.62	35.80	35.17	35.36
	0.20	31.55	31.71	30.75	31.00	33.62	33.81	33.05	33.26	35.18	35.38	34.70	34.91
Bus	0.03	28.74	28.90	28.25	28.43	30.89	31.01	29.59	30.64	32.61	32.76	32.20	32.37
	0.05	28.86	29.02	28.45	28.64	31.01	31.12	29.86	30.82	32.74	32.91	32.39	32.55
	0.10	27.78	27.92	26.97	27.18	29.96	30.11	28.45	29.58	31.69	31.85	30.99	31.23
	0.20	26.99	27.16	26.17	26.33	29.21	29.34	27.60	28.81	30.95	31.12	30.19	30.43
Soccer	0.03	33.41	33.59	32.92	33.10	35.55	35.74	35.11	35.34	37.22	37.40	36.68	37.02
	0.05	33.49	33.67	32.95	33.23	35.62	35.81	35.13	35.45	37.29	37.47	36.88	37.12
	0.10	32.83	33.01	32.11	32.36	35.01	35.22	34.40	34.68	36.68	36.85	36.17	36.38
	0.20	32.32	32.49	31.52	31.77	34.56	34.75	33.91	34.17	36.22	36.40	35.62	35.88

Table 4.3: PSNR in dB of system **A** (JM 12.4 encoder (JM)+Belfiore's concealment), system **B** (JM 12.4 encoder (JM)+our noncausal error concealment), system **C** (our two-stage encoder (two-stage)+Belfiore's concealment), and system **D** (our two-stage encoder (two-stage)+our noncausal error concealment) over measured packet traces from [60], with rate control turned on. Encoded bit rate is shown in the top row of the table.

Sequence	**PLR**	**100kbs**			**150kbs**			**200kbs**		
		Δ_{B-A}	Δ_{C-A}	Δ_{D-A}	Δ_{B-A}	Δ_{C-A}	Δ_{D-A}	Δ_{B-A}	Δ_{C-A}	Δ_{D-A}
	0.03	0.18	−0.81	−0.58	0.20	−0.63	−0.40	0.22	−0.63	−0.32
Foreman	0.05	0.19	−0.86	−0.68	0.18	−0.80	−0.48	0.22	−0.71	−0.38
	0.10	0.19	−1.13	−0.82	0.20	−0.89	−0.59	0.19	−0.70	−0.47
	0.20	0.20	−1.14	−0.91	0.19	−0.94	−0.68	0.20	−0.90	−0.56
	0.03	0.16	−0.67	−0.37	0.18	−0.52	−0.26	0.20	−0.51	−0.21
Crew	0.05	0.18	−0.59	−0.42	0.20	−0.51	−0.29	0.18	−0.48	−0.23
	0.10	0.18	−0.65	−0.48	0.19	−0.58	−0.33	0.17	−0.51	−0.25
	0.20	0.16	−0.84	−0.54	0.19	−0.61	−0.36	0.18	−0.53	−0.27
	0.03	0.16	−0.55	−0.32	0.18	−0.42	−0.24	0.16	−0.40	−0.24
Bus	0.05	0.16	−0.65	−0.37	0.31	−0.43	−0.16	0.17	−0.57	−0.31
	0.10	0.14	−0.69	−0.52	0.33	−0.54	−0.25	0.18	−0.73	−0.42
	0.20	0.14	−0.93	−0.66	0.23	−0.91	−0.36	0.17	−0.81	−0.52
	0.03	0.18	−0.57	−0.31	0.21	−0.52	−0.21	0.19	−0.48	−0.20
Soccer	0.05	0.17	−0.60	−0.36	0.20	−0.43	−0.25	0.17	−0.43	−0.23
	0.10	0.18	−0.67	−0.44	0.19	−0.56	−0.31	0.22	−0.53	−0.29
	0.20	0.15	−0.85	−0.55	0.18	−0.61	−0.38	0.21	−0.66	−0.34

Table 4.4: PSNR difference of different systems with system **A** in dB when the losses happen independently, with rate control turned on. Encoded bit rate is shown in the top row of the table. System **A** is (JM 12.4 encoder (JM)+Belfiore's concealment), system **B** is (JM 12.4 encoder (JM)+our noncausal error concealment), system **C** is (our two-stage encoder (two-stage)+Belfiore's concealment), and system **D** is (our two-stage encoder (two-stage)+our noncausal error concealment)

Sequence	**PLR**	**100kbs**			**150kbs**			**200kbs**		
		Δ_{B-A}	Δ_{C-A}	Δ_{D-A}	Δ_{B-A}	Δ_{C-A}	Δ_{D-A}	Δ_{B-A}	Δ_{C-A}	Δ_{D-A}
	0.03	0.19	−0.73	−0.43	0.21	−0.59	−0.29	0.20	−0.52	−0.21
Foreman	0.05	0.20	−0.82	−0.60	0.19	−0.68	−0.41	0.21	−0.63	−0.32
	0.10	0.21	−0.94	−0.74	0.20	−0.70	−0.51	0.24	−0.68	−0.41
	0.20	0.18	−1.15	−0.87	0.22	−0.87	−0.65	0.20	−0.81	−0.51
	0.03	0.17	−0.48	−0.26	0.18	−0.47	−0.18	0.19	−0.42	−0.15
Crew	0.05	0.18	−0.54	−0.36	0.16	−0.48	−0.25	0.21	−0.37	−0.19
	0.10	0.16	−0.66	−0.43	0.17	−0.52	−0.30	0.19	−0.50	−0.23
	0.20	0.17	−0.72	−0.51	0.17	−0.55	−0.33	0.18	−0.55	−0.25
	0.03	0.15	−0.38	−0.17	0.12	−1.15	−0.14	0.16	−0.46	−0.15
Bus	0.05	0.15	−0.45	−0.27	0.13	−1.25	−0.23	0.15	−0.47	−0.23
	0.10	0.13	−0.59	−0.42	0.14	−1.41	−0.35	0.17	−0.64	−0.34
	0.20	0.15	−0.79	−0.59	0.17	−1.42	−0.47	0.16	−0.70	−0.45
	0.03	0.18	−0.40	−0.22	0.18	−0.45	−0.13	0.17	−0.36	−0.14
Soccer	0.05	0.18	−0.54	−0.30	0.18	−0.49	−0.19	0.20	−0.42	−0.19
	0.10	0.20	−0.55	−0.40	0.18	−0.46	−0.27	0.20	−0.45	−0.24
	0.20	0.17	−0.71	−0.53	0.17	−0.53	−0.36	0.17	−0.64	−0.33

Table 4.5: PSNR difference of different systems with system **A** in dB over a Gilbert channel with $L_B = 2.5$, with rate control turned on. Encoded bit rate is shown in the top row of the table. System **A** is (JM 12.4 encoder (JM)+Belfiore's concealment), system **B** is (JM 12.4 encoder (JM)+our noncausal error concealment), system **C** is (our two-stage encoder (two-stage)+Belfiore's concealment), and system **D** is (our two-stage encoder (two-stage)+our noncausal error concealment)

Sequence	**PLR**	**100kbs**			**150kbs**			**200kbs**		
		Δ_{B-A}	Δ_{C-A}	Δ_{D-A}	Δ_{B-A}	Δ_{C-A}	Δ_{D-A}	Δ_{B-A}	Δ_{C-A}	Δ_{D-A}
Foreman	0.03	0.17	−0.89	−0.60	0.19	−0.66	−0.43	0.20	−0.67	−0.33
	0.05	0.20	−0.72	−0.52	0.22	−0.66	−0.36	0.19	−0.54	−0.28
	0.10	0.19	−1.12	−0.84	0.20	−0.81	−0.62	0.19	−0.76	−0.49
	0.20	0.19	−1.18	−0.90	0.18	−1.02	−0.69	0.20	−0.77	−0.56
Crew	0.03	0.18	−0.65	−0.38	0.17	−0.51	−0.26	0.18	−0.50	−0.21
	0.05	0.18	−0.60	−0.33	0.18	−0.52	−0.23	0.19	−0.40	−0.19
	0.10	0.18	−0.77	−0.54	0.20	−0.57	−0.34	0.18	−0.45	−0.26
	0.20	0.16	−0.80	−0.55	0.18	−0.58	−0.36	0.20	−0.48	−0.27
Bus	0.03	0.15	−0.49	−0.31	0.12	−1.30	−0.25	0.15	−0.41	−0.24
	0.05	0.16	−0.41	−0.22	0.11	−1.15	−0.19	0.17	−0.35	−0.19
	0.10	0.14	−0.80	−0.59	0.14	−1.52	−0.38	0.16	−0.69	−0.46
	0.20	0.17	−0.83	−0.66	0.13	−1.61	−0.40	0.17	−0.76	−0.52
Soccer	0.03	0.18	−0.49	−0.31	0.20	−0.43	−0.21	0.18	−0.54	−0.20
	0.05	0.18	−0.54	−0.26	0.19	−0.48	−0.17	0.18	−0.41	−0.17
	0.10	0.19	−0.72	−0.47	0.21	−0.61	−0.33	0.17	−0.51	−0.30
	0.20	0.17	−0.80	−0.55	0.19	−0.65	−0.38	0.18	−0.60	−0.34

Table 4.6: PSNR difference of different systems with system **A** in dB over measured packet traces from [60], with rate control turned on. Encoded bit rate is shown in the top row of the table. System **A** is (JM 12.4 encoder (JM)+Belfiore's concealment), system **B** is (JM 12.4 encoder (JM)+our noncausal error concealment), system **C** is (our two-stage encoder (two-stage)+Belfiore's concealment), and system **D** is (our two-stage encoder (two-stage)+our noncausal error concealment)

- system **C**

In addition to the end user PSNR quality, another important factor which should be taken into account when selecting a system suitable for a certain application is the computational complexity of each of the elements of the system. Table 4.7 uses the reported results in tables 2.5 and 3.3 to compare system **A**, system **B**, system **C**, and system **D** in terms of their server side and client side speed. Based on this comparison, system **D** is the only feasible real time system that can be implemented in software. Meanwhile, its performance is within 1dB of the state-of-the-art system **A**.

4.2 Conclusions

Both error concealment algorithms used for these systems are designed based on the estimation of motion vectors for a lost frame. Also both of the methods assume that objects

System name	Realtime encoder	Realtime decoder
System **A**	✗	✗
System **B**	✗	✓
System **C**	✓	✗
System **D**	✓	✓

Table 4.7: Comparison between computation complexity of different systems.

do not have acceleration in their movement, and their speed and direction do not change dramatically between neighboring frames. In the JM encoder, motion vectors are more accurate than the two-stage encoder. Therefore, when a bitstream is encoded using a JM encoder, the zero acceleration remains more realistic for both error concealment algorithms. In addition, in two-stage encoder, some inaccuracy for motion vectors exists. The mechanism which is used in both error concealment algorithms can not vanish this inaccuracy, therefore it accumulates the error to the motion recovery error. In conclusion, both error concealment algorithms provide better performance, when they are paired with JM encoder.

Chapter 5

Channel Coding for Video Multicast

Successful development of network technologies and their increasing popularity, provide a great demand for new applications for new networks. One fascinating application is providing a service such as video delivery for multiple users. As an example, nowadays, many news websites provide video content as part of their news coverage. Even though current network technologies are fairly capable of providing an acceptable video delivery service for a single client, the technology does not provide enough resources of video streaming for multiple clients. Therefore, in this challenge, video communication systems need to be equipped with specific tactics. Technically, there are four different approaches for video delivery for multiple users.

- **Virtual Multicast:** In this approach, the server virtually provides a multicast streaming for a group of clients. In fact, the server starts a separate unicast for each user [32]. This approach is quite simple and easy to implement. But, it utilizes more resources than necessary and suffers from the lack of efficiency. Thus, the maximum number of clients is highly restricted depending on the available resources at the server and the bandwidth.

- **Multicast:** In contrast to virtual multicast, the clients in video multicast session share the same channel; hence the server can put the data or stream on the channel once and clients can receive it. If a feedback channel is available, the users can send feedback, therefore the server can monitor the transmission quality and interact with users, but usually feedback implosion is an issue which needs to be considered carefully.

- **Broadcast:** In contrast to video multicasting, a video broadcasting server is responsible for a large number of users, up to millions in TV stations [25]. Therefore, the server can not monitor the receivers' quality or channel conditions, and can not interact with the users.

- **Peer to Peer:** In contrast to the traditional client-server applications, where the servers only provide the data and the users consume the data, a peer-to-peer distributed network is composed of peers that are both suppliers and consumers of data. Each peer utilizes a portion of its uplink bandwidth to provide data for other peers without intermediary hosts or servers. Therefore, these networks do not require any dedicated infrastructure and are self-scaling as the resources of the network increase with the number of users [50]. The major difference between a general peer-to-peer system and a peer-to-peer media streaming system is that in the data sharing mode among peers, the former uses the *open-after-downloading* mode, while the latter uses the *play-while-downloading* mode which implies certain delay constraints [64].

This chapter describes the subset selection scheme that we proposed in [4].

5.1 Conventional Error Control Techniques for Multicasting

Before introducing the proposed coding scheme for video transmission, we need to become familiar with the capabilities of some famous conventional error control techniques.

5.1.1 FEC

It can be shown that in a multicast scenario, the encoder can send data with low error probability with any rate less than the worst channel capacity by using FEC [33]. This rate is an upper bound for error-free data communication rate and the encoder can only achieve the rate if it uses very long code words. Very long code word might provide long delay in decoding, because when error happens the decoder can not provide output until receiving enough parity packets. Furthermore, this situation is not desirable for users with better channel conditions, because they receive more parity packets than they really need, and can not achieve the best possible quality. On the other hand, the encoder can choose a different strategy, for example designing the FEC code for an average channel. Consequently, users with high quality channel conditions can achieve a better quality, but users with poor channel quality may not able to use the FEC due to the low number of parity packets.

5.1.2 ARQ

In ARQ and other retransmission-based algorithms, the encoder retransmits damaged or lost frames. In unicast scenarios, ARQ might provide a long latency in the communication. In multicast scenarios, when the number of users in a multicast group increases, the probability that a frame is lost at least in one of the users, increases. Therefore, the ARQ requires retransmitting more frames. In a multicast group with n users, if losses happen independently for different users with probability P_l, the server should retransmit each frame with probability of $P_{ARQ} = 1 - (1 - P_l)^n$ (Figure 5.1), which decreases the whole system throughput.

5.1.3 Hybrid ARQ/FEC type-II

As mentioned in section 1.5, the encoder can use the combined advantages of ARQ and FEC by using hybrid ARQ/FEC. Hybrid ARQ/FEC type-II is more interesting for multicast

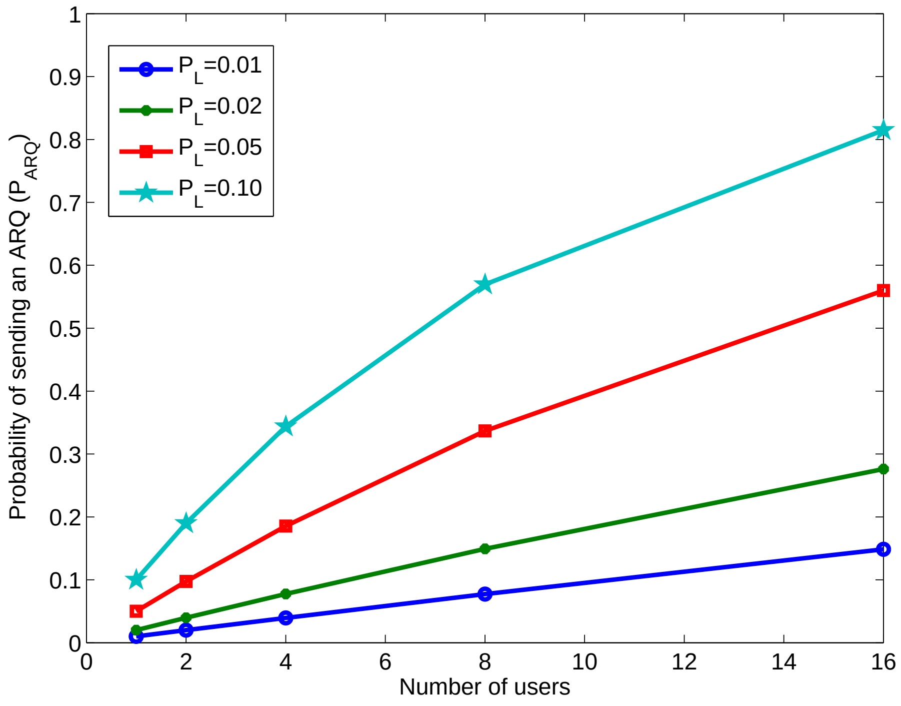


Figure 5.1: Probability of retransmission

scenarios. In Hybrid ARQ/FEC type-II, the encoder first sends a group of packets through the multicast channel. Then each user replies with the number of lost frames. At this point in time, the encoder generates the necessary number of parity packets from a systematic Reed-Solomon code and sends these parity packets to the users. To make every user able to recover all losses, the encoder needs to generate at least as many parity packets as the maximum number of losses at any one user. In this scheme, the encoder adaptively tunes the FEC rate, and also the expected number of required parity packets is less than the expected number of required ARQ retransmissions.

As an example, Figure 5.2 demonstrates a situation in which server sends packets in groups of $N = 12$ packets to n receivers, and after sending each group, it will send parities or ARQ packets, and losses happen independently.

In this situation, (5.1) shows the average number of required ARQ packets. To derive the corresponding equation for the number of required packets for Hybrid ARQ/FEC type-II, we first find the cumulative distribution function of the number of required parity packets, (5.2), then the probability mass function, (5.3). (5.4) can be calculated by using (5.3). ($b(k,n,p)$ is the probability density function of a binomial distribution and $\mathcal{B}(k,n,p)$ is its cumulative function, and l_i is the number of losses in user i)

$$E\left[\#\text{ARQ packets}\right] = N\left(1-\left(1-P_l\right)^n\right) \tag{5.1}$$

$$\begin{aligned} P\left[\max_{i=0,1,\cdots,n}\left(l_i\right) < l_0\right] &= P\left[l_1 < l_0, \cdots, \text{and} l_n < l_0\right] = P\left[l < l_0\right]^n \\ &= \left(\mathcal{B}\left(l_0, N, P_l\right)\right)^n \end{aligned} \tag{5.2}$$

$$\begin{aligned} P\left[\max_{i=0,1,\cdots,n}\left(l_i\right) = l_{max}\right] &= P\left[\max_{i=0,1,\cdots,n}\left(l_i\right) < l_{max}+1\right] - P\left[\max_{i=0,1,\cdots,n}\left(l_i\right) < l_{max}\right] \\ &= \left(\mathcal{B}\left(l_{max}+1, N, P_l\right)\right)^n - \left(\mathcal{B}\left(l_{max}, N, P_l\right)\right)^n \end{aligned} \tag{5.3}$$

$$E\left[\#\text{Parity packets in type-II Hybrid ARQ/FEC}\right] = \sum_{l=1}^{N} l.\left[\left(\mathcal{B}\left(l+1, N, P_l\right)\right)^n - \left(\mathcal{B}\left(l, N, P_l\right)\right)^n\right] \tag{5.4}$$

5.1.4 RPS

RPS has been studied in subsection 1.3.2 for unicast scenarios. But, RPS does not show an acceptable performance in multicast scenarios [11]. The main problem with RPS happens when we have a large multicast group. In such a situation, it is difficult to find a reference frame which has been received correctly by all users. However, there are situations in which RPS can be suitable. For example where errors in different users are correlated or the channel has burst error. For these situation, it is easier for the encoder to find an appropriate reference frame.

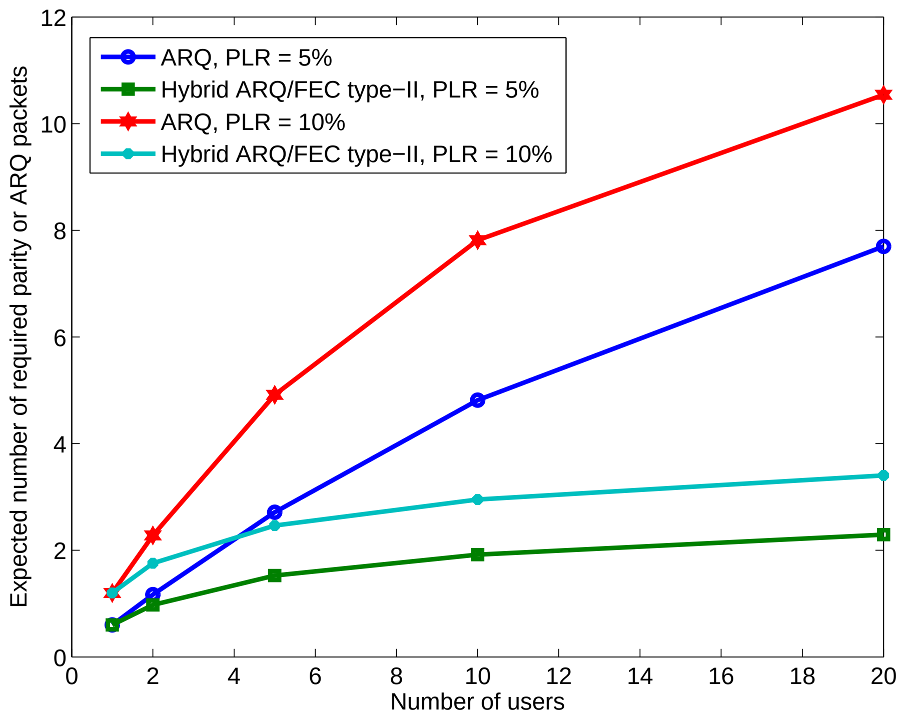


Figure 5.2: Comparison of ARQ and Hybrid ARQ/FEC type-II.($N = 12$)

5.1.5 Rate Distortion Optimization (RaDiO) and Collision Distortion Optimization (CoDiO)

These state-of-the-art methods provide media packet scheduling based on, respectively, rate-distortion and congestion-distortion optimization. A good overview of these methods can be found in [35]. Using a general distortion measure, RaDiO provides a transmission schedule of media packets that minimizes the expected distortion under a bandwidth constraint. Thus, the server requires knowledge about the available bandwidth. On the other hand, CoDiO provides a similar schedule under an end-to-end delay constraint, which makes the server independent of any bandwidth allocation or bandwidth estimation algorithm.

5.2 Subset Selection in Hybrid ARQ/FEC for Video Multicast

In this section, we propose an error control technique for video multicast based on the Type-II hybrid ARQ/FEC [41]. The proposed method uses the feedback information about the losses in the previously transmitted Group of Pictures (GOP). Based on this information, the server chooses a subset of frames in the previous GOP, computes parity packets for the selected frames using a Reed-Solomon (RS) encoder, and multicasts these parity packets to the users.

In conventional Type-II hybrid ARQ/FEC, for each block of n packets, each user informs the server of the number of packets (l_i) it has lost in the previous block. The server then finds the maximum (l_{max}) of these numbers and multicasts l_{max} parity packets to the users. Due to limited bandwidth, however, it may be impossible to always send as many parity packets as required for recovering all losses at all users. Consequently, some users may end up being unable to recover any losses, because they did not receive a sufficient number of parity packets.

To solve this problem, we propose the following strategy. Instead of assigning parity for the entire block of packets, the server selects a subset of packets for which parity will be assigned. Then it computes the parity packets for this subset. In a more general scenario, the server can select multiple subsets and assign separate parity to each of them. In subsection 5.3, we consider the case of a single subset, and in section 5.6 we consider the case of multiple subsets. This strategy may allow partial loss recovery at some users that otherwise would not able to recover any lost packets. To illustrate this point, consider the situation shown at the top of Figure 5.3, where each of the two users has lost two packets (shaded in gray) out of a block of five packets. Suppose the bandwidth limitations allow the server to multicast only one parity packet for this block. If the parity is computed across all five packets in the block, neither user can recover any of its lost packets. However, if the parity

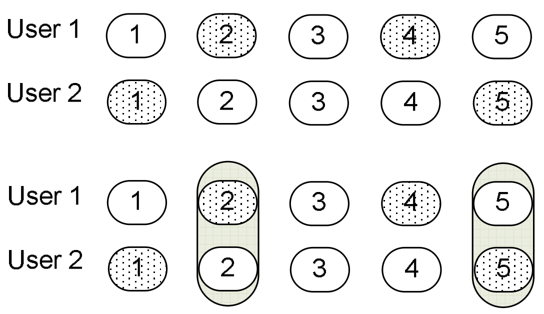


Figure 5.3: Motivation for subset selection in Type-II hybrid ARQ/FEC.

is computed only for, say, packets 2 and 5, as shown in the bottom of Figure 5.3, then user 1 will be able to recover packet 2, and user 2 will be able to recover packet 5.

In the context of video multicast, each packet affects decoding of one or more frames at all users. The server needs to consider the effect of the loss of each packet in order to select the subset of packets for which parity will be generated. The best subset of packets in this context is the one that will maximize decoded video quality across all users.

5.3 Subset Selection in Type-II Hybrid ARQ/FEC for Video Multicast with One Subset

5.3.1 Full Search for the Best Subset

Before introducing the full search algorithm, we need to define several symbols.

- $E^{\emptyset}$ is the loss indicator matrix which indicates the loss or reception of each packet at each user. It is generated based on user feedback as follows.

$$E^{\emptyset}[i,j] = \begin{cases} 0, & j\text{-th frame of } i\text{-th user received,} \\ 1, & j\text{-th frame of } i\text{-th user lost.} \end{cases} \tag{5.5}$$

  The i-th user sends its feedback to the server as a bitmap corresponding to the i-th row of $E^{\emptyset}$. For typical GOP sizes of 15 to 30 frames, feedback information requires less then 4 Bytes/GOP/user, which is reasonably small.

- $E^{k,S}$ is the loss indicator matrix after the correct reception of k parity packets for the subset S of the packets from the GOP. Since k parity packets will correct all losses at users that have k or fewer lost packets in S, we can find $E^{k,S}$ as follows:

$$E^{k,S}[i,j] = \begin{cases} 0, & j \in S \text{ and } \sum_{f \in S} E^{\emptyset}[i,f] \leq k, \\ E^{\emptyset}[i,j], & \text{otherwise.} \end{cases} \tag{5.6}$$

- $D_i^{\emptyset}$ is the decoded video distortion of one GOP at user $i \in \{1, 2, ..., N\}$ prior to receiving any parity packets for that GOP. Note that $D_i^{\emptyset}$ is a function of the loss indicator matrix $E^{\emptyset}$ or, more precisely, a function of the i-th row of $E^{\emptyset}$. Its estimation is discussed below.

- $D^{\emptyset}$ is the total distortion at all users for one GOP prior to receiving any parity packets for that GOP. It is computed as the sum of individual user's distortions: $D^{\emptyset} = \sum_{i=1}^{N} D_i^{\emptyset}$.

- $D_i^{k,S}$ is the distortion at user i after receiving k parity packets for subset S of the packets from the GOP. It is a function of the i-th row of the loss indicator matrix $E^{k,S}$.

- $D^{k,S}$ is the total distortion at all users after receiving k parity packets for subset S of the packets from the GOP, defined as $D^{k,S} = \sum_{i=1}^{N} D_i^{k,S}$.

In the following, we assume that the users buffer the packets from a GOP and wait for the error control packets for that GOP before decoding and playing out the frames. This is a reasonable assumption, since the common media players such as Windows Media Player or RealPlayer usually use buffers that are at least several seconds long [10].

Estimating distortion at each user (i.e., finding $D_i^{\emptyset}$ and $D_i^{k,S}$) is not an easy task. In principle, one could simply decode the given GOP at the server using the loss pattern specified by the i-th row of the loss indicator matrix to find the exact distortion, but this process is extremely time-consuming when there are many users, and needs to be repeated for each GOP. Several methods have been proposed to estimate the distortion at the decoder, including Recursive Optimal per-Pixel Estimation of decoder distortion (ROPE) [67] and Multi-Decoder Distortion Estimation (MDDE) [54], among others. Many of these distortion estimation methods are not suitable for our task, because they provide an estimate of the distortion as a function of the packet loss rate, rather than a given loss pattern. Instead, we adopt the technique of Setton *et al.* [49, 48], who proposed a very simple and effective measure of the importance of a frame, which can be related to the distortion caused by the loss of that frame. In their technique, the importance of a given frame is equal to the number of frames that would be affected by the loss of that frame. They show that the performance of a video streaming system that uses this measure of importance is fairly close to the performance achieved by using the exact distortion.

To illustrate this measure of importance of a frame, consider the example in Figure 5.4 that shows a GOP of size 12 frames with structure IBBBPBBBP... . If a user loses an I-frame, decoding of 15 frames will be affected (12 frames from the same GOP plus three B-frames from the previous GOP), because the I-frame is used as a direct or indirect reference for encoding these frames. Hence, in the scheme of Setton *et al.* [49, 48], the I-frame would receive a weight of 15. The weights for other frames in this example would be as shown in the bottom part of Figure 5.4. Note that the weights in this scheme are associated with the loss of a single frame. For example, the recovery of a lost I-frame would improve the quality of 15 frames provided no other frame in that GOP is lost; if that is not the case, the number of affected frames may be different. Despite this deficiency, we adopt the described method as a simple and effective way to help us choose which packets should receive parity in our error control scheme. We assume that each frame is stored in a single packet, so the importance (weight) of a frame becomes the weight of its packet.

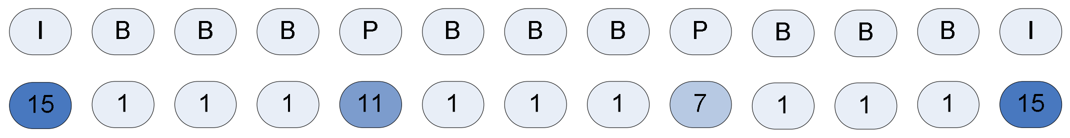


Figure 5.4: Different frames have different importance for the decoded video quality.

Because the maximum packet size (MTU) is limited for different networks, it is possible to have an encoded frame larger than the MTU, especially when the encoder is dealing with I-frames. In this situation, encoders usually use more than one slice to keep the size of each encoded slice less than the MTU. Therefore, the proposed distortion model in [49, 48] is not appropriate to this problem and needs to be modified. One modification can be assuming that by losing one slice, the corresponding frame will be partially distorted. For other frames, which use this frame as their reference, this error will propagate and distort larger part of them. By summing up all these distorted parts of frames, one can find a more accurate approximation of the total distortion in a GOP.

In the subset selection algorithm, the server needs to know which packets have been received for each user. Therefore each user must send a feedback message to the server about its received packets and lost packets. When the server deals with a large multicast group, collecting feedback from all users requires a large amount of resources (especially time and bandwidth). This problem is usually called feedback implosion in the literature [11]. To prevent this problem in the subset selection algorithm, we can divide users into several smaller subgroups, then for each subgroup choose a proxy to manage feedbacks from the members of that subgroup and to send a summary of all feedback messages to the server or higher proxy layers. The important information for the server is the total amount of distortion removed by recovering each frame in the multicast group. Hence, the summarization procedure in each subgroup can easily provide this information for its own subgroup and send it for the server. This procedure involves the addition of the distortion estimations of all users together and therefore it does not require much CPU usage.

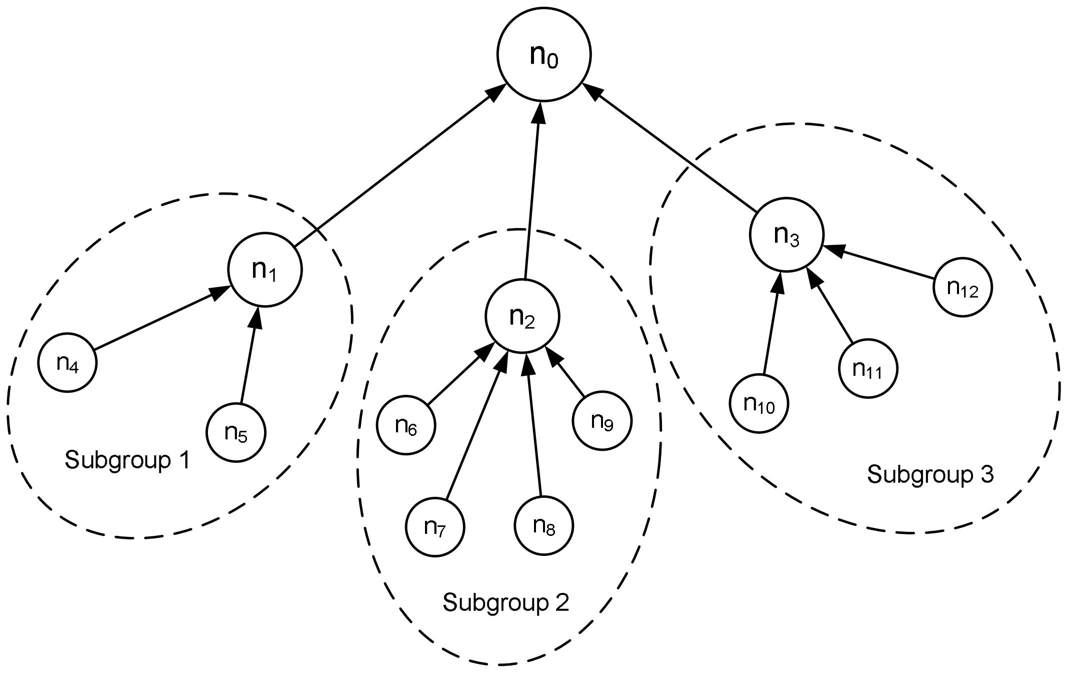


Figure 5.5: Creating subgroups of receiver for gathering feedback from receivers when the server deals with a large group of clients

For example, in Figure 5.5, the server (n_0) multicasts information for twelve users. To prevent feedback implosion, we can divide users into 3 subgroups and choose a proxy for each subgroup (user n_1 for subgroup 1, user n_2 for subgroup 2, and user n_3 for subgroup 3). Each proxy receives feedback from its subgroup members, and then provides the distortion estimation message for the whole subgroup and transmits the summary to the server (n_0).

The full search algorithm for finding the best subset is based on the exhaustive search through all possible subsets of the packets from the GOP. The algorithm estimates what the total distortion, $D^{k,S}$, across all users would be after sending k parity packets for each subset S. Finally, it returns the subset $S_{optimal}$ which promises the lowest total distortion.

$$S_{optimal} = \arg\min_{S} \left(D^{k,S} \right). \tag{5.7}$$

The value for k is usually determined by the constraints of the system, such as the maximum delay or the maximum available bandwidth. To illustrate how subset selection

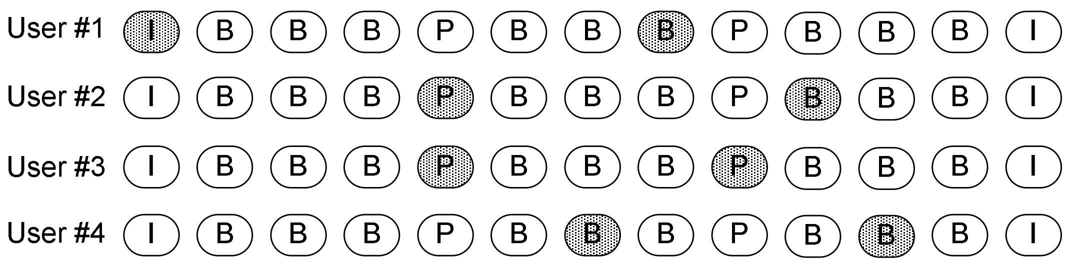


Figure 5.6: Lost frames indicated by darker shading.

works, assume there are four users in the system, and the loss pattern for a particular GOP is as shown in Figure 5.6, where darker shading indicates lost packets/frames. Also, assume the available bandwidth only allows the server to send one error control packet ($k = 1$), either by retransmitting one of the lost packets, or by sending one parity packet. In this example, a total of eight frame losses across four users cause a total of 39 distorted frames (across all users). Let us examine how various error control schemes might react in this case.

ARQ

Being a transport (or sometimes link) layer error control scheme, ARQ has no concept of packet or frame importance. It may choose to retransmit any one of the seven lost frames, for example the ninth frame of the GOP, as shown in Figure 5.7. In this particular case, even if the retransmitted frame is received correctly, none of the lost frames can be fully recovered, because the ninth frame of user #3 is dependent on the fifth frame, which is also lost.

Type-II hybrid ARQ/FEC

One parity packet will be generated for the entire GOP and multicast to the users. However, even if correctly received, this parity packet will not help any of the users, because each user has lost two packets.

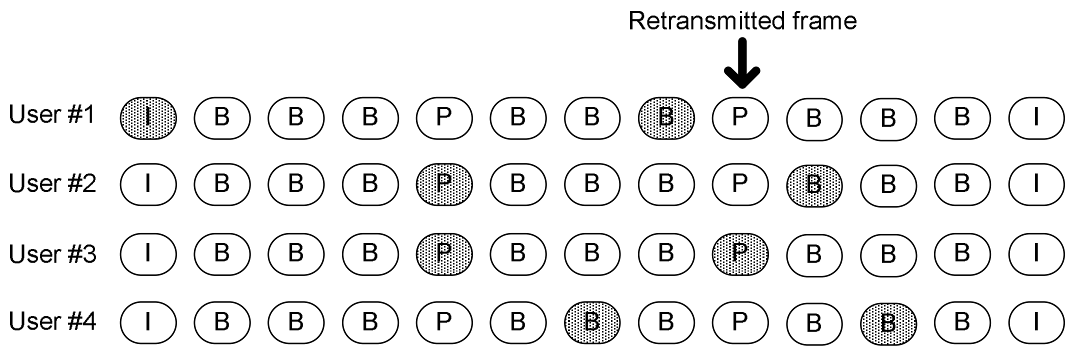


Figure 5.7: ARQ may choose to retransmit any of the lost frames, say the ninth frame, since it cannot distinguish their importance. In this particular case, there is no benefit even to user #3, because the ninth frame depends on the fifth frame, which is also lost.

RaDiO and CoDiO

These state-of-the-art methods provide media packet scheduling based on, respectively, rate-distortion and congestion-distortion optimization. A good overview of these methods can be found in [35]. Using a general distortion measure, RaDiO provides a transmission schedule of media packets that minimizes the expected distortion under a bandwidth constraint. On the other hand, CoDiO provides a similar schedule under an end-to-end delay constraint. Although neither of these schemes considers multicast explicitly, for the purposes of this paper, we could simply define the distortion measure as $D^{k,S}$ in the previous section, and run these algorithms to decide on scheduling retransmissions. Figure 5.8 shows how RaDiO or CoDiO might work in our example. Because only one retransmission is allowed, both algorithms search over different possible retransmissions and try to find the one which would minimize $D^{k,S}$. In this particular case, it is best to retransmit the fifth frame of the GOP, because this leads to the recovery of 14 frames across the users.

Proposed Subset Selection for Type-II hybrid ARQ/FEC

With this scheme, the server will select the best subset of packets/frames from the GOP and assign one parity packet to them. The best subset, $S_{optimal}$, in this case is shown in

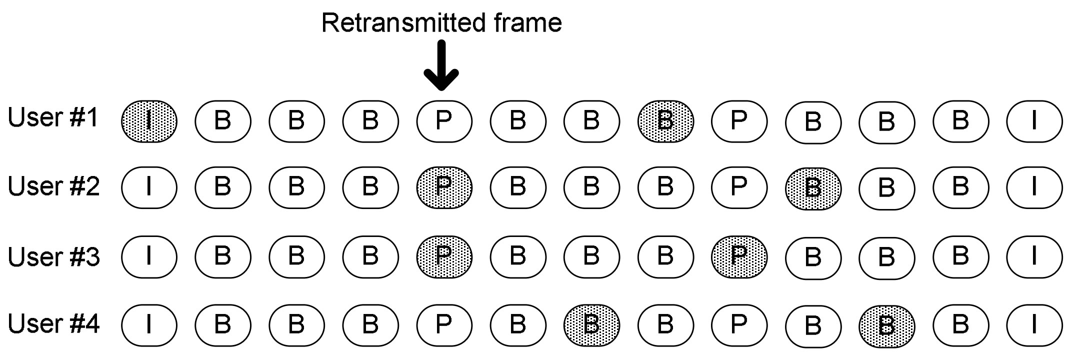

Retransmitted frame
User #1
I B B B P B B B P B B B I
User #2
I B B B P B B B P B B B I
User #3
I B B B P B B B P B B B I
User #4
I B B B P B B B P B B B I

Figure 5.8: RaDiO and CoDiO schedulers retransmit the fifth frame of the GOP, which leads to the recovery of 14 frames across the users. In this case, only users #2 and #3 benefit from the retransmission.

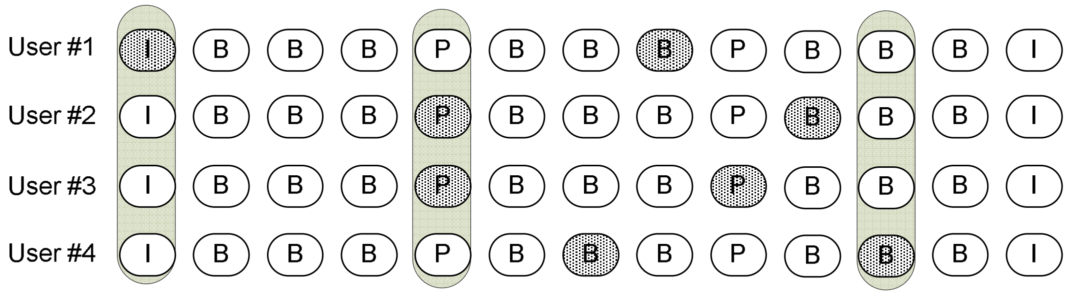

User #1
I B B B P B B B P B B B I
User #2
I B B B P B B B P B B B I
User #3
I B B B P B B B P B B B I
User #4
I B B B P B B B P B B B I

Figure 5.9: The optimal subset for parity generation is composed of the first, fifth, and eleventh frame of the GOP. Reception of the parity packet at all users would lead to the recovery of 26 frames.

Figure 5.9, and is composed of the first, fifth, and eleventh frame. The reception of the parity packet for this susbset will allow user #1 to recover the first frame of the GOP, users #2 and #3 will be able to recover the fifth frame of the GOP, and user #4 will be able to recover the eleventh frame of the GOP. This would lead to the recovery of the total of 26 frames across the users. Note that the optimal subset is not unique in this case. For example, we could include the seventh (instead of eleventh) frame, or the second frame (which has not been lost at any of the users) into the subset, and obtain the same performance.

5.3.2 Suboptimal Search Based on Simulated Annealing

High compression efficiency usually requires choosing fairly large GOPs. However, since the number of subsets grows exponentially with the size of the GOP, subset selection via full search becomes impractical for large GOPs, as will be illustrated in Section 5.4. Because of this, we develop a suboptimal, but faster, search method based on simulated annealing [37]. Algorithm 7 shows how simulated annealing can be used to solve the subset selection problem. S_{best} stores the best subset selected until now and D_{min} is the corresponding estimated distortion. Current subset, S, is initialized with a random subset of the GOP. Then, for each iteration in the main loop, the algorithm randomly finds one of the "neighbors" (S_n) of the current subset S, and estimates the corresponding D^{k,S_n}. In our implementation, a "neighbor" of the subset is another subset which has either one more frame, or one less frame in it. The algorithm accepts S_n as the current subset S with probability $P\left(D, D_n, iter/iter_{max}\right)$ based on the Metropolis state calculation rule [43]:

$$P\left(D, D_n, \frac{iter}{iter_{max}}\right) = \begin{cases} 1, & D_n < D, \\ exp\left(\frac{D_n - D}{1 - \frac{iter}{iter_{max}}}\right), & D_n \geq D. \end{cases} \tag{5.8}$$

Hence, when $D_n < D$, S_n is accepted as the current subset S with probability 1. Otherwise, S_n can still be accepted with a certain probability which reduces as the number of iterations ($iter$) increases. This helps the algorithm escape some of the local optima. The maximum number of iterations ($iter_{max}$) was set to 3000 in our experiments.

5.4 Experimental Results

In our simulations, we used two standard test sequences (*Foreman* and *Bus*), both CIF resolution at 30 fps, encoded using the H.264 JM 12.4 encoder. Unless otherwise specified, the GOP was 16 frames long, with a structure IBBBPBBBP... The server multicasts video packets to five users, each of which sees an independent packet loss channel.

Algorithm 7 Subset selection based on simulated annealing

```
$S_{best} \leftarrow GOP; D_{min} \leftarrow D^{k,GOP}$
$S \leftarrow RandomSubset\,(GOP)\,; D \leftarrow D^{k,S}; iter \leftarrow 0$
**repeat**
  $S_n \leftarrow neighbor\,(S)\,; D_n \leftarrow D^{k,S_n}$
  **with probability of** $(P\,(D, D_n, iter/iter_{max}))$
  $S \leftarrow S_n; D \leftarrow D_n$
  **if** $D < D_{best}$ **then**
    $S_{best} \leftarrow S; D_{min} \leftarrow D$
  **end if**
  **end with**
  $iter++$
**until** $D_{min} == 0$ or $iter == iter_{max}$
```

In the first set of experiments, we measure the number of frames, across all users, affected by packet loss. These measurements are important because our distortion metric $D^{k,S}$ is directly related to the number of frames affected by packet loss. The results are reported in terms of the probability of affected frames, computed as the ratio of the total number of affected frames to the total number of transmitted frames, across all users. Figure 5.10 shows the probability of affected frames vs. packet loss probability of the two proposed methods for subset selection (Full Search - FS, and Simulated Annealing - SA) and other error control methods mentioned in the previous section - ARQ, Type-II hybrid ARQ/FEC and RaDiO. The performance of a system which does not use error control (labeled as 'Not protected' in the figure) is also shown. In these simulations, the server is allowed to send only one error control packet (either retransmission or parity) per GOP. As seen in the figure, the proposed subset selection methods outperform other error control techniques by a healthy margin. Also note that the results obtained by simulated annealing are virtually indistinguishable from the ones obtained by full search. Similar results are shown in Figure 5.11, where we plot the performance of the error control schemes when two error control packets are allowd per GOP.

In Figure 5.12 we show how the performance varies with the number of users in the

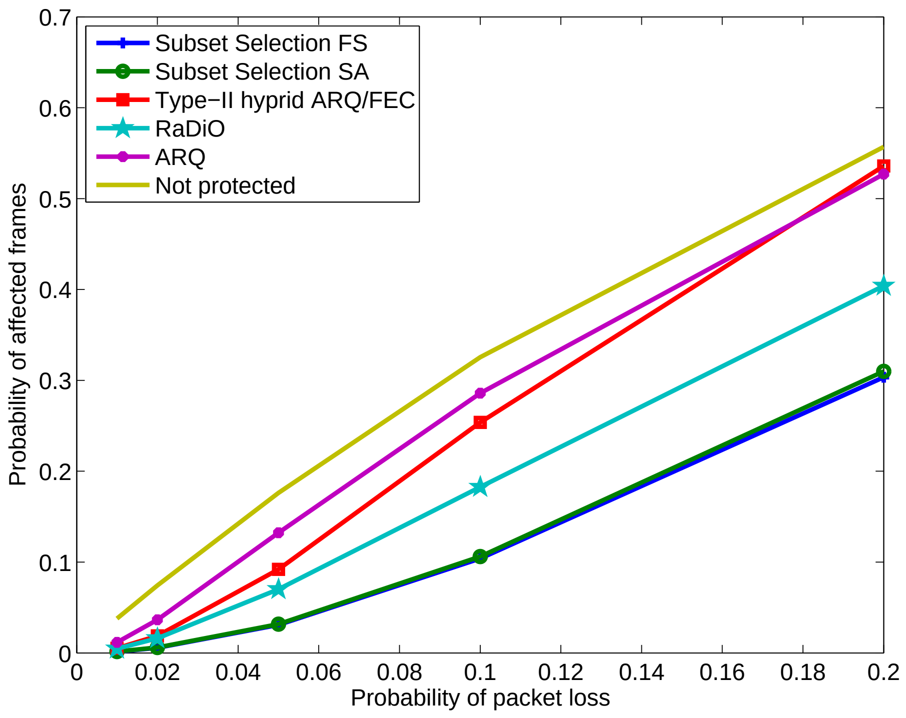


Figure 5.10: Performance comparison of different error control schemes as a function of packet loss probability, with one error control packet per GOP.

system. As before, one error control packet is allowed per GOP. Note that the performance of the proposed subset selection is fairly steady with the number of users, a property inherited from Type-II hybrid ARQ/FEC. ARQ and RaDiO are less scalable, and their performance decreases noticeably as the number of users increases.

In Figure 5.13 we show the Peak Signal-to-Noise Ratio (PSNR) of the Y-component of the *Foreman* sequence, averaged over all users, as a function of the packet loss rate. One error control packet was allowed per GOP. The sequence was encoded at 750 kbps for the cases when error control is used and, for fair comparison, at a higher rate of 800 kbps for the system without error control (Not protected), because this system does not use error control packets. The proposed subset selection can improve upon the plain Type-II hybrid ARQ/FEC by over 4 dB in decoded video PSNR at higher loss rates. In addition, its

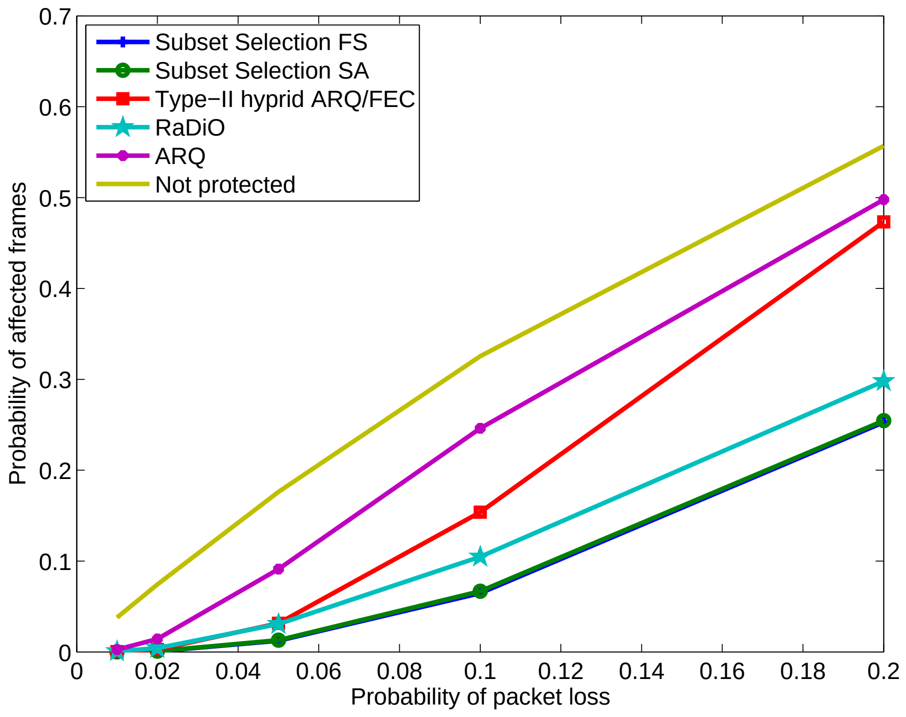


Figure 5.11: Performance comparison of different error control schemes as a function of packet loss probability, with two error control packets per GOP.

advantage compared to the next best scheme, RaDiO, is about 1-1.5 dB.

It is interesting to compare the performance of the non-protected system against plain ARQ and plain Type-II hybrid ARQ/FEC. At very low loss rates, even a single error control packet allows both ARQ and Type-II hybrid ARQ/FEC to provide slight improvement over the non-protected system. At moderate loss rates (around 0.05), non-protected system apparently becomes slightly better than ARQ. Here, users are more likely to lose two or more packets per GOP, while ARQ can only recover one of them, so it seems better to spend bits on video coding rather than retransmission. However, at higher loss rates (0.15 and above), ARQ becomes better than either non-protected system or plain Type-II hybrid ARQ/FEC. Now, many users lose more than one packet per GOP, so plain Type-II hybrid ARQ/FEC with one parity packet cannot recover any of them. ARQ at least recovers one

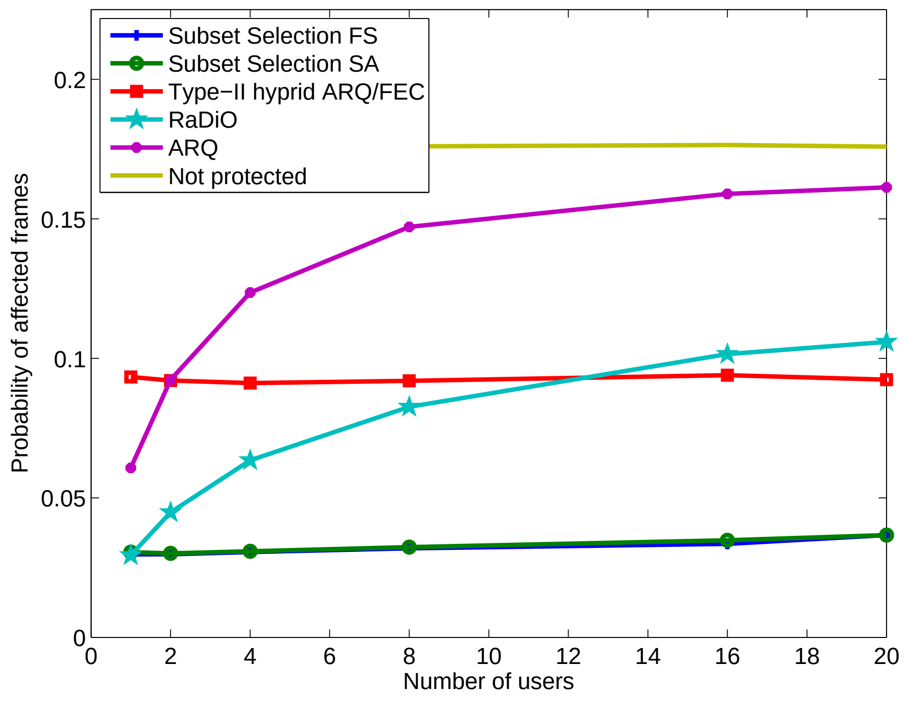


Figure 5.12: Performance comparison of different error control schemes as a function of the number of users, with one error control packet per GOP.

lost packet.

Finally, to assess the practicality of the proposed subset selection algorithms, we implemented them in C++, compiled them using the Microsoft Visual Studio 2005, and ran them on an Intel Core 2 machine with 2.13 GHz CPU and 2 GB of RAM. We measured the CPU usage when running subset selection for 5 users at 30 frames per second. The results, summarized in Table 5.1, show that full search is only practical for short GOPs (in this configuration, 12 frames or less), while the proposed suboptimal algorithm can run with an acceptable CPU usage even for long GOPs. For GOP sizes of 20 and higher, full search took much longer than the time interval of the GOP, so it was not able to run in real time. Hence, no result is shown in the table for these cases.

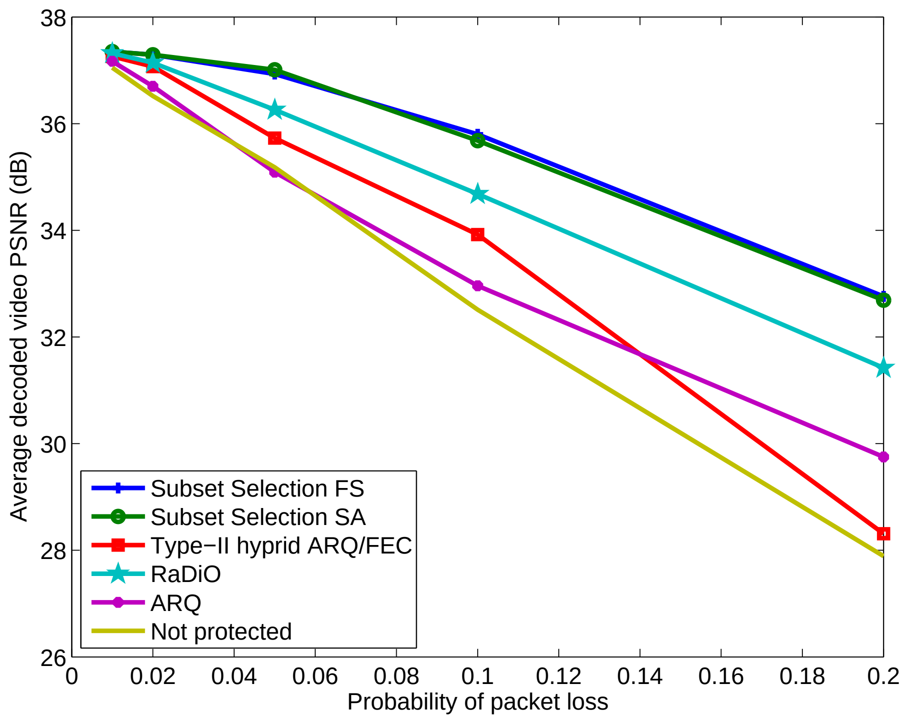


Figure 5.13: Average PSNR in dB for the *Foreman* sequence with one error control packet per GOP.

5.5 Analytical Results

In subsection 5.4, we demonstrated the performance of subset selection based on experimental results. In this subsection, we examine the performance of subset selection analytically.

GOP size	4	8	12	16	20	24
FS	0.01%	0.17%	2.14%	32.70%	-	-
SA	0.78%	0.78%	0.78%	0.81%	0.87%	0.99%

Table 5.1: CPU usage by the two subset selection algorithms.

5.5.1 Error Distribution

Before starting our analysis, we need to find the *error distribution* in different channel conditions. We define error distribution for a block of received packets as $P(m, n)$, which determines the probability of having m lost packets among n transmitted packets. For multicast environments, Four popular channel models have been used in the literature [55, 65]. In all of these models, the assumption is that the channel qualities between different users and the server are identical.

Model 1: In model 1, losses happen independently between different users, and for each user losses at different times happen independently. In this model, users experience losses independently and identically, therefore $P(n, m)$ can be calculated independently for one of the users and be used for others. In this case, $P(n, m)$ is a simple binominal distribution, (5.9), where P_l is the packet loss rate:

$$P(m, n) = \binom{m}{n} (1 - P_l)^{n-m} p_l^m. \tag{5.9}$$

Model 2: In model 2, losses happen independently between different users, and for each user, losses at different times are correlated. To model the temporal correlation between losses, we use Gilbert model (2-state Markov process), as depicted in Figure 5.14. In this model, state B stands for bad channel condition, where a loss happens. State G stands for a good channel condition, where packet will be received correctly. The channel state can be changed from good condition to bad condition with probability p_{GB}, or remain in good condition with probability $1 - p_{GB}$. In the same way, the channel state can be changed from bad condition to good condition with probability p_{BG}, or remain in bad condition by probability of $1 - p_{BG}$.

To find the error distribution function for a Gilbert channel, we modify the technique

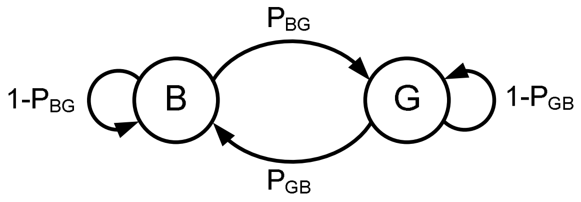


Figure 5.14: Gilbert Model. p_{BG} be the probability of going to a G state from a B state, and p_{GB} is the probability of going to a B state from a G state

proposed in [23]. First, let $g(v) = P\left(0^{\nu-1}1 \mid 1\right)^{12}$ be the probability of a gap with length ν, in which after a loss, $\nu - 1$ packets are received correctly. Equation (5.10) shows the value of $g(v)$ for a Gilbert process.

$$g(v) = \begin{cases} 1 - p_{BG}, & \nu = 1, \\ p_{BG}(1 - p_{GB})^{\nu-2} p_{GB}, & \nu > 1. \end{cases} \tag{5.10}$$

Consequently, we can define $G(v) = P\left(0^{\nu-1} \mid 1\right)$ which is the probability of all the gaps with more than $v - 1$ losses. Equation 5.11 shows the value of $G(v)$ for a Gilbert process.

$$G(v) = \begin{cases} 1, & \nu = 1, \\ p_{BG}(1 - pGB)^{\nu-2}, & \nu > 1. \end{cases} \tag{5.11}$$

Let $\pi(m, n)$ be the probability of $m - 1$ packet losses within the next $n - 1$ packets following a lost packet. It can be calculated using the recursive method in (5.12):

$$\pi(m, n) = \begin{cases} G(n), & m = 1, \\ \sum_{\nu=1}^{n-m+1} g(\nu)\, \pi(m-1, n-\nu), & 2 \le m \le n. \end{cases} \tag{5.12}$$

[1]0 indicates a correct reception, and 1 indicates a loss.

[2]0^{α} means α packets are lost consecutively.

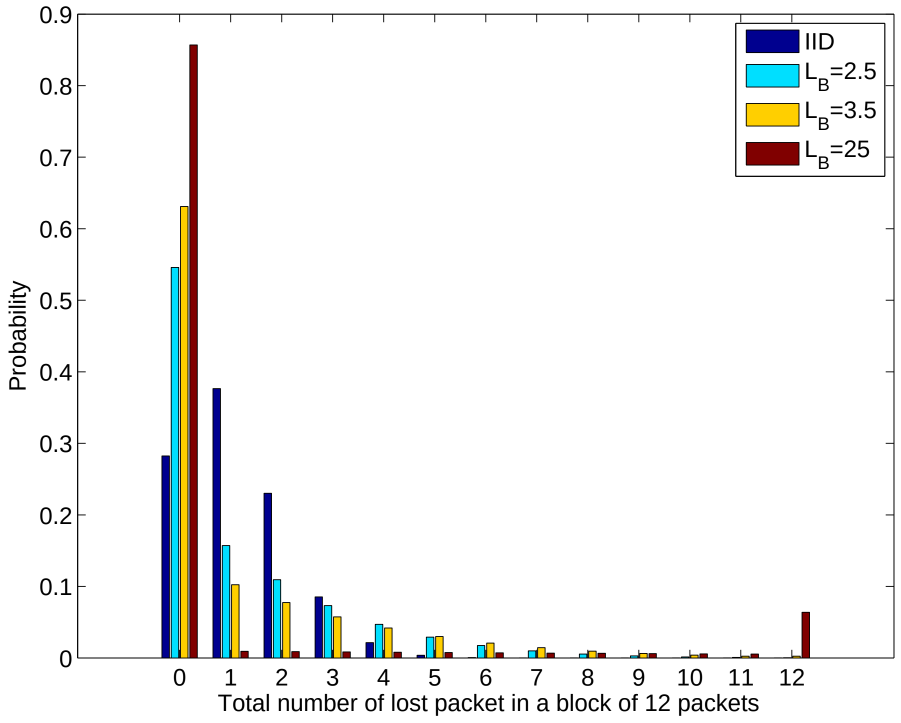


Figure 5.15: Error distribution for different packet loss patterns, when the packet loss rate is 10%.

Now, we can calculate $P(m,n)$ using $\pi(m,n)$ as (5.13), where $P_B = \frac{p_{BG}}{p_{BG}+p_{GB}}$ is the loss probability.

$$P(m,n) = \sum_{\nu=1}^{n-m+1} P_B G(\nu)\,\pi(m,n-\nu+1)\,, 1 \le m \le n \tag{5.13}$$

Figure 5.15 demonstrates the results of using (5.9) and (5.13) for calculating error distribution, $P(m,n)$, for different patterns of packet loss.

Model 3: In model 3, losses between different users are correlated, and for each user losses at different times happen independently. We do not use it in our analysis.

Model 4: In model 3, losses between different users are correlated, and for each user, losses are correlated at different times. This model is the most realistic model, but due to its complexity we do not use it in our analysis.

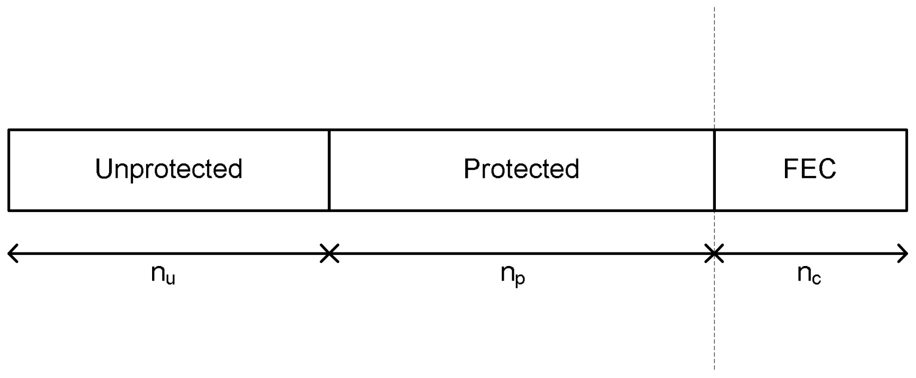


Figure 5.16: A simple subset protection for pure FEC. In this scheme n_u information packet are left unprotected, and the remaining information packets are protected with n_c parity packets

5.5.2 Subset Selection Effects on the Residual Packet Loss Rate without Feedback

Due to the restrictions on the available bandwidth, the encoder sometimes can not send enough parity packets to protect the transmitted information completely. The problem shows up when the number of received parity packets is not high enough for recovering lost packets. In such a condition, the channel decoder can not recover any of the lost packets by parity packets. In this subsection, we will show that subset selection can be useful even it is combined with a pure FEC[3]. Figure 5.16 demonstrates a simple structure to combine subset selection with FEC. In this structure, the transmitter has $n = n_u + n_p$ packets to send. The first n_u packets are left unprotected, and the next n_p packets are protected with n_c parity packets.

The average number of lost information packets (l_t) is simply equal to the average number of lost packets in unprotected area (l_u) plus the average number of lost packets in protected area (l_p). The average number of lost packets in the unprotected (l_u) part of this structure can easily be calculated by $l_u = pn_u$.

[3]when there is no feedback available.

If the total number of lost packets in the protected part and parity packets is not greater than n_c, the channel decoder can recover the lost packets entirely. Therefore, the average number of lost packets in the protected part and in parity packets, l_{p+c}, can be found by (5.14):

$$l_{p+c} = \sum_{j=n_c+1}^{n_p+n_c} jP\left(j, n_p + n_c\right) \tag{5.14}$$

When **model 1** is used as the channel model, losses happen independently. Therefore,the portion of losses in the information part is equal to the code rate of the protected part:

$$l_p = \frac{n_p}{n_p + n_c} \sum_{j=n_c+1}^{n_p+n_c} jP\left(j, n_p + n_c\right). \tag{5.15}$$

Therefore when the channel model is **model 1**, and $P\left(.,.\right)$ is its corresponding loss distribution function, the residual packet loss probability can be computed using (5.16):

$$p_r = \frac{l_u + l_p}{n_u + n_p} = \frac{n_u}{n_u + n_p} p + \frac{1}{n_u + n_p} \frac{n_p}{n_p + n_c} \sum_{j=n_c+1}^{n_p+n_c} jP\left(j, n_p + n_c\right). \tag{5.16}$$

Figure 5.17 demonstrates the numerical results of the residual packet loss probability when subset selection without feedback is used for error protection. For a certain number of protected packets (n_p), the code provides the best results. And, Figure 5.18 demonstrates the numerical results of the achieved coding gain when subset selection without feedback is used for error protection. In this figure, the achieved coding gain is the ratio of the residual packet loss rate after using FEC and after using subset selection without feedback, therefore a larger value of this ratio means better error protection. For a certain number of protected packets (n_p) the code provides the best results.

When **model 2** is used as the channel model, losses do not happen independently anymore. To find the portion of losses in the information part, we first need to find the probability of three new events.

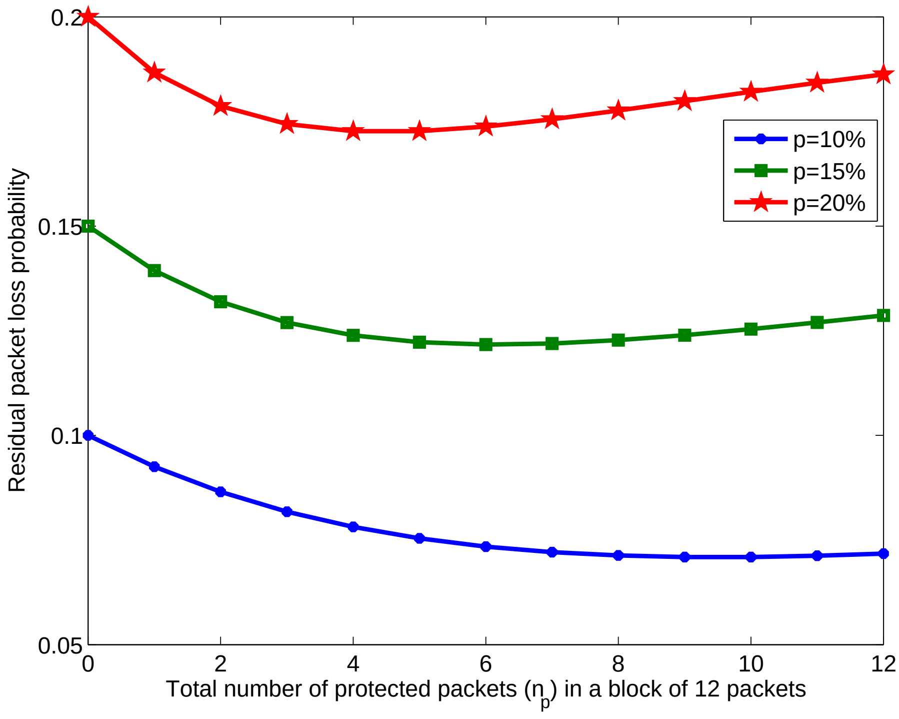


Figure 5.17: The residual packet loss probability in the *subset selection without feedback* scheme for channel **model 1**

- $\acute{g}(v) = P\left(0^{\nu-1}1 \mid 0\right)$ is the probability of $\nu - 1$ correctly received packets after one correctly received packet, and can be calculated using (5.17):

$$\acute{g}(v) = \begin{cases} p_{GB}, & \nu = 1, \\ (1 - p_{GB})^{\nu-1} p_{GB}, & \nu > 1. \end{cases} \tag{5.17}$$

- $\acute{\pi}(m, n)$ is the probability of $m - 1$ losses within the next $n - 1$ packets following a correctly received packet, and can be calculated using (5.18):

$$\acute{\pi}(m, n) = \begin{cases} (1 - p)(1 - p_{GB})^{n-1}, & m = 1, \\ \sum_{\nu=1}^{n-m+1} \acute{g}(\nu)\, \pi(m - 1, n - \nu), & 2 \leq m \leq n. \end{cases} \tag{5.18}$$

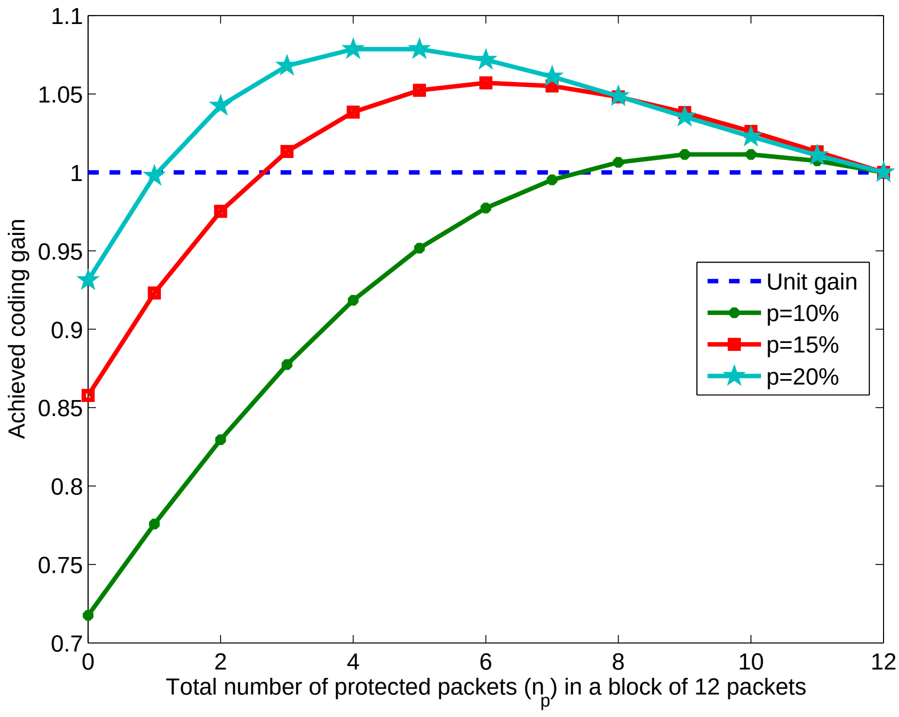


Figure 5.18: The achieved coding gain in the *subset selection without feedback* scheme for channel **model 1**

As Figure 5.19 shows, the probability of having m_c lost packets among n_c parity packets can be found using $\pi\left(m+1,n+1\right)$, if the last protected packet has been received correctly. Otherwise this probability can be found using $\acute{\pi}\left(m+1,n+1\right)$. Since the sequence of channel realizations in model 2 is an ergodic process[4], the probability of having a lost packet at the last position of protected packets is $\frac{m_p}{n_p}$, when m_p packets are lost among n_c protected packets. And, the probability of having a correctly received packet in that position is $1-\frac{m_p}{n_p}$. Thus, we can write the probability of having m_c losses among n_c parity packets as (5.19), when we know than m_p packets have been lost among n_p protected packets.

[4]It can be shown that a finite state irreducible Markov chain is ergodic if it has an aperiodic state [51] and the Gilbert model which is used in this thesis has this property

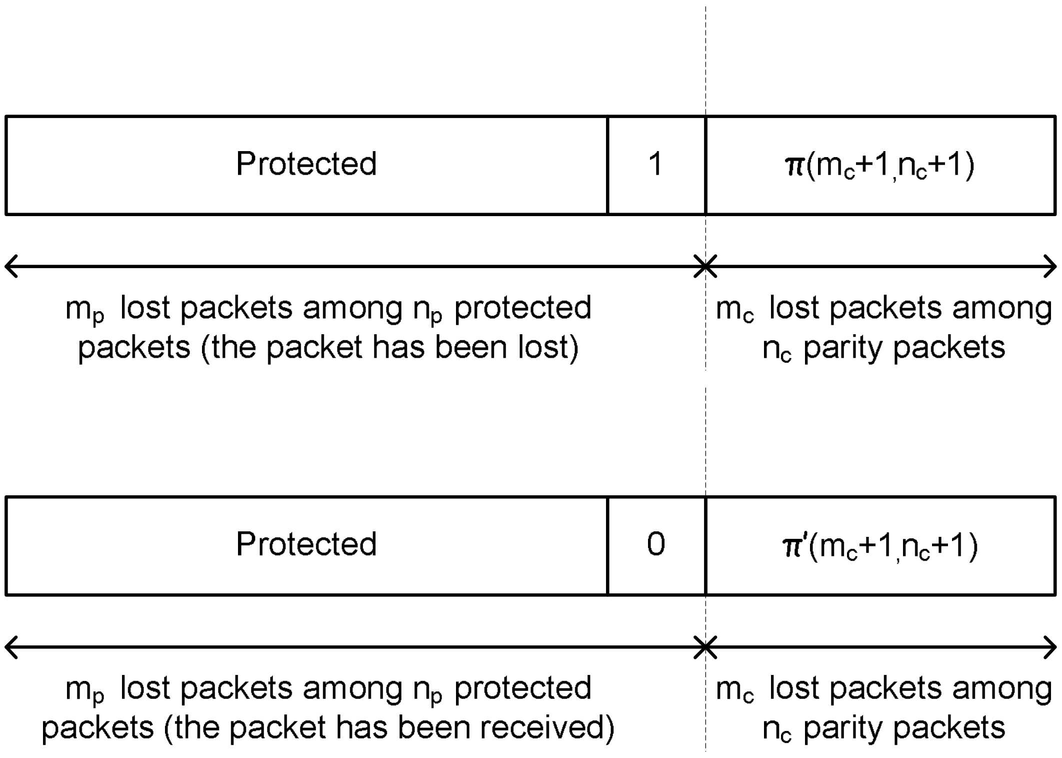

Protected
1
π(mc+1,nc+1)
mp lost packets among np protected packets (the packet has been lost)
mc lost packets among nc parity packets
Protected
0
π'(mc+1,nc+1)
mp lost packets among np protected packets (the packet has been received)
mc lost packets among nc parity packets

Figure 5.19: The probability of having m_c lost packets among n_c parity packets can be found using $\pi(m+1, n+1)$ when the last protected packed is received correctly (top image) or using $\acute{\pi}(m+1, n+1)$ when the last protected packed is lost (bottom image)

$$P(m_c, n_c, m_p, n_p) = \frac{m_p}{n_p}\pi(m_c+1, n_c+1) + \left(1 - \frac{m_p}{n_p}\right)\acute{\pi}(m_c+1, n_c+1). \tag{5.19}$$

Now, we can calculate the expected number of losses in the protected area of the code, $E[m_p \mid m]$, using (5.21):

$$P(m_p, n_c, n_p \mid m) = \frac{P(m_p, n_p)}{P(m, n_c+n_p)} P(m_c, n_c, m_p, n_p), \tag{5.20}$$

$$E[m_p \mid m] = \sum_{m_p=0}^{m} m_p P(m_p, n_c, n_p \mid m), \tag{5.21}$$

$$l_p = \sum_{j=n_c+1}^{n_p+n_c} E\left[m_p \mid j\right] P\left(j, n_p + n_c\right). \tag{5.22}$$

Therefore, after channel decoding, the average number of lost packets in the protected area can be determined using (5.22) and the total residual loss probability can be calculated using (5.23)

$$p_r = \frac{l_p + l_u}{n_u + n_p} \tag{5.23}$$

5.5.3 Subset Selection Effects on the Receivers Distortion without Feedback

When there is no feedback information, there is still a chance to use subset selection algorithm for increasing the quality of decoder's outputs. In such a situation, more important packets are more probable to be selected for the subset and consequently protected by a channel code. In this thesis, we will not analyze this case, because it is a very simple case of unequal error protection of a video bitstream, which has been studied in the literature [18, 30, 36].

5.6 Subset Selection in Type-II Hybrid ARQ/FEC for Video Multicast with Multiple subsets

The proposed subset selection algorithm can be modified to divide available bandwidth (parity packets) between more than one subset. The modified algorithm has a higher degree of freedom, therefore it may be able achieve better performance in controlling errors.

To illustrate this point, consider the situation in Figure 5.20, where each of the users has lost two packets (shaded in gray) out of the block of five packets. Suppose the bandwidth

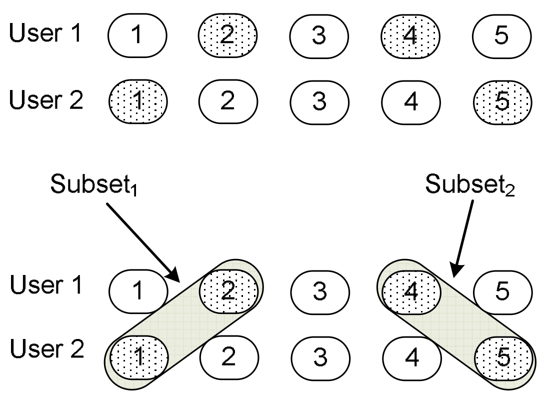


Figure 5.20: Motivation for subset selection in Type-II hybrid ARQ/FEC with multiple subsets.

limitation allows the server to multicast only two parity packets. In this situation subset selection with only one subset can not achieve any gain compare to plain type-II hybrid ARQ/FEC. Each decoder can recover two packets if and only if it receives both of the parity packets. And, if a decoder only receives one of the parity packets, it can not use it to recover lost packets.

However, if the server chooses two separate subsets, namely $subset_1 = \{Frame_1, Frame_2\}$ and $subset_2 = \{Frame_4, Frame_5\}$ as shown in Figure 5.20, then, computes one parity packet for each subset. Thus, a decoder is able to recover lost packets partially when it received only one parity packet.

If we assume that losses happen independently with probability of p,

- In the first scheme, when there is only one subset, each decoder only can recover two lost frames, if it correctly receives both parity packets (with probability of $(1-p)^2$). Otherwise, it can not recover any lost frame (with probability of $1-(1-p)^2$).

- In the second scheme, when there are two subsets, each decoder can recover two lost frames, if it correctly receives both parity packets (with probability of $(1-p)^2$). In addition, it can recover one frame, if it received only one of the parity packets (with probability of $2 * p\,(1-p)$).

Equation (5.24) shows the achievable gain in coding efficiency when the decoder uses two subsets in the the example of Figure 5.20 (when independent losses happen with probability of p):

$$Gain = \frac{2 * (1-p)^2 + 2 * p\,(1-p)}{2 * (1-p)^2} = \frac{1}{1-p} \tag{5.24}$$

Extending the proposed full search and simulated annealing optimization for the subset selection in subsections 5.3.1 and 5.3.2 to use more than one subset may be the future research direction based on the proposed subset selection in this chapter.

Chapter 6

Conclusion and Future Work

In this thesis, we focused on one of the most attractive problems in the new technology world. Nowadays, people have more demand for high-quality videos over the Internet and wireless networks. Our focus was on one important challenge of video delivery, which is combating errors that occur during video communication.

Combating errors can be accomplished in different parts of a video communication system. In this thesis, we divided these techniques into three different categories:

1. The encoder can increase the robustness of the encoded bitstream,

2. the transmission module can use effective channel codes for video transmission,

3. and the decoder can use various techniques to recover damaged frames.

We have analyzed all of these categories and for each of them we tried to propose new improvements. In chapter 2, we proposed a new structure for the H.264 encoder. The new proposed structure enables the encoder to increase the resilience of the encoded bitstream using fast reference picture selection (RPS). In chapter 3, the focus was on proposing a new error concealment algorithm for decoders. The new proposed algorithm uses the information available in the noncausal part of the bitstream to recover damaged frames in the received bitstream. And in chapter 5, we focused on the development of a new error control technique for video multicast.

The future works and improvements based on this thesis can be categorized as bellow:

1. The first stage of the proposed encoder structure needs to run offline, because it is responsible for motion estimation, which is the most time consuming part of video compression. Therefore the two-stage encoder can only be useful for pre-recorded video streaming scenarios, where the video contents are available before the transmission time. The next generation for this encoder can find a solution to run all the steps in real time and be able to stream live and interactive video contents. One suggestion to have such an encoder is using available motion vectors in the encoder to reduce search space for RPS reset frame's motion vectors.

   Also, in the second stage of the current version of the two-stage encoder, the encoder deals with raw video files which require a huge amount of space on the storage device. Using such large data storages makes the implementation of the video storage impractical and unjustified. To improve the two-stage encoder, one can store compressed video files and precomputed motion vectors on the streaming server. During the transmission time the server streams precompressed video packets. Whenever the server realizes that an invalid frame was used for encoding a subsequent frame, it switches to two-stage encoder for fast RPS. Since the two-stage encoder requires raw video, the streaming system needs to run the decoder to provide uncompressed version of this frame. This hybrid structure is illustrated in Figure 6.1

2. In the chapter 3, we proposed a noncausal error concealment algorithm, which uses the noncausal information hidden in frames after the damaged part of the bitstream. The proposed algorithm does not use the scene information such as conceptual shape models [31] for frame concealment. One improvement based on this part of the work can be done by using this information for frame recovery. The advantage of this

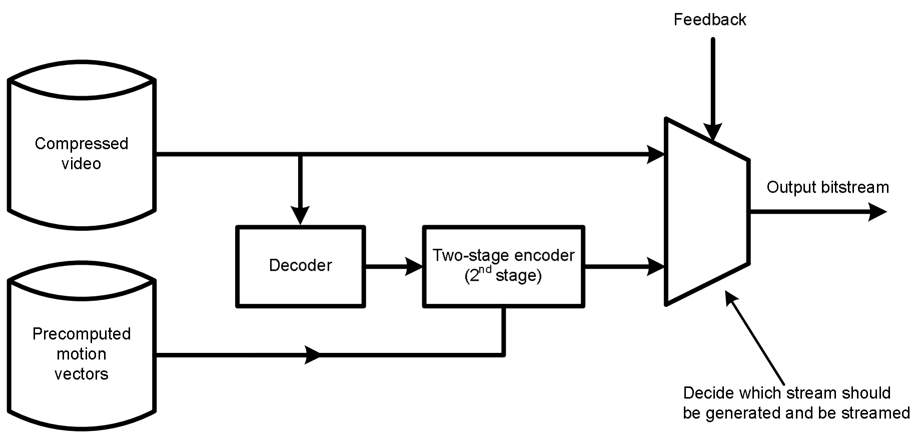

Feedback
Compressed video
Decoder
Two-stage encoder (2^{nd} stage)
Output bitstream
Precomputed motion vectors
Decide which stream should be generated and be streamed

Figure 6.1: Hybrid structure for streaming server. In default, this structure uses pre-compressed video stream. But whenever a frame uses an invalid reference it switched to two-stage encoder for fast RPS.

improvement is not only limited to achieving better PSNRs, but it may also be able to enhance the perceived subjective video quality for the person who is watching the received video.

3. In chapter 5, we proposed a subset selection algorithm for type-II hybrid ARQ/FEC. The proposed algorithm chooses a subset among lost frames to try to recover using Reed-Solomon code. The first improvement for this algorithm can be developing an extension for selecting multiple subsets. Nowadays, new video technologies motivate us to develop more effective tools for multi-view video transmission. The subset selection can be easily developed to fit in the requirement of multi-view video transmission.

Appendix A

Performance Modeling of the Two-stage Encoder

Since the value of the QP parameter may be different in the two encoding stages, the motion vectors obtained in the first stage may be different from the best motion vectors that could be obtained in the second stage of the two-stage encoder, if motion estimation were to be performed in the second stage. Therefore, to analyze the performance of the proposed encoder, first we need to know how different these motion vectors are.

A.1 Modeling of the motion vector estimation error

In motion compensated prediction we have

$$X = Y\left(\overrightarrow{MV_t}\right) + \epsilon,$$

where X is the current frame, Y is the reference frame, and ϵ is the motion-compensated prediction residual. In this notation, $\overrightarrow{MV_t}$ is used as the *true displacement* to indicate that we assume that the true motion between Y and X is translatory (i.e. displacement), and that is different from the estimated motion vector, $\overrightarrow{\widehat{MV}_Y}$.

Encoders usually use least mean square error (LMSE) estimator for motion vector estimation:

$$\overrightarrow{\widehat{MV_Y}} = arg\min_{\vec{\nu}} \sum \sum_{D} \left(X - Y\left(\vec{\nu}\right)\right)^2 .$$

By treating frames as 2-D continuous functions, we can formulate this estimation as bellow (D is the domain of X and Y):

$$\begin{aligned}\overrightarrow{\widehat{MV_Y}} &= arg\min_{\vec{\nu}} \int\int_D \left(X - Y\left(\vec{\nu}\right)\right)^2 dA \\ &= arg\min_{\vec{\nu}} \int\int_D \left(X^2 - 2XY\left(\vec{\nu}\right) + Y^2\left(\vec{\nu}\right)\right) dA.\end{aligned} \tag{A.1}$$

X^2 does not depend on $\vec{\nu}$, therefore it has no effect on the minimization with respect to $\vec{\nu}$. $Y^2\left(\vec{\nu}\right)$ does depend on $\vec{\nu}$. But, if the MV field $\vec{\nu}$ is reasonably smooth, then $\int\int_D Y^2\left(\vec{\nu}\right) dA \approx$ *const.*, which means $\int\int_D Y^2\left(\vec{\nu}\right) dA$ does not depend on $\vec{\nu}$ very much. Hence, (A.1) can be reduced to a simpler form. When $\Lambda\left(\vec{\nu}\right)$ is defined as $\int\int_D \left(X.Y\left(\vec{\nu}\right)\right) dA$ (the correlation of X and Y), then $\Lambda\left(\vec{\nu}\right)$ can be used for motion vector estimation.

$$\begin{aligned}\overrightarrow{\widehat{MV_Y}} &= arg\min_{\vec{\nu}} \int\int_D \left(X - Y\left(\vec{\nu}\right)\right)^2 dA \\ &= arg\min_{\vec{\nu}} \int\int_D \left(X^2 - 2XY\left(\vec{\nu}\right) + Y^2\left(\vec{\nu}\right) dA\right) \\ &\approx arg\min_{\vec{\nu}} \int\int_D -2XY\left(\vec{\nu}\right) dA \\ &= arg\max_{\vec{\nu}} \int\int XY\left(\vec{\nu}\right) dA = arg\max_{\vec{\nu}} \Lambda\left(\vec{\nu}\right).\end{aligned} \tag{A.2}$$

We can further rewrite correlation as follows:

$$\begin{aligned}\overrightarrow{\widehat{MV_Y}} &= arg\max_{\vec{\nu}} \Lambda\left(\vec{\nu}\right) \\ &= arg\max_{\vec{\nu}} \int\int_D \left(X.Y\left(\vec{\nu}\right)\right) dA \\ &= arg\max_{\vec{\nu}} \int\int_D \left[\left(Y\left(\overrightarrow{MV_t}\right) + \epsilon\right).Y\left(\vec{\nu}\right)\right] dA \\ &= arg\max_{\vec{\nu}} \left[\int\int_D Y\left(\overrightarrow{MV_t}\right).Y\left(\vec{\nu}\right) dA \right. \\ &\quad \left. + \int\int_D \epsilon.Y\left(\vec{\nu}\right) dA\right].\end{aligned} \tag{A.3}$$

If the motion estimation is reasonably accurate then it is supposed to estimate the value of $\widehat{MV}_Y$ very close to the $\overrightarrow{MV_t}$. Also, the cross-correlation of two continuous functions with limited energies is always bounded and has a maximum [45]. And, being a continuous function, cross-correlation is always concave around its peak [24]. Therefore, a necessary condition for maximum of $\Lambda(\vec{\nu})$ is that its gradient vanishes:

$$\begin{aligned}\nabla\Lambda\left(\overrightarrow{\widehat{MV}_Y}\right) &= \nabla\underbrace{\int\int_D Y\left(\overrightarrow{MV_t}\right).Y\left(\overrightarrow{\widehat{MV}_Y}\right)dA}_{C_{YY}\left(\overrightarrow{\widehat{MV}_Y}-\overrightarrow{MV_t}\right)} \\ &\quad+\underbrace{\nabla\int\int_D \epsilon.Y\left(\overrightarrow{\widehat{MV}_Y}\right)dA}_{\vec{\xi}} \\ &= \nabla C_{YY}\left(\overrightarrow{\widehat{MV}_Y}-\overrightarrow{MV_t}\right)+\vec{\xi}=\vec{0}.\end{aligned} \tag{A.4}$$

It should be noted that gradient operators ∇ in (A.4) are gradients with respect to horizontal and vertical components of $\overrightarrow{\widehat{MV}_Y}$. $Y\left(\overrightarrow{MV_t}\right)$ is constant with respect to these parameters, so we can write

$$\begin{aligned}\nabla C_{YY}\left(\overrightarrow{\widehat{MV}_Y}-\overrightarrow{MV_t}\right) &= \nabla\left[\int\int_D Y\left(\overrightarrow{MV_t}\right).Y\left(\overrightarrow{\widehat{MV}_Y}\right)dA\right] \\ &= \int\int_D \underbrace{\nabla Y\left(\overrightarrow{MV_t}\right)}_{\text{equal to 0}}.Y\left(\overrightarrow{\widehat{MV}_Y}\right)dA+\int\int_D Y\left(\overrightarrow{MV_t}\right).\nabla Y\left(\overrightarrow{\widehat{MV}_Y}\right)dA \\ &= \int\int_D Y\left(\overrightarrow{MV_t}\right).\nabla Y\left(\overrightarrow{\widehat{MV}_Y}\right)dA.\end{aligned} \tag{A.5}$$

Since finding this gradient is not straightforward, we use (A.6) and (A.7) to find the gradient in terms of spatial gradient of frame Y to rewrite (A.4) as (A.9).

$$\begin{aligned}
\frac{\partial Y\left(x-\overrightarrow{\widehat{MV}_{Y,x}},y-\overrightarrow{\widehat{MV}_{Y,y}}\right)}{\partial\overrightarrow{\widehat{MV}_{Y,x}}} &= \lim_{d_{mvx}\to 0}\frac{Y\left(x-\left(\overrightarrow{\widehat{MV}_{Y,x}}+d_{mvx}\right),y-\overrightarrow{\widehat{MV}_{Y,y}}\right)}{d_{mvx}} \\
&= \lim_{d_{mvx}\to 0}\frac{Y\left((x-d_{mvx})-\overrightarrow{\widehat{MV}_{Y,x}},y-\overrightarrow{\widehat{MV}_{Y,y}}\right)}{d_{mvx}} \\
&= \lim_{d_{mvx}\to 0}\frac{Y\left((x+d_{mvx})-\overrightarrow{\widehat{MV}_{Y,x}},y-\overrightarrow{\widehat{MV}_{Y,y}}\right)}{-d_{mvx}} \\
&= -\frac{\partial Y\left(x-\overrightarrow{\widehat{MV}_{Y,x}},y-\overrightarrow{\widehat{MV}_{Y,y}}\right)}{\partial x}
\end{aligned} \tag{A.6}$$

$$\begin{aligned}
\frac{\partial Y\left(x-\overrightarrow{\widehat{MV}_{Y,x}},y-\overrightarrow{\widehat{MV}_{Y,y}}\right)}{\partial\overrightarrow{\widehat{MV}_{Y,y}}} &= \lim_{d_{mvy}\to 0}\frac{Y\left(x-\overrightarrow{\widehat{MV}_{Y,x}},y-\left(\overrightarrow{\widehat{MV}_{Y,y}}+d_{mvy}\right)\right)}{d_{mvy}} \\
&= \lim_{d_{mvy}\to 0}\frac{Y\left(x-\overrightarrow{\widehat{MV}_{Y,x}},(y-d_{mvy})-\overrightarrow{\widehat{MV}_{Y,y}}\right)}{d_{mvx}} \\
&= \lim_{d_{mvy}\to 0}\frac{Y\left(x-\overrightarrow{\widehat{MV}_{Y,x}},(y+d_{mvy})-\overrightarrow{\widehat{MV}_{Y,y}}\right)}{-d_{mvx}} \\
&= -\frac{\partial Y\left(x-\overrightarrow{\widehat{MV}_{Y,x}},y-\overrightarrow{\widehat{MV}_{Y,y}}\right)}{\partial y}
\end{aligned} \tag{A.7}$$

Hence,

$$-\nabla C_{YY}\left(\vec{e}_{mv,Y}\right)+\vec{\xi}=\vec{0}, \tag{A.8}$$

or

$$\nabla C_{YY}\left(\vec{e}_{mv,Y}\right)-\vec{\xi}=\vec{0}. \tag{A.9}$$

The general analytical solution for (A.9) is not easily obtained. Therefore, we use the linear approximation, (A.10), of $\nabla C_{YY}\left(\vec{e}_{mv,Y}\right)$ to solve (A.9):

$$\begin{aligned}\nabla C_{YY}\left(\vec{e}_{mv,Y}\right) &\approx G\left(\vec{e}_{mv,Y}\right).\vec{i}+H\left(\vec{e}_{mv,Y}\right).\vec{j} \\ &= \begin{bmatrix} H_x\left(\vec{(0)}\right) & H_y\left(\vec{(0)}\right) \\ G_x\left(\vec{(0)}\right) & G_y\left(\vec{(0)}\right) \end{bmatrix}\begin{bmatrix} e_{mx} \\ e_{my}\end{bmatrix} = \Omega.\begin{bmatrix} e_{mx} \\ e_{my}\end{bmatrix}. \end{aligned} \tag{A.10}$$

If Ω is an invertible matrix we can find $\vec{e}_{mv,Y}$ as (A.11), otherwise there are multiple solutions to (A.3). This situation can happen in smooth areas without strong texture.

$$\vec{e}_{mv,Y} = \begin{bmatrix} e_{mx} \\ e_{my}\end{bmatrix} = \Omega^{-1}.\vec{\xi}. \tag{A.11}$$

In this text, we use a Gaussian distribution as an approximation for the distribution of the residual,ϵ, to keep our model tractable. When the residual ϵ has a Gaussian distribution with variance of σ^2 and a zero mean, $\vec{\xi} = \int\int_D \epsilon.\nabla Y\left(\overrightarrow{\widehat{MV_Y}}\right) dA$ has a Gaussian distribution too. Therefore, $\vec{e}_{mv,Y} = \overrightarrow{\widehat{MV_Y}} - \overrightarrow{MV_t}$ has a Gaussian distribution as well. Therefore, we can find the covariance matrices for $\vec{\xi}$ and $\vec{e}_{mv,Y}$ as bellow:

$$\Sigma_{\vec{\xi}} = \sigma_\epsilon^2 \begin{bmatrix} \left(\frac{\partial Y}{\partial x}\right)^2 & \frac{\partial Y}{\partial x}\cdot\frac{\partial Y}{\partial y} \\ \frac{\partial Y}{\partial x}\cdot\frac{\partial Y}{\partial y} & \left(\frac{\partial Y}{\partial y}\right)^2 \end{bmatrix}, \tag{A.12}$$

$$\Sigma_{\vec{e}_{mv,Y}} = \Omega^{-1}\Sigma_\xi {\Omega^{-1}}^T. \tag{A.13}$$

A.2 Modeling of the difference between two motion vector fields

Equations (A.12) and (A.13) show the covariance matrix for the diffference between the estimated motion vector field and the *true motion vector* field. But the interesting parameter for us is finding the probability density function of the difference between the estimated motion vector fields using $\hat{Y}_1$ and $\hat{Y}_2$, where $X = \hat{Y}_1\left(\overrightarrow{\widehat{MV_{\hat{Y}_1}}}\right)+\epsilon_1$ and $X = \hat{Y}_2\left(\overrightarrow{\widehat{MV_{\hat{Y}_2}}}\right)+\epsilon_2$.

One can calculate the difference ($\Delta \vec{mv}_{\hat{Y}_1,\hat{Y}_2}$) between the estimated motion vector fields using $\hat{Y}_1$ and $\hat{Y}_2$ (Figure 2.3) by (A.14):

$$\begin{aligned}
\Delta \vec{mv}_{\hat{Y}_1,\hat{Y}_2} &= \overrightarrow{\widehat{MV}_{\hat{Y}_1}} - \overrightarrow{\widehat{MV}_{\hat{Y}_2}} \\
&= \overrightarrow{\widehat{MV}_{\hat{Y}_1}} - \overrightarrow{MV_t} + \overrightarrow{MV_t} - \overrightarrow{\widehat{MV}_{\hat{Y}_2}} \\
&= \vec{e}_{mv,\hat{Y}_1} - \vec{e}_{mv,\hat{Y}_2} \\
&= \Omega_{\hat{Y}_1}^{-1} \vec{\xi}_{\hat{Y}_1} - \Omega_{\hat{Y}_2}^{-1} \vec{\xi}_{\hat{Y}_2}. \qquad (A.14)
\end{aligned}$$

If $\hat{Y}_1$ and $\hat{Y}_2$ represent the same reference frame quantized by two different quantizers, as in our case, then we expect $\Omega_{\hat{Y}_1} \approx \Omega_{\hat{Y}_2} \approx \Omega_{\hat{Y}}$, and $\epsilon_1 \approx \epsilon_2 \approx \epsilon$. Hence, (A.14) can be approximated by (A.15):

$$\begin{aligned}
\Delta \vec{mv}_{\hat{Y}_1,\hat{Y}_2} &= \Omega_{\hat{Y}_1}^{-1} \vec{\xi}_{\hat{Y}_1} - \Omega_{\hat{Y}_2}^{-1} \vec{\xi}_{\hat{Y}_2} \\
&= \Omega_{\hat{Y}}^{-1} \left(\vec{\xi}_{\hat{Y}_1} - \vec{\xi}_{\hat{Y}_2} \right) \\
&= \Omega_{\hat{Y}}^{-1} \left[\int\int_D \epsilon_1 . \nabla \hat{Y}_1 \left(\overrightarrow{\widehat{MV}_{\hat{Y}_1}} \right) dA \right. \\
&\quad \left. - \int\int_D \epsilon_2 . \nabla \hat{Y}_2 \left(\overrightarrow{\widehat{MV}_{\hat{Y}_2}} \right) dA \right] \\
&\approx \Omega_{\hat{Y}}^{-1} \int\int_D \epsilon . \left[\nabla \underbrace{\left(\hat{Y}_1 \left(\overrightarrow{\widehat{MV}_{\hat{Y}_1}} \right) - \hat{Y}_2 \left(\overrightarrow{\widehat{MV}_{\hat{Y}_2}} \right) \right)}_{\hat{N}_{1,2}} \right] dA \\
&= \Omega_{\hat{Y}}^{-1} \underbrace{\int\int_D \epsilon . \nabla \hat{N}_{1,2} . dA}_{\psi_0}
\end{aligned} \qquad (A.15)$$

The top linear space invariance (LSI) system in Figure A.1 generates $\psi(x, y) = \epsilon * \left(h_\nabla * \hat{N}_{1,2}(x, y) \right)$ where the input is $\hat{N}_{1,2}(x, y)$. In this situation, the intermediate signal before entering the LSI system with impulse response ϵ, is $\nabla \hat{N}_{1,2}(x, y)$. So based on the definition of ψ_0, the output value at point $(0, 0)$ is $\psi(0, 0) = \psi_0$. On the other hand, because

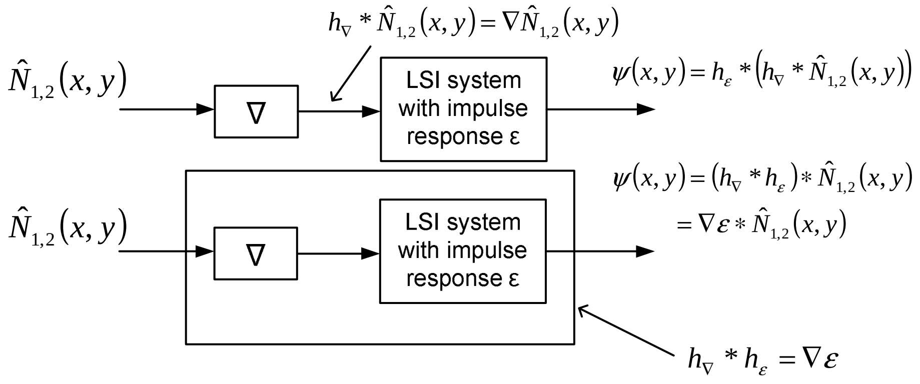

$h_\nabla * \hat{N}_{1,2}(x,y) = \nabla\hat{N}_{1,2}(x,y)$
$\hat{N}_{1,2}(x,y)$
∇
LSI system with impulse response ε
$\psi(x,y) = h_\varepsilon * (h_\nabla * \hat{N}_{1,2}(x,y))$
$\hat{N}_{1,2}(x,y)$
∇
LSI system with impulse response ε
$\psi(x,y) = (h_\nabla * h_\varepsilon) * \hat{N}_{1,2}(x,y) = \nabla\varepsilon * \hat{N}_{1,2}(x,y)$
$h_\nabla * h_\varepsilon = \nabla\varepsilon$

Figure A.1: Two equivalent linear space invariant (LSI) systems that can produce $\psi(x,y)$

this system is a LSI system we can combine two internal LSI systems (as shown at the bottom of Figure A.1) and then convolve it with $\hat{N}_{1,2}(x,y)$. In this case, the impulse response of the combined system is $\epsilon * h_\nabla = \nabla\epsilon$, therefore we can find the value of $\psi_0 = \psi(0,0)$ using equation (A.16). Then, the covariance matrix for ψ_0 and $\Delta\vec{mv}_{\hat{Y}_1,\hat{Y}_2}$ can be calculated by ((A.17)) and ((A.18)).

$$\psi_0 = \int\int_D \epsilon.\nabla\hat{N}_{1,2}.dA = \int\int_D \nabla\epsilon.\hat{N}_{1,2}.dA \tag{A.16}$$

$$\Sigma_{\psi_0} = \sigma^2_{\hat{N}_{1,2}} \int\int_D \nabla\epsilon\,(x,y)\,(\nabla\epsilon\,(x,y))^T\,dxdy \tag{A.17}$$

$$\Sigma_{\Delta\vec{mv}_{\hat{Y}_1,\hat{Y}_2}} = \Omega_{\hat{Y}}^{-1}\Sigma_{\psi_0}\Omega_{\hat{Y}}^{-1^T} \tag{A.18}$$

(A.18) does not directly connect the $\Sigma_{\Delta\vec{mv}_{\hat{Y}_1,\hat{Y}_2}}$ to the coding parameters, such as quantization parameters (QP) or encoding rate. Since $\Sigma_{\Delta\vec{mv}_{\hat{Y}_1,\hat{Y}_2}}$ is a function of $\Omega_{\hat{Y}}$ and Σ_{ψ_0}, to use (A.18) we need to use $\Omega_{\hat{Y}}$ and Σ_{ψ_0}.

To find $\Omega_{\hat{Y}}$, we can calculate the autocorrelation of the reconstructed reference in the encoder buffer, find its gradient, then use (A.10) to find $\Omega_{\hat{Y}}$.

To find Σ_{ψ_0}, we need to calculate

- $\sigma^2_{\hat{N}_{1,2}}$: for calculation of $\hat{N}_{1,2}$, we need to subtract to motion compensated predictions of the current frame using the first set of motion vectors ($\overrightarrow{\widehat{MV}_{\hat{Y}_1}}$) and reference frame

$(\hat{Y}_1)$, and the second set of motion vector $\overrightarrow{\widehat{MV}_{\hat{Y}_2}}$ and reference frame $(\hat{Y}_2)$. These can be generated using two different quantization parameter(QP) in the encoder. Then, we can compute the variance of $\hat{N}_{1,2}$

- $\nabla\epsilon\,(x,y)$: ϵ is the residual of the prediction in the encoder.

then plug these values into (A.17).

Appendix B

User Guide

This appendix provides a user guide for the enclosed DVD which contain developed codes during this thesis and simulation scripts to regenerate the results for future developments. Section B.1 briefly describes different files on the DVD.

B.1 Files on the DVD

The enclosed DVD contains several folders and files. In this section, we briefly describe them.

- **data**
    - ∗ **ErrorPatternsForInternetExperiments**: This folder measured packet traces from [60]
    - ∗ **RawVideoSequences**: This folder contains raw video sequences which have been used for experimental results in the thesis. There are also two applications which can be useful when someone want to work with these sequences. *seqview.exe* is a very handy player for YUV file, and *yuv2avi.exe* is a handy converter to convert YUV files to AVI files

- **SourceCodes**
  - ∗ **jm16.0.zip** : *JM16.0* is the latest version of H.264/AVC reference software.
  - ∗ **jm12.4** *JM12.4* is the version of H.264/AVC reference software, which has been used for development of the proposed algorithm in this thesis.
  - ∗ **JM2Stage** This folder contains the implementation of the proposed two-stage encoder. This folder contains both JM encoder and JM decoder, but the only modified part is the encoder to form two-stage encoder. A file named *loss.txt* must be present in the folder of the executable file. *loss.txt* helps the encoder to simulate feedback messages from the receiver.
  - ∗ **NoncausalErrorConcealment** The proposed noncausal error concealment algorithm is implemented in two separate codes. To run the code
    - ⋄ we first need to run the JM decoder to extract motion vector information of the bitstream, then run the C++ code in *offlineConcealment* folder to recover the motion vectors which belong to the lost frame (this part of code can perform the algorithm from [14] and the proposed noncausal error concealment algorithm in chapter 3).
    - ⋄ Then we need to run the modified JM decoder in folder *JMNCDecoder*.
  - ∗ **SubsetSelection**
    - ⋄ **Simulator** This folder contains required C++ codes for simulating the proposed subset selection algorithm, the algorithm from [35], ARQ, and Reed-Solomon codes.
    - ⋄ **matlab** This folder contains required Matlab codes for analyzing different coding schemes and regenerating the analytical results in section 5.5.
  - ∗ **PacketDropper** In many of the simulation, we need to simulate the effect of the channel on the bitstream. The *packetDropper* is a simple program that performs

this task for us. *packetDropper* assumes that the bitstream contains RTP packets and removes the lost packets from the bitstream. There are three ways to use *packetDropper*.

1. `packetdropper.exe`

   In this mode, *packetDropper* uses default parameters. `test.264` and the input file, `out.264` as the output file, and `loss.txt` for loss drop pattern file.

2. `packetdropper.exe input-file output-file drop-pattern`: In this mode, the *packetDropper* used the user-defined file names.

3. `packetdropper.exe input-file output-file drop-pattern` ···

   ··· `FPS buffer-length` In this mode, the *packetDropper* used the user-defined file names. In addition to the previous mode, in this mode the *packetDropper* consider a buffer for the decoder, and drop packets which are received to late.

- **Simulation**

  ∗ **readme.xls**: this excel file describe the content and tasks of the folder here.

  ∗ **sequences**: The raw video sequences have been used for the simulation. this folder should not be changed or moved, because some the simulations need to use the files here.

  ∗ **STEP1**: Runs the first stage for two-stage encoder to store all motion vectors on the disk. By running `batch1.m`, it generate `*.smv` files. In this script, the encoder uses 8 reference frames for encoding, and stores motion vector for different conditions (refer to `readme.xls`).

  ∗ **STEP2**: Generates all error patterns. By running `allSTLossPatterns.m`, it will generate all loss pattern and to store them in `./stdloss`. These loss pattern will be used by other scripts.

* **STEP3**: Encodes sequences using JM encoder and two-stage encoder and then drops lost packets and sorts the results in `./std264`. By running `batch3.m`, the encoder uses all the loss patterns in `stdlosses` to generate h.264 files.
* **STEP4**: Measures the performance of the proposed noncausal whole-frame concealment algorithm. By running `batch4.m`, the script decode different compressed video sequences and measures their resulted PSNR, where different error concealment algorithms have been used.
* **STEP5**: Simulates systems A, B, C, and D which are described in chapter 4. To run this test, one should first runs `batch5.m` to simulates the systems, then runs `extract-snr.m` to extract the results.
* **SimulationSubsetSelection**: Runs an error protection algorithm for video multicast. To change the error protection, one only needs to change the value of `sMethod` in `test.m` and run `test.m`
* **MotionAccuracy:** This simulation can be used to validate the proposed rate-distortion model for the two stage encoder. To reproduce the results for motion accuracy and the required data for drawing Figure 2.4, one can use `test1.m`. To reproduce the results for rate penalty, one can use `extrabit.m` to obtain the results illustrated in Figure 2.5.

Bibliography

[1] Traffic details for youtube from alexa. http://www.alexa.com/siteinfo/youtube.com.

[2] Google to acquire youtube for 1.65 billion dollar in stock. Google Press Center, October 9, 2006.

[3] I. Ahmad, W. Zheng, J. Luo, and M. Liou. A fast adaptive motion estimation algorithm. *IEEE Transactions on Circuits and Systems for Video Technology*, 16(3):420–438, 2006.

[4] S. M. Amiri and I. V. Bajić. Subset selection in type-ii hybrid arq/fec for video multicast. In *IEEE International Conference on Communications (ICC 2009)*, June 2009.

[5] S.M. Amiri and I.V. Bajić. A two-stage H.264/AVC encoder for video streaming with fast reference picture selection. In *Proceedings of the 4th ACM workshop on Wireless multimedia networking and performance modeling*, pages 37–44. Vancovuer, BC, Canada, 2008.

[6] S.M. Amiri and I.V. Bajic. A novel noncausal whole-frame concealment algorithm for video streaming. In *IEEE International Symposium on Multimedia (ISM 2008)*, pages 154–159. IEEE Computer Society, Berkeley, CA, USA, 2008.

[7] J.G Apostolopoulos and M.D Trott. Path diversity for enhanced media streaming. *IEEE Communications Magazine*, 42(8):80–87, 2004.

[8] P. Baccichet, D. Bagni, A. Chimienti, L. Pezzoni, and F.S Rovati. Frame concealment for H.264/AVC decoders. *IEEE Transactions on Consumer Electronics*, 51(1):227–233, 2005.

[9] P. Baccichet, S. Rane, A. Chimienti, and B. Girod. Robust low-delay video transmission using H.264/AVC redundant slices and flexible macroblock ordering. In *IEEE International Conference on Image Processing (ICIP2007)*, volume 4, 2007.

[10] I.V. Bajić. Efficient cross-layer error control for wireless video multicast. *IEEE Transactions on Broadcasting*, 53(1):276–285, 2007.

[11] I.V. Bajić. *Error control for broadcasting and multicasting: An overview.* in Mobile Multimedia Broadcasting Standards: Technology and Practice, (F.-L. Luo, ed.). Springer, 2008.

[12] I.V. Bajić. Noncausal error control for video streaming over wireless packet networks. *IEEE Transactions on Multimedia*, 8(6):1263–1273, Dec. 2006.

[13] I.V. Bajić and J.W. Woods. Error concealment for scalable motion-compensated subband/wavelet video coders. *IEEE Transactions on Circuits and Systems for Video Technology*, 17(4):508–514, 2007.

[14] S. Belfiore, M. Grangetto, E. Magli, and G. Olmo. Concealment of whole-frame losses for wireless low bit-rate video based on multiframe optical flow estimation. *IEEE Transactions on Multimedia*, 7(2):316–329, 2005.

[15] V. Bhaskaran and K. Konstantinides. *Image and video compression standards: algorithms and architectures.* Kluwer academic publishers, 1997.

[16] H. Chen, Z. Han, R. Hu, and R. Ruan. Adaptive FMO selection strategy for error resilient H.264 coding. In *International Conference on Audio, Language and Image Processing (ICALIP2008)*, pages 868–872, 2008.

[17] Y. Chen, Y. Hu, O.C. Au, H. Li, and C.W. Chen. Video error concealment using spatio-temporal boundary matching and partial differential equation. *IEEE Transactions on Multimedia*, 10(1):2–15, 2008.

[18] L. Cheng, W. Zhang, and L. Chen. Rate-distortion optimized unequal loss protection for fgs compressed video. *IEEE Transactions on Broadcasting*, 50(2):126–131, 2004.

[19] C.H. Cheung and L.M. Po. Novel cross-diamond-hexagonal search algorithms for fast block motion estimation. *IEEE Transactions on Multimedia*, 7(1):16–22, 2005.

[20] P.A. Chou and Z. Miao. Rate-distortion optimized streaming of packetized media. *IEEE Transactions on Multimedia*, 8(2):390–404, 2006.

[21] R. Duncan. A survey of parallel computer architectures. *Computer*, 23(2):5–16, 1990.

[22] P. Elias. The noisy channel coding theorem for erasure channels. *The American Mathematical Monthly*, 81(8):853–862, 1974.

[23] E.O. Elliot. A model of the switched telephone network for data communications. *Bell Systems Technical Journal*, 44:89–109, 1965.

[24] G.B. Folland. *Real Analysis: Modern Techniques and Their Applications.* Pure and Applied Mathematics: A Wiley-Interscience Series of Texts, Monographs and Tracts, 1999.

[25] A. Ganjam and H. Zhang. Internet multicast video delivery. *Proceedings of the IEEE*, 93(1):159–170, 2005.

[26] H. Gharavi and S. Gao. Spatial interpolation algorithm for error concealment. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP2008)*, pages 1153–1156, 2008.

[27] B. Girod. Motion-compensating prediction with fractional-pel accuracy. *IEEE Transactions on Communications*, 53(3):1053–1060, 1993.

[28] B. Girod. Efficiency analysis of multihypothesis motion-compensated prediction for video coding. *IEEE Transactions on Image Processing*, 9(2):173–183, 2000.

[29] B. Girod and N. Farber. Feedback-based error control for mobile video transmission. *Proceedings of the IEEE*, 87(10):1707–1723, 1999.

[30] J. Goshi, A.E. Mohr, R.E. Ladner, E.A. Riskin, and A. Lippman. Unequal loss protection for H.263 compressed video. *IEEE Transactions on Circuits and Systems for Video Technology*, 15(3):412–419, 2005.

[31] A. Hampapur, L. Brown, J. Connell, A. Ekin, N. Haas, M. Lu, H. Merkl, and S. Pankanti. Smart video surveillance: exploring the concept of multiscale spatiotemporal tracking. *IEEE Signal Processing Magazine*, 22(2):38–51, 2005.

[32] E. Hladka, P. Holub, and J. Denemark. User empowered virtual multicast for multimedia communication. In *International Conference on Networking (ICN2004)*, 2004.

[33] T. Ho and D. Lun. *Network coding: an introduction.* Cambridge University Press, 2008.

[34] S. Huang and S. Kuo. Temporal error concealment for H.264/AVC using optimum regression plane. *Lecture Notes in Computer Science*, 4903:402, 2008.

[35] M. Kalman and B. Girod. *Network-Adaptive Media Transport.* Multimedia Over IP and Wireless Networks (P. A. Chou and M. van der Schaar, Eds.). Academic Press, 2007.

[36] J. Kim, R.M. Mersereau, and Y. Altunbasak. Distributed video streaming using multiple description coding and unequal error protection. *IEEE Transactions on Image Processing*, 14(7):849–861, 2005.

[37] S. Kirkpatrick. Optimization by simulated annealing: Quantitative studies. *Journal of Statistical Physics*, 34(5):975–986, 1984.

[38] S. Kumar, L. Xu, M.K. Mandal, and S. Panchanathan. Error resiliency schemes in H.264/AVC standard. *Journal of Visual Communication and Image Representation*, 17(2):425–450, 2006.

[39] D.X. Li, W. Zheng, and M. Zhang. Architecture design for H.264/AVC integer motion estimation with minimum memory bandwidth. *IEEE Transactions on Consumer Electronics*, 53(3):1053–1060, 2007.

[40] Y.J. Liang, B. Girod, and Q.C. Technol. Network-adaptive low-latency video communication over best-effort networks. *IEEE Transactions on Circuits and Systems for Video Technology*, 16(1):72–81, 2006.

[41] S. Lin and D.J. Costello. *Error Control Coding: Fundamentals and Applications.* Prentice Hall, 1983.

[42] J.D.C. Little. A proof of the queuing formula l= w. *Operations Research*, 9(3):383–387, 1961.

[43] N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, and E. Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087, 1953.

[44] P. Noll and N.S Jayant. *Digital Coding of Waveforms: Principles and Applications to Speech and Video.* Prentice Hall, Englewood Clis, New Jersey, 1984.

[45] A. Papoulis and S.U. Pillai. *Probability, random variables, and stochastic processes, 4th Edition.* McGraw-Hill New York, 1991.

[46] D. Persson, T. Eriksson, and P. Hedelin. Packet video error concealment with gaussian mixture models. *IEEE Transactions on Image Processing*, 17(2):145–154, 2008.

[47] I.E.G. Richardson. *H.264 and MPEG-4 video compression: video coding for next-generation multimedia.* Wiley, 2003.

[48] E. Setton, P. Baccichet, and B. Girod. Peer-to-peer live multicast: a video perspective. *Proceedings of the IEEE*, 96:25–38, 2008.

[49] E. Setton and B. Girod. *Peer-to-Peer Video Streaming.* Springer, 2007.

[50] E. Setton, J. Noh, and B. Girod. Rate-distortion optimized video peer-to-peer multicast streaming. In *ACM workshop on Advances in peer-to-peer multimedia streaming*, pages 39–48. ACM New York, NY, USA, 2005.

[51] K.S. Shanmugam. *Digital and analog communication systems.* Chichester, 1985.

[52] C.E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 1954.

[53] W. Stallings. *Data and computer communications, 8th Edition.* Prentice hall, 2007.

[54] T. Stockhammer, M.M. Hannuksela, and T. Wiegand. H.264/AVC in wireless environments. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(7):657–673, 2003.

[55] W. Tan and A. Zakhor. Video multicast using layered fec and scalable compression. *IEEE Transactions on Circuits and Systems for Video Technology*, 11(3):373–386, 2001.

[56] Joint Video Team (JVT) of ISO/IEC 14496-10 and ITU-T VCEG. Draft ITU-T recommendation and final draft international standard of joint video specification (ITU-T rec. H.264— ISO/IEC 14496-10 AVC). Technical report, JVT-G050r1, Geneva, Switzerland, 2003.

[57] A. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269, 1967.

[58] Y. Wang, Y. Zhang, and J. Ostermann. *Video Processing and Communications*. Prentice Hall PTR Upper Saddle River, NJ, USA, 2001.

[59] Y.K. Wang, M.M. Hannuksela, V. Varsa, A. Hourunranta, and M. Gabbouj. The error concealment feature in the H.264/AVC test model. In *IEEE International Conference on Image Processing (ICIP2002)*, 2002.

[60] S. Wenger. Proposed error patterns for Internet experiments. ITU-T *Study Group 16* H.263+ V*ideo* E*xperts* G*roup*, 15, 1999.

[61] S. Wenger, A.G. Teles, and G. Berlin. H.264/AVC over IP. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(7):645–656, 2003.

[62] T. Wiegand, H. Schwarz, A. Joch, F. Kossentini, and G.J. Sullivan. Rate-constrained coder control and comparison of video coding standards. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(7):688–703, 2003.

[63] T. Wiegand, G.J. Sullivan, G. Bjontegaard, and A. Luthra. Overview of the h. 264/AVC video coding standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(7):560–576, 2003.

[64] D. Xu, M. Hefeeda, S. Hambrusch, and B. Bhargava. On peer-to-peer media streaming. In *22nd International Conference on Distributed Computing Systems (ICDCS 2002)*, pages 363–371, 2002.

[65] M. Yajnik, J. Kurose, and D. Towsley. Packet loss correlation in the mbone multicast network. In *Global Telecommunications Conference, (GLOBECOM 1996)*, pages 94–99, 1996.

[66] Youtube's website. http://youtube.com.

[67] R. Zhang, S. L. Regunathan, and K. Rose. Video coding with optimal inter/intra-mode switching for packetloss resilience. *IEEE Journal on Selected Areas in Communications*, 18(6):966–976, 2000.

[68] Y. Zhang, W. Gao, Y. Lu, Q. Huang, and D. Zhao. Joint source-channel rate-distortion optimization for h. 264 video coding over error-prone networks. *IEEE Transactions on Multimedia*, 9(3):445–454, 2007.

[69] C. Zhu, X. Lin, and L.P. Chau. Hexagon-based search pattern for fast block motion estimation. *IEEE Transactions on Circuits and Systems for Video Technology*, 12(5):349–355, 2002.

[70] C. Zhu, Y.K. Wang, and H. Li. Adaptive redundant picture for error resilient video coding. In *IEEE International Conference on Image Processing (ICIP2007)*, volume 4, pages 253–256, 2007.

Index

ACK, 12
ACK message, 25
Alexa, 1
aliasing, 45
anti-causal, 47, 49
Arbitrary Slice Ordering, 8
ARQ, 2, 11, 12, 16, 19, 69, 79, 83
ASO, 8
Automatic Repeat reQuest, 2, 16, 19
average burst length, 35

B-frame, 7, 12
Binary symmetric channel, 17
binomial distribution, 71
bit-error, 18
bitrate, 4
block-based whole-frame concealment, 44
Boundary matching algorithym, 50
boundary-matching algorithm, 46
Broadcast, 68
Broadcasting, 6
BSC, 17
burst error, 15

CABAC, 8
cable, 6
causal, 46, 49
CAVLC, 8
CBR video, 35
channel capacity, 16
Channel Coding, 3, 16
CIF, 1, 2
client-server, 68
CoDiO, 72, 80
Collision Distortion Optimization, 72
commercial video streaming systems, 23
Compression, 1
conferencing, 5
Context Adaptive Binary Arithmetic Coding, 8
Context adaptive entropy coding, 8
Context Adaptive Variable-Length Coding, 8
Conversational, 6
conversational, 5
Convolutional Codes, 17
convolutional encoder, 18
CRC, 18, 19
cumulative error, 5
cumulative function, 71
current frame, 24
Cyclic redundancy check, 18

Data Partitioning, 10
Data partitioning, 8
DCT, 3
dequantization, 5
digital TV broadcasting, 7
Digital Video, 1
Directional spatial prediction, 8
Discrete Cosine Transform, 3, 4
displaying order, 7
DP, 10
drift, 5

edge preserving smoothing, 43
entropy coding, 4, 24
erasure channel, 16–18
Error Concealment, 3, 14
error concealment, 10, 43
error detection, 18

Error Distribution, 88
Error propagation, 23
error propagation, 9
Error Resilience, 3, 8
exhaustive search, 78

fast RPS, 28
FEC, 2, 12, 16, 19, 69
feedback, 9, 11
feedback channel, 68
feedback implosion, 68
File Transfer Protocol, 2
Flexible Macroblock Ordering, 8, 10
Flexible Reference Frame Concept, 11
Flexible slice size, 8
FMO, 8, 10
Forward Error Correction, 2, 16
frame buffer, 24
FS, 83
FTP, 2
Full Search, 74, 83
full search algorithm, 78

Gilbert, 35
Google, 1
GOP, 73
Group of Pictures, 73

H.261, 5
H.262, 5
H.263, 5
H.263+, 5
H.264/AVC, 2, 5, 24
H.32X, 7
HD, 2
HD Video, 1
header, 10
Human Vision System (HVS), 3
Hybrid ARQ/FEC, 20
Hybrid ARQ/FEC type-II, 69

I-frame, 12, 76
IDR, 11, 12
IEEE 802.11n, 2
IID, 35
independent, identically distributed, 35
infrequent error, 20
Instantaneous Decoder Refresh, 11
Interactive, 6
interleaver, 16
Internet, 1, 5, 6
Intra coding, 8, 12
intra-coded pixels, 46
intra-coding, 49
intra-frame encoder, 25
intra-refresh, 9, 12
inverse transform, 5, 24
IP network, 2
ISO MP4, 7
ISO/IEC Moving Picture Expert Group, 5
ITU-T Video Coding Expert Group, 5

joint video team, 5
JVT, 5

Karhunen-Loeve Transform, 3
KLT, 3

Lagrange multiplier, 25
Lagrangian cost function, 25
LAN, 6
linear space invariance, 107
local optima, 82
Loop Filter, 8
loss indicator matrix, 75
loss-aware rate-distortion optimization, 9
lossless compression, 3
lossy channel, 2
lossy compression, 3
LSI, 107

Macroblocks, 9
MCPR, 4, 24
MDDE, 76
measured packet traces, 35
median filter, 45
memoryless binary symmetric channel, 16
Metropolis state calculation rule, 82

MIMD, 37
mobile network, 6
Motion Compensated Prediction Residual, 4
motion compensation, 3–5
Motion estimation, 24
motion estimation, 3–5
motion vectors, 10
motion-compensated prediction, 24
motion-compensated prediction residual, 24
MPEG, 5
MPEG-1, 5, 9
MPEG-2, 5, 9
MPEG-4 (Part 2), 5
MPEG-ISO/IES, 5
Multi-Decoder Distortion Estimation, 76
multi-frame optical flow estimation, 44
Multicast, 68
multicast group, 71
multicast streaming, 67
Multiple Instruction, Multiple Data, 37
Multiple reference, 7

NACK message, 25
NAL, 6, 7
Network Abstraction Layer, 6
network-friendly, 5
non-conversational, 5
noncausal, 49
Noncausal optical flow, 46
noncausal source, 48
Not protected, 83

Occlusion, 49
open-after-downloading, 68
optical flow estimation, 15
Optical Flow principle, 46
over-designed FEC, 16

P-frame, 12
P2P, 68
Parameter set structure, 8
Peer to Peer, 68
pixel-based whole-frame concealment, 44
play-while-downloading, 68
positive acknowledgement, 12
power spectral density, 30
pre-computed motion vectors, 27
predictively-coded video, 23
PSD, 30
PSNR, 53

QCIF, 15
quantization, 24
quantized MCPR, 24
quantizer, 4

RaDiO, 72, 80, 83
Rate Distortion Optimization, 72
rate-distortion, 25
RealPlayer, 75
Recursive Optimal per-Pixel Estimation of decoder distortion, 76
redundancy, 3
Redundant pictures, 8
Redundant Slices, 10
Reed-Solomon, 73
Reed-Solomon code, 17
Reference Picture Selection, 9, 11, 23
referencing order, 7
Region Of Interest, 11
residuals, 10
retransmission-based algorithms, 69
ROI, 11
ROPE, 76
Round Trip Time, 25, 51
RPS, 9, 11, 23, 71
RPS-ACK, 12, 13, 27
RPS-NACK, 12–14, 27, 48
RS, 73
RS code, 17
RTP, 35
RTP/IP, 7
RTT, 25, 51

SA, 83
satellite, 6
self-scaling, 68
serial storage, 6

Shannon's channel coding theorem, 16
Signal-to-Noise Ratio, 53
SIMD, 37
Simulated Annealing, 82, 83
Single Instruction, Multiple Data, 37
SISD, 37
Slice Coding, 9
SP/SI, 8
spatial domain, 43
Spatial Error Concealment, 14
spatial redundancy, 3
spatiotemporal, 50
Spatiotemporal boundary matching, 50
spatiotemporal domain, 43
Spatiotemporal Error Concealment, 15
standard video coder, 4, 24
statistical redundancy, 3
Suboptimal Search, 82
Subpixel motion vector accuracy, 7
subset, 73
Subset Selection, 73, 80
synchronization/switching pictures, 8
systematic Reed-Solomon code, 70

TCP, 2
temporal domain, 43
Temporal Error Concealment, 14
temporal redundancy, 3
traffic congestion, 15
transform, 24
TV stations, 68
two-stage encoder, 25, 27
two-state Markov, 35
two-way channel, 19
Type-I Hybrid ARQ/FEC, 20
Type-II Hybrid ARQ/FEC, 20
Type-II hybrid ARQ/FEC, 73, 79, 80, 83

under-designed FEC, 16
unicast, 20, 67
uplink, 68

variable block-size, 7
VBR video, 35
VCEG, 5
VCL, 6
video, 2
Video Coding Layer, 6
Video Compression, 3
video sequences, 34
video-on-demand, 24
Virtual Multicast, 67
Viterbi algorithm, 17
VLSI, 37

Weighted prediction, 7
Whole Frame Concealment, 15
Wide-Sense Stationary, 32
Windows Media, 75
wireless networks, 5
WSS, 32

YouTube, 1
YUV 4:2:0, 28