Combined Forward Error Correction and Error Concealment for Digital Video Transmission

Yan Mei

A Thesis
in
The Department
of
Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements
for the Degree of Master of Applied Science at
Concordia University
Montréal, Québec, Canada

June 2001

© Yan Mei, 2001
The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author’s permission.

L’auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L’auteur conserve la propriété du droit d’auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-64061-2
ABSTRACT

Combined Forward Error Correction and Error Concealment for Digital Video Transmission

Yan Mei

Forward Error Correction (FEC) coding has been widely used in digital communications to improve the transmission performance over noisy channels. Based on the introduced redundancy and encoding structure, a maximum likelihood (ML) decoding scheme can be used to correct significant part of transmission errors. For digital video transmission, further enhancement of the reconstructed video signal can be done by error concealment based on the residual redundancy of the video signal. FEC and error concealment are usually treated as independent processes.

This thesis presents an error resilience scheme, which combines Forward Error Correction (FEC) and Error Concealment (EC) in order to further enhance the quality of the video signal transmitted over a noisy channel. Of particular, for a given FEC scheme in use, a Multiple Candidate Likelihood (MCL) channel decoding strategy is proposed. Unlike the traditional ML decoder that provides only one corrected sequence, the proposed MCL decoding strategy offers a number of candidates with their reliability information. These candidates are further examined in terms of syntax/semantic validity and their discontinuity measure, which are related to the video source coding.

Performance of video signals in terms of the peak signal-to-noise ratio (PSNR) as well as subjective measure by visual inspection of video frames and sequence is investigated for different video sequences and various FEC coding schemes. Both simulation and analysis are used to evaluate performance of the proposed schemes and to assess their complexity. Research results indicate that the proposed strategy outperforms the traditional ML decoding technique for the same FEC code in use, especially at the high channel bit error rates (BER). Simulation results are in a good agreement with the analytical prediction. Selection of key parameters and characteristics of the proposed combined FEC and EC are discussed.

Keywords: Video transmission, error concealment, forward error correction, multiple candidate likelihood decoder, syntax, discontinuity
Dedicated to my father and mother ......
ACKNOWLEDGEMENTS

First I would like to express my deepest appreciation to my advisors Dr. Le-Ngoc and Dr. Lynch for proposing the research topic, providing a continuous support and directing this work. I would also like to thank all the professors with whom I have interacted during my studies at Concordia University.

I also would like to thank my family and my boyfriend Xu. Their love, trust and support throughout all these years made the impossible possible. I am grateful to you.

Last but not least, thanks to all my friends in Concordia for their constant moral support and for sharing with me a challenging experience.

Thanks also for financial supports from the Natural Sciences and Engineering Research Council of Canada.
# TABLE OF CONTENTS

 LIST OF TABLES .......................................................... viii  
 LIST OF FIGURES .......................................................... ix  
 LIST OF ABBREVIATIONS AND SYMBOLS .......................... xiii  

## 1 Introduction  
1.1 Transmission of compressed video ............................. 2  
1.2 Problem statement .................................................. 6  
1.3 Figures of Merit ..................................................... 7  
1.4 Outline of the thesis .............................................. 9  

## 2 Error Resilience of Compressed Video over Noisy Channel: An Overview  
2.1 MPEG2 video standard .............................................. 12  
  2.1.1 Context of the MPEG standard ............................... 12  
  2.1.2 Video encoder .................................................. 15  
  2.1.3 Video decoder .................................................. 17  
  2.1.4 Structure of the MPEG2 bitstream ......................... 18  
2.2 Error control coding .............................................. 21  
2.3 Error concealment techniques ................................... 23  
2.4 Survey on joint FEC/EC techniques ............................. 25  

## 3 A Combined FEC and Error Concealment Technique  
3.1 Source information applicable to error concealment .......... 30  
  3.1.1 Syntax .......................................................... 30  
  3.1.2 Discontinuity measure ....................................... 32  
3.2 Error concealment using syntax information .................. 36  
  3.2.1 Syntax Based Error Concealment (SBEC) .................... 36  
  3.2.2 Syntax and Discontinuity Based Error Concealment (SDBEC) 37  
3.3 Proposed scheme .................................................. 37  
  3.3.1 Multiple-Candidate Likelihood (MCL) decoder ............. 39  
  3.3.2 Benefits of MCL decoding ................................... 41  
  3.3.3 Size of slice candidate groups ............................. 43  
  3.3.4 Syntax checker ............................................... 44  
  3.3.5 Discontinuity measure selector ............................. 44
Performance Evaluation

4.1 Performance of classical ML receiver using (16,8) quasi-cyclic code .......................... 57

4.2 Performance of the proposed combined FEC/EC scheme using a (16,8) quasi-cyclic code

4.2.1 Using slice candidates with shortest distance .......................................................... 63

4.2.2 Using slice candidates in $G_0$ and $G_1$ ................................................................. 69

4.2.3 Using the slice candidates in $G_0$, $G_1$ and $G_2$ .................................................. 79

4.3 Simulation results with a higher-rate (21,16) code ...................................................... 85

4.3.1 Performance of a classical ML decoder ................................................................. 85

4.3.2 Performance of the proposed scheme ...................................................................... 91

4.4 Complexity of the proposed scheme ........................................................................... 95

4.4.1 Number of slice candidates to be decompressed ...................................................... 95

4.4.2 Number of bits extracted from the bitstream .......................................................... 98

4.5 Further discussions ...................................................................................................... 102

4.5.1 Value of expanding the search space ......................................................................... 102

4.5.2 Evaluate discontinuity measure and $\alpha_a$ ............................................................. 105

5 Conclusions and Future Work

5.1 Contribution .................................................................................................................. 108

5.2 Conclusions .................................................................................................................. 109

5.3 Future work .................................................................................................................. 109

Bibliography ....................................................................................................................... 111

A (16,8) quasi-cyclic code ................................................................................................. 114

B Matlab source code to derive the distance profile for the (16,8) quasi-cyclic code 116

C Variances of $N_0$, $N_1$ and $N_2$ .................................................................................. 126
LIST OF TABLES

3.1 The distance profile $\gamma(r,j)$ for the received words, with probability calculated for
BER of $1.2 \times 10^{-2}$ ................................................. 47

3.2 Values of $E(f_d(x_i))$ for $p = 1.2 \times 10^{-2}$ and (16,8) quasi-cyclic code ................. 51

4.1 PSNR for different $\alpha_1$, at BER of $1.2 \times 10^{-2}$ ................................................................. 73

4.2 PSNR at different $\alpha_2$, at BER of $1.2 \times 10^{-2}$ ................................................................. 79

4.3 Number of bits extracted for different search space ................................................................. 99

4.4 Number of slices whose correct slice candidate $S_{oi}/s$ is in each group, for different BER's ................................................................. 103

4.5 Number of slices selecting slice candidates correctly, at BER $1.2 \times 10^{-2}$ ................. 103

4.6 PSNR of slices out of search space, at BER $1 \times 10^{-2}$ ................................................................. 104

4.7 PSNR of slices out of search space, at BER $1.2 \times 10^{-2}$ ......................................................... 104

4.8 Slices selecting wrong groups and correct groups but wrong slice candidates ................. 106
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Frames with bit errors</td>
<td>4</td>
</tr>
<tr>
<td>1.2</td>
<td>Temporal effect of bit errors</td>
<td>5</td>
</tr>
<tr>
<td>1.3</td>
<td>Problem statement</td>
<td>6</td>
</tr>
<tr>
<td>2.1</td>
<td>Block diagram of an MPEG encoder</td>
<td>16</td>
</tr>
<tr>
<td>2.2</td>
<td>Block diagram of an MPEG2 decoder</td>
<td>17</td>
</tr>
<tr>
<td>2.3</td>
<td>Layered structure of MPEG2 bitstream</td>
<td>19</td>
</tr>
<tr>
<td>2.4</td>
<td>Protecting MPEG2 using a FEC code</td>
<td>21</td>
</tr>
<tr>
<td>3.1</td>
<td>Frame 15 from “Table Tennis”, with a edge between the right leg and the background</td>
<td>32</td>
</tr>
<tr>
<td>3.2</td>
<td>The pixel values at the edge</td>
<td>33</td>
</tr>
<tr>
<td>3.3</td>
<td>Differentiation of the pixel values</td>
<td>33</td>
</tr>
<tr>
<td>3.4</td>
<td>Location of pixels to be used in discontinuity measure</td>
<td>34</td>
</tr>
<tr>
<td>3.5</td>
<td>The SBEC scheme (no dashed line), and the SDBEC scheme (includes dashed line)</td>
<td>37</td>
</tr>
<tr>
<td>3.6</td>
<td>Structure of proposed combined FEC/EC scheme</td>
<td>38</td>
</tr>
<tr>
<td>3.7</td>
<td>Simulated number of slice candidates in $G_0$ for each slice of the sequence “Table Tennis”, at BER of $1.2 \times 10^{-2}$</td>
<td>43</td>
</tr>
<tr>
<td>3.8</td>
<td>Central limit and Chebyshev limits for 90% confidence interval for $N_0$</td>
<td>53</td>
</tr>
<tr>
<td>3.9</td>
<td>Chebyshev’s limit for 90% confidence interval for $N_1$</td>
<td>54</td>
</tr>
<tr>
<td>3.10</td>
<td>Chebyshev’s limit of 90% confidence interval for $N_2$</td>
<td>54</td>
</tr>
<tr>
<td>4.1</td>
<td>PSNR versus channel BER for “Table Tennis”, without and with FEC (from top to bottom: Luminance, Blue Chrominance, Red Chrominance)</td>
<td>58</td>
</tr>
<tr>
<td>4.2</td>
<td>Post decoding error probability versus input channel BER for the (16,8) quasi-cyclic code</td>
<td>59</td>
</tr>
<tr>
<td>4.3</td>
<td>Frames 120 from compressed “Table Tennis” (2Mb/s), from top to bottom: no error, without FEC and with channel BER of $1 \times 10^{-4}$, with classic FEC and with channel BER of $1 \times 10^{-4}$</td>
<td>60</td>
</tr>
<tr>
<td>4.4</td>
<td>Frames 69 from compressed “Table Tennis” (2Mb/s): from top to bottom: no error, without FEC and with channel BER of $2 \times 10^{-3}$, with classic FEC and with channel BER of $2 \times 10^{-3}$</td>
<td>61</td>
</tr>
</tbody>
</table>
4.5 Frames 0 from compressed "Table Tennis" (2Mb/s), from top to bottom: no error, without FEC and with channel BER of $1.2 \times 10^{-2}$, with classic FEC and with channel BER of $1.2 \times 10^{-2}$ ........................................ 62

4.6 PSNR versus channel BER for "Flower Garden", without FEC and with FEC (from top to bottom: Luminance, Blue Chrominance, and Red Chrominance) .................. 64

4.7 Frames 0 from compressed "Flower Garden" (2Mb/s), from top to bottom: no error, without FEC and with channel BER of $1 \times 10^{-4}$, with classic FEC and with channel BER of $1 \times 10^{-4}$ ................................................................. 65

4.8 Frames 136 from compressed "Flower Garden" (2Mb/s), from top to bottom: no error, without FEC and with channel BER of $2 \times 10^{-3}$, with classic FEC and with channel BER of $2 \times 10^{-3}$ ................................................................. 66

4.9 Frames 14 from compressed "Flower Garden" (2Mb/s), from top to bottom: no error, without FEC and with channel BER of $1.2 \times 10^{-2}$, with classic FEC and with channel BER of $1.2 \times 10^{-2}$ ........................................................................ 67

4.10 PSNR versus BER for "Table Tennis", classic FEC and SSG0, from top to bottom:
Luminance, Blue Chrominance and Red Chrominance ......................................... 68

4.11 Frames 2 from compressed "Table Tennis" (2Mb/s): from top to bottom: no error, with classic FEC and with channel BER of $1.2 \times 10^{-2}$, with SSG0 and with channel BER of $1.2 \times 10^{-2}$ ................................................................. 70

4.12 Frames 135 from compressed "Table Tennis" (2Mb/s): from top to bottom: no error, with classic FEC and with channel BER of $1.2 \times 10^{-2}$, with SSG0 and with channel BER of $1.2 \times 10^{-2}$ ................................................................. 71

4.13 Frames 179 from compressed "Table Tennis" (2Mb/s), from top to bottom: no error, with classic FEC and with channel BER of $1.2 \times 10^{-2}$, with SSG0 and with channel BER of $1.2 \times 10^{-2}$ ................................................................. 72

4.14 PSNR versus $a_1$ for "Table Tennis" at BER of $1.2 \times 10^{-2}$, from top to bottom:
Luminance, Blue Chrominance, and Red Chrominance ......................................... 74

4.15 PSNR versus BER for "Table Tennis" (2Mb/s), with SSG0 and SSG1, from top to bottom: Luminance, Blue Chrominance, and Red Chrominance ......................................... 75

4.16 Frames 2 from compressed "Table Tennis" (2Mb/s), from top to bottom: no error, with SSG0 and with channel BER of $1.2 \times 10^{-2}$, with SSG1 and with BER of $1.2 \times 10^{-2}$ 76

4.17 Frames 46 from compressed "Table Tennis" (2Mb/s), from top to bottom: no error, with SSG0 and with channel BER of $1.2 \times 10^{-2}$, with SSG1 and with channel BER of $1.2 \times 10^{-2}$ ................................................................. 77
4.18 Frames 169 from compressed “Table Tennis” (2Mb/s), from top to bottom: no error, with SSG0 and with channel BER of $1.2 \times 10^{-2}$, with SSG1 and with channel BER of $1.2 \times 10^{-2}$ .......................... 78

4.19 PSNR versus $\alpha_2$ for “Table Tennis” (2Mb/s), at BER of $1.2 \times 10^{-2}$, from top to bottom: Luminance, Blue Chrominance, Red chrominance ............................ 80

4.20 PSNR versus BER for “Table Tennis” (2Mb/s), from top to bottom: Luminance, Blue Chrominance and Red Chrominance .......................... 81

4.21 Frames 2 from compressed “Table Tennis” (2Mb/s), from top to bottom: no error, with SSG1 and with BER of $1.2 \times 10^{-2}$, with SSG2 and with BER of $1.2 \times 10^{-2}$ 82

4.22 Frames 75 from compressed “Table Tennis” (2Mb/s), from top to bottom: no error, with SSG1 and with BER of $1.2 \times 10^{-2}$, with SSG2 and with BER of $1.2 \times 10^{-2}$ 83

4.23 Frames 169 from compressed “Table Tennis” (2Mb/s), from top to bottom: no error, with SSG1 and with BER of $1.2 \times 10^{-2}$, with SSG2 and with BER of $1.2 \times 10^{-2}$ 84

4.24 PSNR versus BER for “Flower Garden” (2Mb/s) with classic FEC and SSG2, from top to bottom: Luminance, Blue Chrominance, and Red Chrominance for “Flower Garden” 86

4.25 Frames 25 from compressed “Flower Garden” (2Mb/s), from top to bottom: no error, with classic FEC and with BER of $1.2 \times 10^{-2}$, with SSG2 and with BER of $1.2 \times 10^{-2}$ 87

4.26 Frames 100 from compressed “Flower Garden” (2Mb/s), from top to bottom: no error, with classic FEC and with BER of $1.2 \times 10^{-2}$, with SSG2 and with BER of $1.2 \times 10^{-2}$ 88

4.27 Frames 159 from compressed “Flower Garden” (2Mb/s), from top to bottom: no error, with classic FEC and with BER of $1.2 \times 10^{-2}$, with SSG2 and with BER of $1.2 \times 10^{-2}$ 89

4.28 PSNR versus BER for “Table Tennis” (2Mb/s), without FEC and with (21,16) FEC code, from top to bottom: Luminance, Blue Chrominance and Red Chrominance 90

4.29 Post decoding error probability versus input channel BER for the (21,16) FEC code 91

4.30 Frames 2 from compressed “Table Tennis” (2Mb/s), from top to bottom: no error, without FEC and with channel BER of $2 \times 10^{-3}$, with (21,16) FEC and with channel BER of $2 \times 10^{-3}$ 92

4.31 PSNR versus BER for “Table Tennis” (2Mb/s), with (21,16) FEC code and with SSG2, from top to bottom: Luminance, Blue Chrominance and Red Chrominance 93

4.32 Frame 2 from compressed “Table Tennis” (2Mb/s), from top to bottom: no error, with (21,16) FEC and with BER of $2 \times 10^{-3}$, with SSG2 and with BER of $2 \times 10^{-3}$ 94
4.33 Statistical collection of numbers of slice candidates in $G_0$ versus slice length, at BER of $1.2 \times 10^{-2}$ .............................................................. 95
4.34 Statistical collection of numbers of slice candidates in $G_1$ versus slice length, at BER of $1.2 \times 10^{-2}$ .............................................................. 96
4.35 Statistical collection of numbers of slice candidates in $G_2$ versus slice length, at BER of $1.2 \times 10^{-2}$ .............................................................. 96
4.36 Average number of slice candidates in $G_0$ versus slice length ........................................ 97
4.37 Average number of slice candidates in $G_1$ versus slice length ........................................ 97
4.38 Average number of slice candidates in $G_2$ versus slice length ........................................ 98
4.39 Central limit confidence limit of 90% confidence interval for $N_0$ ........................................ 99
4.40 Chebyshev limit of 90% confidence interval for $N_0$ ......................................................... 99
4.41 Central limit and Chebyshev limits of 90% confidence interval for $N_0$ ................................. 100
4.42 Chebyshev’s limit for 90% confidence interval for $N_1$ .................................................... 100
4.43 Chebyshev’s limit for 90% confidence interval for $N_2$ .................................................... 101
4.44 Number of bits extracted versus BER for different search space ......................................... 101

C.1 Calculation figure for $E\{b^2\}$ ........................................................................ 129
C.2 Calculation figure for $E\{ab\}$ ........................................................................ 133
**LIST OF ABBREVIATIONS AND SYMBOLS**

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATM</td>
<td>Asynchronous Transmission Mode</td>
</tr>
<tr>
<td>BE</td>
<td>Bandwidth Expansion</td>
</tr>
<tr>
<td>BER</td>
<td>Bit Error Rate</td>
</tr>
<tr>
<td>BSC</td>
<td>Binary Symmetric Channel</td>
</tr>
<tr>
<td>DC</td>
<td>Director Current</td>
</tr>
<tr>
<td>DCT</td>
<td>Discrete Cosine Transform</td>
</tr>
<tr>
<td>EC</td>
<td>Error Concealment</td>
</tr>
<tr>
<td>FEC</td>
<td>Forward Error Correction</td>
</tr>
<tr>
<td>FLC</td>
<td>Fixed Length Code</td>
</tr>
<tr>
<td>GF</td>
<td>Galois Field</td>
</tr>
<tr>
<td>GOP</td>
<td>Group Of Pictures</td>
</tr>
<tr>
<td>HDTV</td>
<td>High Definition TV</td>
</tr>
<tr>
<td>IDCT</td>
<td>Inverse Discrete Cosine Transform</td>
</tr>
<tr>
<td>ISO</td>
<td>International Standard Organization</td>
</tr>
<tr>
<td>MAP</td>
<td>Maximum A Posteriori</td>
</tr>
<tr>
<td>MAD</td>
<td>Mean Absolute Difference</td>
</tr>
<tr>
<td>MB</td>
<td>Macroblock</td>
</tr>
<tr>
<td>MCL</td>
<td>Multiple Candidate Likelihood</td>
</tr>
<tr>
<td>ML</td>
<td>Maximum Likelihood</td>
</tr>
<tr>
<td>MPEG</td>
<td>Moving Picture Experts Group</td>
</tr>
<tr>
<td>MSE</td>
<td>Mean Squared Error</td>
</tr>
<tr>
<td>PSNR</td>
<td>Peak Signal to Noise Ratio</td>
</tr>
<tr>
<td>SBEC</td>
<td>Syntax Based Error Concealment</td>
</tr>
<tr>
<td>SDBEC</td>
<td>Syntax and Discontinuity Based Error Concealment</td>
</tr>
<tr>
<td>SNR</td>
<td>Signal to Noise Ratio</td>
</tr>
<tr>
<td>SSG0</td>
<td>Search Space Group G0</td>
</tr>
<tr>
<td>SSG1</td>
<td>Search Space Group G1</td>
</tr>
<tr>
<td>SSG2</td>
<td>Search Space Group G2</td>
</tr>
<tr>
<td>VLC</td>
<td>Variable Length Code</td>
</tr>
<tr>
<td>VLSI</td>
<td>Very Large Scale Integration</td>
</tr>
<tr>
<td>VQ</td>
<td>Vector Quantization</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction
1.1 Transmission of compressed video

With the rapid deployment and growth of communication systems, image and video transmission is a particularly important and challenging application that deserves attention. Video signals require a huge amount of resources to transmit when compared to audio or text information. For example, for a one-second sequence at 30 frames/sec and 704*480 pixels for the luminance component [1], which is closely related to the perception of brightness, and subsampled in both spatial directions for the color components (4:2:0 format), which are related to the perception of color hue and saturation, requires about 15 Mbytes or 120 Mb/s. Despite the development of broadband networks, compression techniques are needed to reduce the required transmission bandwidth. Even assuming the compression ratio is only 15:1, the transmission speed for the video above would be reduced to 8 Mb/s. If the compression ratio is higher, i.e., 25:1 or 40:1, the required transmission bandwidth will be lower.

Compression is likewise important when storing image or video signals despite the availability of larger storage devices. Modern image and video compression techniques offer the possibility to store and transmit the vast amount of data necessary to represent digital images and video in an efficient way. The video compression algorithms developed by the Moving Pictures Expert Group (MPEG) [2] have developed into important and successful video coding standards worldwide. The group has produced the MPEG-1, MPEG-2 and MPEG-4 standards. An increasing number of MPEG VLSI chip-sets and products are becoming available on the market [3]. This has led to a wide range of applications [4], such as video on demand, digital TV/HDTV, terrestrial and satellite broadcasting, multimedia, image/video database services.

While storage devices are generally reliable, communication networks often introduce errors. Two types of errors that may occur are cell loss and data corruption [5]. In packet switched networks, cell loss due to network congestion is an important source of transmission errors. When several sources transmit at their peak rates
simultaneously, the buffer space at some switches may be inadequate. The congestion at those switches will lead to cell loss due to buffer overflow. Thus in cell loss many bits often from the same spatial-temporal area of the video are lost. The large number of bits lost often results in large areas of the image being corrupted. The impact of cell loss may become more significant as the packet or cell size become larger.

Data corruption occurs when noise and/or interference in the communication channel introduces bit errors. Variable length codes (VLC's) like the Huffman code are often used in image/video compression. If an error causes the decompressor to lose track of where the current VLC begins and ends then not only that VLC but many or all subsequent VLC's may be decoded incorrectly. Thus a single bit error in a coded bitstream can cause a propagating error that can lead to objectionable degradations over a large portion of the video. Generally standards insert resynchronization points to stop indefinite propagation. Since these resynchronization points are somewhat costly in bits they occur relatively infrequently. See Section 2.1.3 for the placement of resynchronization points in MPEG2. In addition to propagation due to loss of VLC synchronization, errors may also propagate in the temporal direction because of the motion compensated predictive interframe coding.

In this thesis techniques to alleviate bit errors are studied and developed. Cell or packet loss is not considered.

Figure 1.1 shows one frame from the sequence "Table Tennis" decompressed, with a channel Bit Error Rate (BER) of $10^{-5}$ and $10^{-4}$, respectively. A single bit error causes the lose of pixels and thus the black stripe in half a line in the first frame. The second frame contains a greater degradation due to the higher BER. Figure 1.2 shows the impact of bit errors in the temporal direction. The first figure shows frame 0, where bit errors occurred, while the second figure shows the impact of those bit errors after 12 frames.
Figure 1.1: Frames with bit errors
Figure 1.2: Temporal effect of bit errors
1.2 Problem statement

In this thesis, error resilience techniques to alleviate the effects of bit errors are studied and developed. Figure 1.3 shows the block diagram of a typical video communication systems.

Traditionally, channel encoder/decoder is used to enhance the transmission performance, independently of the video decompressor. The channel encoder adds redundancy to the transmitted data in a systematic manner so that the channel decoder can base on the encoding structure to correct errors occurring in the channel. Compressed video signals also contain residual correlation, which can be used to conceal errors. Various error concealment techniques have been proposed [6]. The channel coding/decoding and error concealment are generally treated in an independent manner.

In this work, an error resilience scheme is proposed. It combines the channel decoding and error concealment in order to further enhance the quality of the video signal transmitted over a noisy channel. Specifically, a multiple-candidate likelihood (MCL) channel decoding scheme is considered. It provides a number of candidates with their corresponding reliability information rather than only one solution. These candidates are further examined by the error concealment scheme based on the...
residual redundancy of the video signal to select the final solution.

The proposed error resilience scheme aims to make use of the existing channel coding and to exploit a joint error decoding/concealment strategy, which maximizes the capability of the channel code and residual redundancy of the video signal.

1.3 Figures of Merit

Video quality or fidelity can be evaluated subjectively or objectively. At this time a rigorous subjective quality assessment requires a set of impartial human viewers (twenty five or more), either expert or non expert to rate the degraded video either alone or in relation to an “original” sequence [7, 8, 9]. Since human observers are the ultimate consumers of most video this is considered the “best” way to measure video quality. However, this can be prohibitively costly and time consuming to assess the subjective quality of video in this way, in particular at the algorithmic development and the parameter tweaking stage. There are efforts [9] to automate the subjective evaluation of video quality but they have not been successful up to now. In this thesis non-rigorous methods of subjective assessment are used. Specifically two expediencies are employed: individual frames are shown in the thesis for the readers’ own judgment and the author’s comments on subjective quality are inserted in the text.

Objective measures of video quality do not need to use human subjects and are generally automated (which is their main advantage). Mean square of error (MSE), peak signal to noise ratio (PSNR), mean absolute difference (MAD) are among the common objective measures. PSNR[10], which is a variation of the MSE is used as an objective measure of goodness in this work. PSNR for the luminance component[1] is defined as

\[
PSNR_Y \triangleq 10 \log_{10} \frac{255^2}{\frac{1}{N_Y} \sum_{i=0}^{N_Y} (x_{oy}[i] - x_{ry}[i])^2} (dB)
\]  

(1.1)

where \(x_{oy}[i]\) denotes luminance component of original image, \(x_{ry}[i]\) denotes the
luminance of reconstructed image, and \( N_Y \) denotes the total number of luminance pixels. Similarly, the PSNR for the chrominance components can be defined as

\[
PSNR_u = 10 \log_{10} \frac{255^2}{\frac{1}{N_u} \sum_{i=0}^{N_u} (x_{o_u}[i] - x_{r_u}[i])^2} (dB)
\]

\[
PSNR_v = 10 \log_{10} \frac{255^2}{\frac{1}{N_v} \sum_{i=0}^{N_v} (x_{o_v}[i] - x_{r_v}[i])^2} (dB)
\]

where \( x_{o_u}[i], x_{o_v}[i] \) denote the blue and red chrominance components of the original image, \( x_{r_u}[i], x_{r_v}[i] \) denote those of reconstructed image, and \( N_u, N_v \) denote the number of chrominance pixels. Note that in PSNR, the peak signal value of 255 is used for the signal value. That is because traditional SNR is unfortunate in that a very bright image might mask the error since the signal part would be much larger. Thus PSNR that is the MSE put on a logarithmic scale with a suitable common value (255) put into to compare against is used.

Due to the compression distortion, the PSNR of the sequence before transmission is not infinite. Generally speaking, video with a PSNR of 40 dB is close to perfect. A picture with a PSNR 30 dB is acceptable. But when the PSNR drops below mid 25 dB, the quality of picture is not good. All of these figures can vary with the content of the video and the nature of the degradation.

In this thesis, for every simulation sequence, a PSNR is calculated and is used to draw PSNR versus Bit Error Rate (BER) curves. It should be noted that for the same sorts of impairments the PSNR and the subjective assessment of human observers will tend to track each other; however, there is no guarantee of this.

The Bandwidth Expansion (BE), defined as

\[
BE = \frac{\text{codewordsize}}{\text{datablocksize}}
\]

represents the increase in required bandwidth to cover extra parity added to the information prior to the transmission. It is the multiplicative inverse of the code
rate of the FEC code in use.

The computational complexity of an algorithm is also a figure of merit. Ideally the complexity can be measured in terms of number of operations. In this work it is often difficult to count the number of operations. Instead, two proxies named "number of slice candidates" and "number of bits extracted" are used for complexity. See section 3.4.

The figures of merit above are affected by the channel BER. They need to be evaluated over a range of BER's. In communications, there are close relationships between BER and transmitted power, speed, and complexity etc [11]. As BER increases, both the subjective and objective quality of the transmitted pictures would degrade. More transmitted power is needed to minimize the effects of noise. To compensate the errors, the complexity of the channel decoder/encoder would increase and the resulting delay would decrease the transmitted speed. Thus it is important to evaluate the performance over a range of BER's.

1.4 Outline of the thesis

This thesis is organized as follows:

Chapter 2 covers background materials related to the work. Important aspects of MPEG2 are introduced. Also, this chapter introduces the classic FEC approach and the general Error Concealment (EC) techniques that make use of the residual redundancy which still exists in the compressed bitstream.

Chapter 3 introduces the techniques that make use of the syntax and discontinuity information and presents the proposed combined FEC/EC scheme. The components of the proposed scheme are presented in detail.

Chapter 4 evaluates the performance of the technique by simulation results, including subjective and objective evaluations for a range of BER's. Experiments to fix parameters of the developed technique are presented. Computational complexity is also discussed.
Chapter 5 concludes the thesis by summarizing the proposed schemes and discussing their performance. Future work is also included in this chapter.
Chapter 2

Error Resilience of Compressed Video over Noisy Channel: An Overview
This chapter briefly covers the necessary background in video compression, transmission and error concealment. Section 2.1 gives an overview of the MPEG2 video compression standard. Classical FEC is introduced in Section 2.2. Section 2.3 covers some error concealment schemes that use the residual redundancy in the video. Section 2.4 gives a survey of joint FEC/EC schemes that combines the error concealment along with the error control coding.

2.1 MPEG2 video standard

This section gives an overview of the MPEG2 video standard. Section 2.1.1 describes the context of the MPEG standards. The MPEG2 video encoder and decoder are introduced in Section 2.1.2 and 2.1.3 respectively. In Section 2.1.4, important information about the structure of MPEG2 bitstream is presented.

2.1.1 Context of the MPEG standard

In 1988 the International Standard Organization (ISO) formed the Moving Picture Experts Group (MPEG) to develop a standard for video and associated audio on digital storage media[12]. The group has produced MPEG-1, MPEG-2 and MPEG-4. The current thrust is MPEG-7 "Multimedia Content Description Interface", scheduled for completion in July 2001. The MPEG activities cover more than video compression since the compression of the associated audio and the issue of the audio visual synchronization can not work independently of the video compression [8, 1, 13].

MPEG-1 is the standard on which such products as Video CD and MP3 are based. It was initially intended for video coding at bit rates of about 1.2 Mb/s with stereo audio coding at bit rates of around 250 Kb/s. It assumes low errors rate and ignores interlaced TV. The standard consists of the following part: system, video, audio, conformance and software [13].

MPEG-2 is the follow-on to MPEG-1 and is the standard on which such products as Digital Television set top boxes and DVD are based. It builds upon the
MPEG-1 capabilities. Five different profiles have been defined for MPEG-2 video, providing particular sets of capabilities ranging from function close to MPEG-1 to the very advanced video compression techniques needed for HDTV. The features offered by MPEG-2 includes: random access, trick modes, multicast to many terminal types, multiple audio and video, and compatible stereoscopic 3D [1, 13].

MPEG-2 systems includes all of the features of MPEG-1 with enhancements that allow it to operate in more error-prone environment. In addition, more video and audio streams can be handled and they are no longer required to have a common time base. MPEG-2 systems still use the pack and packet concepts, but combines them in both backwards compatible and noncompatible ways. Some of the noncompatible ways support asynchronous transmission mode (ATM) applications [1].

MPEG-2 audio is forward and backward compatible with MPEG-1. The requirement to backwards compatible necessitated some coding efficiency compromises. The nonbackwards compatible audio is also being developed [1].

Originally MPEG-2 video was primarily intended for coding of interlaced video of standard TV resolution with good quality in the bit rate range of 4 to 9 Mb/s. However, the scope of MPEG-2 video was considerably revised to include video of higher resolutions such as HDTV at higher bit rate as well as hierarchical video coding for a range of applications. Furthermore, it also supports coding for interlaced video of a variety of resolutions. MPEG-2 video maintains all of the MPEG-1 video syntax, but uses extensions to add additional flexibility and functions. "Scalable" extensions are added to provide video data streams with multiple resolutions for potential normal TV and HDTV coexistence. Other scalable extensions allow the data stream to be partitioned into two pieces. They also offer some temporal flexibility so that not all frames have to be reconstructed. The MPEG-2 video is a syntactic superset of the MPEG-1 video and is thus able to meet the requirement of forward compatibility. The requirement of backward compatibility is achieved via the use of scalability and supported in specific profiles [1, 13].

MPEG-1 and MPEG-2 are very similar in the viewpoint of compression. The
main difference lies in the systems portion of the MPEG-2 standard. MPEG-3 was aimed at higher bit rates (10-20 Mb/s or higher) but it was abandoned when it was clear that MPEG-2 could handle these higher bitrates.

MPEG-4, was originally targeted at very low bitrates (tens to hundreds of kb/s, or video telephone rates). MPEG-1 and MPEG-2 are both block-based compression, i.e. the pictures are partitioned into blocks and processed. MPEG-4 is instead an object-based compression scheme. Its syntax allows the encoder to segment the video into real world objects and then each of these can be compressed separately. The compressor may choose to make its real world objects blocks in which case MPEG-4 is much like MPEG-2. An important advantage of MPEG-4 is that it represents audiovisual data as a composition of multimedia objects, so that they can be transported over network channels providing a QoS appropriate for the nature of the specific media objects; and interact with the audiovisual scene generated at the receiver’s end. MPEG-4 potentially offers better compression than MPEG-2. However, currently its main advantage is that it makes it easier to process different media objects separately and create compound media objects. The work in the future can focus on MPEG-4 bitstream [1, 12].

MPEG-1, MPEG-2 and MPEG-4 are all compression standards. However, MPEG-7 does not focus on compression. It offers a comprehensive set of audiovisual description tools so that large quantities of multimedia objects can be searched/filtered automatically in a database application. This is a challenging task given the broad spectrum of requirements and targeted multimedia applications, and the broad number of audiovisual features of importance in such context. The question of identifying and managing content is not just restricted to database retrieval applications such as digital libraries, but extends to areas like broadcast channel selection, multimedia editing, and multimedia directory services [12].

For further explanation of MPEG, see [13, 3, 1, 8, 12].
2.1.2 Video encoder

Video compression relies on the reduction of the spatial or temporal redundancies in the data. Compression techniques that reduce the spatial correlation are referred to as intraframe coding, while those that reduce the temporal correlation are called interframe techniques. The intraframe coding only uses information from a single picture. It takes advantage of the similarity of a given region of a picture to immediately adjacent areas in the same picture. The interframe coding uses the similarity or predictability from one picture to the neighbors in a sequence. In a first stage the displacement of objects between successive frames is estimated (motion estimation). Then the resulting information is exploited in an efficient interframe predictive coding (motion compensation). The motion compensated predictive interframe coding may cause the error propagation [1].

The MPEG standards do not define the encoding process. They only specify the syntax of the coded bitstream and the decoding process. Based on these requirements, the functions needed by a typical MPEG2 encoder are shown in Figure 2.1 [?, 8].

The uncompressed video is first made some preprocessing. This includes color conversion to YCbCr (luminance and chrominance), format translation (interlace to progressive), prefiltering, and subsampling.

The input video is fed to a motion estimator that feeds a motion vector to the motion compensation, which feeds a prediction to the minus of the subtractor. The prediction error is then passed through to the Discrete Cosine Transform (DCT). DCT based implementations are used because of their high decorrelation performance and the fast DCT algorithms suitable for real-time implementations are available.

The quantization stage comes after the DCT transformation stage. Quantization reduces the possible values for the DCT coefficients and the required number of bits.
Figure 2.1: Block diagram of an MPEG encoder
The final compression stage starts with the serialization of the quantized coefficients. The sequence of coefficients is then coded using Variable Length Code (VLC). The way the VLC allocates code lengths depends on their expected probability of occurrence. Note that the use of VLC, although it provides coding efficiency, makes the compressed bitstream fragile to errors.

Quantized levels are converted to reconstructed DCT coefficients by the dequantizer and Inverse DCT (IDCT) to produce a coded prediction error. The IDCT output is merged with the prediction to form the reconstruction, which is used by the motion estimator to estimate the motion vector. Then the motion vector is sent to the motion compensation to get the prediction.

2.1.3 Video decoder

There are many ways to implement a decoder and the standard does not recommend a particular way. The block diagram of a typical MPEG2 decoder is shown in Figure 2.2 [1, 8].

The bitstream is demultiplexed into overhead information such as motion information, quantization step size, macroblock type and quantized DCT coefficients.
The motion vectors describing the direction and amount of motion of the macroblocks are transmitted to the decoder as part of the bitstream. The decoder then knows which area of the reference picture is used for each prediction. The quantized DCT coefficients are dequantized and are input to the Inverse DCT (IDCT). The IDCT result is added to the result of the prediction to obtained the reconstructed frame. The reconstructed frame is used by the motion compensation to get the prediction.

MPEG2 is an asymmetric system. Decoders, since they merely follow the directions encoded in the bitstream, are relatively simple. Encoders, however, are much more complex than decoders and must have more intelligence.

2.1.4 Structure of the MPEG2 bitstream

In this thesis, the information about MPEG2 bitstream structure is used to do the error concealment. So it is important to clarify the structure of the MPEG2 bitstream.

A video sequence is a series of pictures taken at closely spaced intervals in time. The MPEG2 syntax for the coded video bitstream is given in Figure 2.3. It has an hierarchical representation with six layers[1]:

1. Sequence layer
2. Group of pictures layer
3. Picture layer
4. Slice layer
5. Macroblock layer
6. Block layer

The sequence layer is the top coding layer. It includes a sequence header and is followed by one or more groups of pictures. It ends with a sequence end code. The sequence header includes information like the vertical and horizontal picture size,
Figure 2.3: Layered structure of MPEG2 bitstream
the frame rate, the bit rate, and the minimum buffer size needed by the source decoder, etc.

A Group of Pictures (GOP) is a set of pictures in contiguous display order. MPEG2 supports three types of pictures: I-frames, P-frames and B-frames. Intra-frames (I-frames) are compressed using intraframe coding, without the need to reference to another picture. Predicted frames (P-frames) are coded with reference to the nearest previously coded picture (either I or P frame), using a motion-compensated prediction mechanism. Bidirectional predicted frames (B-frames) use both previous and future I or P frames as reference for motion estimation and compensation. A GOP must contain one I frame. The pictures in a GOP are independent of past GOP’s and thus errors in those past pictures will not propagate temporally between GOP’s. The header of a GOP includes timing information and user data and provides support for random access, fast search, and editing.

A picture in MPEG terminology is the basic unit of display and corresponds to a single frame in the sequence. The header of a picture provides information about the picture type, synchronization, and the resolution and range of the motion vectors.

Each picture is divided into slices. A slice is a horizontal strip of macroblocks within a frame. As shown in Figure 2.3, an MPEG2 slice can be as big as one line of macroblocks across the picture and as small as a single macroblock. It is the basic processing unit in MPEG2, and provides the last VLC resynchronization points within the bitstream. Thus error propagation due to loss of VLC synchronization should not propagate between slices. A slice header contains information for its position within a picture and a quantizer scale factor that can be used by the decompressor to dequantize the coded DCT coefficients. In case of data corruption, the information in the slice header allows for a smoother recovery by the source decoder. The reason for the slice structure is error recovery purpose.

A slice is divided into macroblocks. A macroblock is the basic coding unit with MPEG2. For the 4:2:0 format it consists of a 16×16 pixel array of luminance samples
2.2 Error control coding

The simplified diagram of a video transmission system using error correction is given in Figure 2.4.

After compression, the bitstream is encoded for error correction, and then transmitted through a channel. At the receiver, data passes through the FEC decoder. The decoded sequence is then sent to the video decompressor.

The FEC decoder uses the maximum likelihood (ML) decoding algorithm based on a defined likelihood metric. The FEC decoder select the candidate that gives the largest metric [14]. For a binary symmetric channel (BSC), the largest likelihood metric is equivalent to the shortest Hamming distance.

For example, for an \((n, k)\) binary code, there are \(2^k\) distinct messages. Corresponding to the \(2^k\) possible message blocks, there are \(2^k\) \(n\)-bit codewords. The

Figure 2.4: Protecting MPEG2 using a FEC code
weight of a word is defined as the number of "1"s in the word. Let $r$ be the received $n$-bit word. Let $c$ be any valid codeword. The Hamming distance between $r$ and $c$ is defined as:

$$d_H(r, c) \triangleq w(r \oplus c)$$ \hspace{1cm} (2.1)

where $w(\cdot)$ is the to get the weight of a word, and $\oplus$ is the modulo-2 adder.

Let $C : \{c_0, c_1, \ldots c_{m-1}\}$ be the group of $m = 2^k$ valid codewords. The FEC decoder computes the Hamming distances between $r$ and $c_j$ for $j = 0, 1, \ldots, (m - 1)$ and selects the valid codeword $c_i$ that satisfies

$$w(r \oplus c_i) \leq w(r \oplus c_j)$$ \hspace{1cm} (2.2)

where $j = 1, 2, \ldots, m-1$, $j \neq i$, the Hamming distance between $c_i$ and $r$, i.e., $d_H(c_i, r)$, is the shortest $d^s$ for $r$.

In the case of a tie, (i.e., more than one $c_i$s with the same $d^s$), only one of them is selected randomly.

The minimum distance of an $(n, k)$ binary code is defined as the shortest Hamming distance between any pair of valid codewords, i.e.,

$$d_{\text{min}} = \min\{d_H(c_i, c_j) : c_i, c_j \in C, c_i \neq c_j\}$$ \hspace{1cm} (2.3)

It can be shown that the guaranteed number of correctable bit errors of this $(n, k)$ code is [14]:

$$t = \left\lfloor \frac{d_{\text{min}} - 1}{2} \right\rfloor$$ \hspace{1cm} (2.4)

where $\left\lfloor \frac{d_{\text{min}} - 1}{2} \right\rfloor$ denotes the integer part of $(d_{\text{min}} - 1)/2$.

It follows that, for a BSC with a channel bit error rate of $p$, the upper bound on the post-decoding error rate of an $n$-bit word is [11]:

22
For $p \ll 1$, it can have a rough approximation

$$P_M \leq 1 - \sum_{i=0}^{t} \binom{n}{i} p^i (1-p)^{n-i}$$

which indicates a significant improvement as $t$ increases.

### 2.3 Error concealment techniques

EC techniques make use of residual redundancy in the frequency, temporal and spatial domain after compression to conceal transmission errors. Typically the concealment problem is broken into two parts: first, detection of the area (usually a block) that is corrupted; second, concealment of that damaged area. Often error concealment schemes [4] concentrate on the second of these problems and little mention is made of the first. Generally no effort is made to ensure that the resulting image/video is consistent with the bitstream received.

Error concealment schemes can be divided into four categories. Simple Concealment, Frequency Concealment, Spatial Concealment and Temporal Concealment.

Simple concealment algorithms include simple replacement spatially or temporally adjacent blocks from the previous transmitted blocks. These schemes work best when the BER is not high and when there is not much scene change and motion from frame to frame [4].

Frequency concealment makes use of the fact that the 8*8 block DCT does not result in fully uncorrelated coefficients, especially low frequency coefficients of adjacent blocks. A typical frequency concealment algorithm [4] does linear or polynomial interpolation of adjacent DCT coefficients.
For an example of frequency concealment see Park, Kim and Lee [15]. They use the information in the compressed domain (syntax of the bitstream) to detect an error occurrence. Then a discontinuity measure at the boundary of each macroblock is calculated to determine the exact location of the corrupted block in the slice. When a corrupt block is detected, all information about that block is discarded and a concealment scheme is used that adjusts the lost DCT coefficients to minimize a boundary smoothness objective function. The resulting video is not guaranteed to be close to the transmitted video in terms of number of bits that are different between concealed bitstream and transmitted bitstream. PSNR values drop quickly for moderate BER. This scheme works in the compressed domain. Video is compressed by MPEG2 and bit errors are considered.

Chu and Leou [16] do a similar work to Park[15]. The syntax of the bitstream is used to detect transmission errors in the compressed domain. It will indicate that an error has occurred. The information in the uncompressed domain, namely spatial discontinuity, is used to determine the exact location of the corrupted block in the slice. Then the corrupted blocks are concealed using a combination of spatial and temporal concealment scheme, which switches according to a fitness function also based on spatial and temporal continuity. Video is compressed with H.261 and burst errors are considered.

Spatial concealment [17] is to interpolate directly into the spatial domain using the neighboring blocks within the same picture. Because interpolation is implemented in the spatial domain, finer details of the image can be reconstructed and consistency of the edges can be maintained to a certain degree.

Hung and Chang [18] present a spatial concealment scheme that uses the Vector Quantization (VQ) idea. A VQ codebook is generated using fairly small blocks. This is not used for transmission but only for concealment. Once a lost block is identified blocks are chosen that contain some lost pixels and some uncorrupted pixels. The closest codeword in the VQ code book (i.e. closest to the uncorrupted pixels) is selected and the lost pixels are filled in accordingly. This scheme works in the
Temporal concealment is possible for the compression algorithms where the video sequence is further compressed in the temporal domain. So a lost motion vector can be reconstructed and then the referenced block can be put into the same position. It works well with low motion picture but there is a trade-off between the performance and real-time implementation [4].

The simple, frequency, spatial and temporal concealment techniques conceal the errors by postprocessing at the source decoder. An alternative approach to combat transmission errors from the source side is by using multiple description coding (MDC). With MDC, several coded bitstreams (referred to as “descriptions”) of the same source signals are generated. The MDC encoder and decoder are designed such that the quality of the reconstructed signal is acceptable with any one description and that incremental improvement is achievable with more descriptions.

Wang, Orchard and Reibman [19] use MDC for noisy channels by pairing transform coefficients. The proposed MDC algorithm codes a pair of variables simultaneously. Instead of coding the two variables independently, they are transformed into two other variables, which are quantized and transmitted separately. In the receiver, if both transformed variables are available, they are inversely transformed to recover the original two variables. If only one transformed variable is received, it is used to estimate the original two variables. The proposed scheme can guarantee an acceptable image quality in the presence of any one description with only a small overhead over a standard JPEG coder. Their emphasis is on long burst errors.

2.4 Survey on joint FEC/EC techniques

Forward Error Correction techniques correct transmission errors based on a designed channel code structure. Error Concealment compensates the errors using source information. Generally speaking, these schemes will attempt to make output video as consistent as possible with the bits received, while factoring in information from the source video. Recently, several jointly source/channel decoding schemes using
the residual redundancy in the video together with classic error control coding have been presented. Wang and Wenger[20] gives a view of the error resilience schemes for real-time video transport over unreliable networks. They categorized these techniques into three groups: those introduced at the source and channel encoder to make the bitstream more resilient to potential errors; those invoked at the decoder upon detection of errors to conceal the effect of errors; and those which require interactions between the source encoder, decoder and channels.

The joint FEC/EC scheme can be classified by their compression scheme. Some schemes that work on VQ compressed video are briefly discussed in the following paragraphs.

In [21], a joint source/channel decoding for vector quantized compressed video using Turbo codes is presented. It aims at bit errors. When the channel decoder cannot correct all bit errors in the transmitted codevector indices, the corresponding blocks in the reconstructed image are detected by exploiting spatial contiguity in an image and the reliability information of the channel decoder. For those erroneous blocks, the reliability information of each bit is examined. The bit with lowest reliability is then changed. The source information is fed back to the Turbo decoder to improve the error correction capability.

In [22], error detection and correction of bit errors for pyramid coding are studied. One of the schemes presented here inserts one bit of parity check in the "data block" of pyramid entries to detect the errors. When errors are detected, they are corrected by bit toggling inside the corrupted portions of the bitstream. All corrections are in the compressed domain and the syntax is used to find a "feasible" solution.

Burlina and Alajaji [23] propose a joint source/channel decoding scheme which exploits statistical redundancy in the image data. They model the image data and use that along with a channel model with a maximum a posterior (MAP) detector. For the compressed video, the residual redundancy is exploited via unequal error protection and MAP detection. This scheme is for still images.
There are joint FEC/EC techniques proposed for MPEG, JPEG, H.261[24], and H.263[25] methods. These schemes can be divided into two classes: with and without feedback channel.

With feedback channel, the source encoder and decoder cooperate in the process of error concealment. When the receiver detects an error, it will notify the transmitter, over the feedback channel, which parts of the bitstream are received error-free and/or which parts could not be decompressed and have to be concealed.

Girod and Farber [26] give a review of the feedback error control techniques. H.263 is the target codec. They examine strategies that use feedback from the receiver to the transmitter along with error tracking, error confinement and reference picture selection to alleviate interframe error propagation. A more detailed error tracking scheme is given in [27]. Their scheme assumes that the video compressor is partially controlled by the channel. While this may be feasible in a point to point, interactive application, it presents difficulties in a broadcast or video server scenario. For the video server application they offer the expedient of the server storing a P frame and I frame version of each frame to be switched to as needed by the channel. This in turn introduces a mismatch in the “residue” that is DCT compressed. The mismatch error is less objectionable than channel errors. Broadcast applications or point to multipoint appear not to have been considered.

Other techniques use the forward error concealment and there is no feedback by the channel. There is little interaction between the receiver and transmitter.

Kim [28] uses syntax information to detect errors. Then extra data, namely parity-check DC coefficients, are calculated and inserted into the bitstream to locate the errors. Then a temporal concealment technique is done to retrieve the motion vector of the erroneous block. An error measure, which measures the difference between the parity-check DC coefficients in corrupted and original sequence, is calculated to evaluate which motion vector is the best. This measure is similar to the one used later in this thesis, which is stated in equation 3.1-3.4 in that it matches the boundary value of pixels. However it measures the mean absolute difference, where
this work measure the difference in DC value. Video is compressed with H.263 and burst errors are considered.

In [29], the locations of the bit errors within a bitstream are provided by a simple detection code – parity check. Then temporal/spatial concealment is used to conceal the errors. That is part of the previous work. In the other part of the previous work, the same error detection scheme is used and then the syntax of the bitstream and spatial discontinuity is used to correct/conceal these errors. These schemes will be outlined in Section 3.2.
Chapter 3

A Combined FEC and Error Concealment Technique
This chapter presents a proposed method for recovering transmission errors in MPEG2 compressed bitstreams based on information given by both FEC decoder and video decompressor. Unlike the general FEC scheme that gives only one solution, the proposed FEC decoder can provide a number of candidate solutions along with their reliability. These solutions are further screened by examining the corresponding syntax and discontinuity in an error concealment scheme.

The two kinds of source information used in this thesis, namely syntax and discontinuity are introduced in 3.1. Section 3.2 presents some previous work which is the basis of the work in this thesis. In Section 3.3, the combined FEC/EC scheme is presented and the description of the system is given along with the details concerning the receiver. At last, the evaluation of the complexity is introduced in Section 3.4.

3.1 Source information applicable to error concealment

As reviewed in Section 2.3, different kinds of source information such as spatial, temporal, etc. can be used to conceal errors. In this thesis, two kinds of source information are used to conceal errors: syntax/semantic correctness and the discontinuity of the uncompressed video. Section 3.1.1 outlines the syntax/semantic information while section 3.1.2 outlines the discontinuity information which is quantified by a discontinuity measure.

3.1.1 Syntax

MPEG2 standards are generic in the sense that the values of important parameters are specified in the syntax. Assuming that the valid MPEG2 bitstreams conform to the syntax/semantics in the standard so that the violation of the syntax/semantic is regarded as an obvious error.

The bitstream is claimed to have a syntax/semantic violation, if the incoming bits do not match any MPEG2 codeword in the VLC table, or the decompressed parameters violate the specification imposed by the coding standard [30]. Some example errors are given as follows:
1. An VLC decoding error

The MPEG2 bitstream is a concatenation of VLC and Fixed Length codes (FLC). When the MPEG2 bitstream is decompressed, both FLC and VLC should be parsed and decoded. As stated in Section 1.1, VLC is vulnerable to errors. When an VLC codeword is decoded, if it cannot be found in the corresponding codebook, there is an VLC decoding error, and the bitstream is claimed to have an syntax/semantic error.

2. Motion displacement range violates the semantics

It is specified in MPEG2 standard that for both luminance and chrominance components [1], all motion displacement pixel values should lie strictly within the picture, the size of which is defined in the sequence header. Therefore, the bitstream is claimed to have a syntax/semantic error if any of the motion displacement pixels is out of the picture boundary.

3. Slice_Start_Code is in the middle of blocks

The MPEG2 syntax reserves several start codes for synchronization purpose. As stated in Section 2.1.3, a slice is divided into macroblocks, which are composed of blocks. The Slice_Start_Code should be at the beginning of the block. If a Slice_Start_Code is detected in the middle of the block, then there is a syntax/semantic violation in this slice.

4. The number of DCT coefficients in a block exceeds 64

Since the DCT size is $8 \times 8$, the maximum number of the DCT coefficients in each block should not be greater than 64. When the end_of_block has not been encountered but the coefficient count has already reached 64, a syntax/semantic error is detected.

Because many syntax errors are not recognizable as such until the end of a slice, the decision is made on a slice-by-slice basis. All slice candidates are decompressed and checked for syntax errors and the ones that result in an MPEG syntax/semantic error are rejected. These slice candidates will result in the most egregious degradation of the video.
If no such errors exist then the probability of correctness of the slice candidate can be judged by the discontinuity of the uncompressed slice. A measure of this discontinuity is presented in the next section.

3.1.2 Discontinuity measure

The discontinuity of the pixel values occurs when there is an edge in the picture. But abrupt discontinuity is very unlikely in a natural picture. There would be some transition pixel values at the edge. For example, in Figure 3.1 there is a edge between the right leg of the person and the background.

But when the pixel values are examined in detail as shown in Figure 3.2, it is found that, instead of an abrupt discontinuity, there are some transition points.

If the differentiation of the pixel values is calculated, it can be seen that instead of an impulse (i.e., very narrow pulse), which is the differentiation of an abrupt discontinuity, there is also a transition area. That is shown in Figure 3.3.

Since the abrupt discontinuity would not occur in natural pictures, it can be detected as an error to be concealed. For this, the following discontinuity measure is introduced.
Figure 3.2: The pixel values at the edge

Figure 3.3: Differentiation of the pixel values
Among the slice candidates without syntax/semantic errors the worst degradations result from either errors in the DC values or left/right shift in the macroblocks [31]. Both these egregious errors may propagate along the slice, and then infect the subsequent frames via motion compensation. One feature of these two error types are that they cause abrupt discontinuities at slice boundaries. The discontinuity measure is used to weed out these errors.

The order of decompressing is from top to down and from left to right, and since the slice is one macroblock high and several macroblocks across, the longest slice boundary available when decompressing a slice, is the one between the current slice and the one above. Note that for the sequence used in the simulation, there are 4 slices per line. To test for the two types of degradations outlined above, let \( S_i \) denote the slice \( i \) being decoded, define \( Y_{i,j}^{top} \) as the luminance values of the top row of pixels of \( S_i \), and \( Y_{i-1,j}^{bottom} \) as the luminance values of the bottom row of pixels in the above slice. Here \( j = 1, 2, \ldots, N_Y \), and \( N_Y \) is the width of a slice in luminance pixels. See Figure 3.4 for the illustration of these values in a luminance picture. The two shaded rows of pixels are the ones of interest.
The spatial discontinuity measure is defined for the luminance component as[31]:

\[ M_Y \triangleq \frac{1}{N_Y} \sum_{j=0}^{N_Y-1} (Y_{i-4,j}^{\text{bottom}} - Y_{i,j}^{\text{top}}) \]  

(3.1)

The difference between each pixel in the top line of current slice and the bottom line of slice above is averaged. In Section 2.3 and 2.4, some error concealment schemes also make use of the discontinuity measure. But they are different to the discontinuity measure used in this thesis in that they add the absolute value of the difference instead of average. In that case, the noise will pile up and there is always a bias for the discontinuity measure, no matter there is error or not. But if the difference is averaged, the noise can be cancelled. Only when there are abrupt discontinuities at the slice boundaries, there would be a bias for the discontinuity measure. An adverse effect of this discontinuity measure is that the noise cancellation may also cancel the checkerboard blocks. Thus, the checkerboard blocks cannot be corrected. This will be suggested as future work in Chapter 5.

Similar measures can be defined for the chrominance components \( M_U \) and \( M_V \) [31].

\[ M_U \triangleq \frac{1}{N_U} \sum_{j=0}^{N_U-1} (U_{i-4,j}^{\text{bottom}} - U_{i,j}^{\text{top}}) \]  

(3.2)

\[ M_V \triangleq \frac{1}{N_V} \sum_{j=0}^{N_V-1} (V_{i-4,j}^{\text{bottom}} - V_{i,j}^{\text{top}}) \]  

(3.3)

where \( U_{i,j}^{\text{top}} \), \( V_{i,j}^{\text{top}} \) are the chrominance values of the top row of pixels in \( S_i \), \( U_{i-4,j}^{\text{bottom}} \), \( V_{i-4,j}^{\text{bottom}} \) are the chrominance values of the bottom row of pixel in the slice above \( S_i \), and \( N_U \), \( N_V \) are the width of the slice in chrominance pixels.

What is of interest is the value of the bias instead of the direction, so the overall
discontinuity measure is defined as:

\[ M \triangleq (\lambda \times M_Y^2 + M_U^2 + M_V^2)^{1/2} \]  

(3.4)

where \( \lambda \) is a weight coefficient. Since the chrominance component is subsampled by a factor of two horizontally, i.e., \( N_Y = 2N_U = 2N_V \), when the discontinuity measure is divided by \( \frac{1}{N_Y} \) or \( \frac{1}{N_U} \) and then squared, the coefficient divided by \( M_Y^2 \) is 4 times over that divided by \( M_U^2 \) or \( M_V^2 \). Thus \( \lambda = 4 \) is an appropriate value and is used in the remainder of this thesis.

### 3.2 Error concealment using syntax information

Classical temporal and spatial concealments (see Section 2.3) work in the uncompressed domain and try to have a "smooth" image [31]. In the previous work, the errors are corrected in the compressed domain, in which the primary criteria is a bitstream without syntax/semantic errors. As a refinement a discontinuity measure calculated from the uncompressed domain is added to select among slice candidates without syntax violation.

#### 3.2.1 Syntax Based Error Concealment (SBEC)

In [32], SBEC is presented and its block diagram is shown in Figure 3.5. (The dashed line of "uncompressed video" is present only in the scheme described in section 3.2.2, not in SBEC). To generate the slice candidates, for every block of 12 compressed data bits, a parity check bit is appended. The combined 13 bit block of data and parity is an Error Concealment (EC) block. At the error detection decoder, the received EC block is checked to determine whether or not the parity is satisfied. If the parity is not satisfied, the EC block will be marked as in error. All odd number of errors can be detected in this way. For each flagged erroneous EC block, the slice which contains it will be partially decompressed up to 13 times. In each partial decompression, a different bit in the EC block is toggled until no syntax
violation is found. Only partial decompressions are needed since the video itself is not constructed.

### 3.2.2 Syntax and Discontinuity Based Error Concealment (SDBEC)

SDBEC [31] is based on SBEC and is a refinement of SBEC. It works as SBEC but does a full decompression of the bitstream instead. To generate the slice candidates, the same parity check scheme as that of SBEC is needed. For those slice candidates that do not have a syntax violation, the discontinuity measure is calculated to measure their goodness. All togglings are examined and the one with no syntax violation and the best measure is taken as correct. SDBEC is shown in Figure 3.5 and the dashed line is present.

The scheme proposed in this thesis builds on the work presented above. It is presented in the next section.

### 3.3 Proposed scheme

SDBEC [31] mentioned in section 3.2.2 makes use of the syntax information and
discontinuity measure to correct errors. The proposed scheme does not introduce a new EC algorithm, but uses existing methods [31, 32]. The close coupling of FEC and EC is the main contribution of the proposed scheme. The expansion of the number of slice candidates and the use of FEC reliability information is the main difference between the proposed scheme and SDBEC.

The block diagram of the proposed scheme is illustrated in Figure 3.6. After compression, the bitstream is encoded for error correction, and then transmitted through a channel. At the receiver, data pass through the FEC decoder. Unlike the classic FEC approach, the FEC decoder gives more than one slice candidates along with their reliability information to the syntax checker where syntax/semantic errors are checked for. One decompression for each slice candidate takes place. The slice candidates with syntax/semantic errors are rejected and only those without syntax/semantic violations are sent to the discontinuity measure selector. The discontinuity measure of each slice candidate is calculated and then combined with the Hamming distance to get a modified discontinuity measure. The slice candidate with the best modified discontinuity measure is taken as the correct one.

In the following sub-sections, each part of the receiver is described in detail.
3.3.1 Multiple-Candidate Likelihood (MCL) decoder

Prior to the descriptions of the proposed MCL decoder, it is worth reviewing the principle of a conventional Maximum Likelihood (ML) decoder.

Consider a designed code $C$ containing $m$ valid codewords, $c_j$ for $j = 0, 1, \ldots, (m - 1)$. For a received word $r$, the likelihood metric between $r$ and $c_j$, $l(r, c_j)$ is calculated. An ML decoder will give an output valid codeword $c_i$ that satisfies

$$l(r, c_i) \geq l(r, c_j)$$

(3.5)

for $j = 0, 1, \ldots, (m - 1)$ and $j \neq i$.

For an additive white Gaussian noise (AWGN) channel, the likelihood metric is:

$$l(r, c_j) = -d_E(r, c_j)$$

(3.6)

where $d_E(r, c_j)$ is the Euclidean distance between $r$ and $c_j$.

For a binary symmetric channel (BSC), the likelihood metric is:

$$l(r, c_j) = -d_H(r, c_j)$$

(3.7)

where $d_H(r, c_j)$ is the Hamming distance between $r$ and $c_j$.

From the above relations, it can be seen that the ML decoder will select an output $c_i$ such that

$$d(r, c_i) \leq d(r, c_j)$$

(3.8)

where

$$d(r, c_j) = \begin{cases} d_E(r, c_j) & \text{in an AWGN channel} \\ d_H(r, c_j) & \text{in a BSC} \end{cases}$$

for $j = 0, 1, 2, \ldots, (m - 1)$ and $j \neq i$. 
It can also be seen that, for the case of more than one valid codewords that satisfy the ML criterion, the conventional ML decoder will randomly select one solution.

In the proposed Multiple-Candidate Likelihood (MCL) decoder, rather than selecting one solution, it considers a set:

\[ B_0 = \{ c_i \mid d(r, c_i) = d^s \leq d(r, c_j), c_j \in C \} \]  

(3.9)

e.g., \( B_0 \) is the set of candidate codewords \( c_j \)s with the shortest distance \( d^s \).

In a general case, an extension to a set of valid codewords is considered:

\[ B_k = \{ c_k \mid d(r, c_k) = d^s + a, c_k \in C \} \]  

(3.10)

for a given offset “\( a \)”. \( B_a \) is called the set of “shortest-distance-plus-a” candidates.

The FEC decoding is processed on the unit of a received word with length equal to that of the valid codeword. On the other hand, the decompression (including both syntax/semantic check and discontinuity measure calculation) operates on the slice-by-slice basis. Depending on the selected code length, a received slice, \( S \), is considered as a concatenation of the message portions \( x_i \) of \( L \) received words \( r_i \),

\[ S = x_1 | x_2 | \ldots | x_L \]  

(3.11)

where | denotes concatenation.

For each message portion \( x_i \) of received word \( r_i \), there may have more than one candidates produced by the MCL decoder. Let \( c_i \) denote a codeword candidate and \( y_i \) denote its message portion. The distance between \( r_i \) and \( c_i \) is:

\[ d_i \triangleq d(r_i, c_i) \]  

(3.12)

As a result, there are more than one slice candidates. Let \( S \) denote a slice candidate:
The distance between the received slice $S$ and the slice candidate $\mathcal{S}$ is defined as:

$$ \mathcal{S} = y_1 | y_2 | \ldots | y_L $$

$$ D(S, \mathcal{S}) = \sum_{i=1}^{L} d_i $$

(3.13)

(3.14)

It follows that if all $c_i | s$, $i = 1, 2, \ldots, L$ are selected from sets $B_0 | s$ with the shortest distance $d_i^s$, then the slice candidates $\mathcal{S}$ belongs to $G_0$, the group of slice candidates with the shortest distance

$$ D^s = \sum_{i=1}^{L} d_i^s $$

(3.15)

In general, $G_a$, the group of all slice candidates with distance $(D^s + a)$ for the received slice $S$, is defined as:

$$ G_a \triangleq \{ \mathcal{S} | d(S, \mathcal{S}) = D^s + a \} $$

(3.16)

For convenience, define "search space" as the range of slice candidates to be checked by the syntax checker. It may contain $G_0$ or $G_0$ and $G_a$.

For example, if only one codeword candidate $c_i$ belongs to the set of shortest-distance-plus-a candidates, $B_a$, and the $(L - 1)$ remaining codeword candidates belong to the sets $B_0$ with the shortest distances, then the corresponding slice candidate $\mathcal{S}$ belongs to $G_a$.

3.3.2 Benefits of MCL decoding

If only candidate with the shortest distance $D^s$ is selected, then the ML criterion is achieved. From the channel coding point of view, all slice candidates with the
shortest distance are equally good. Therefore, without further consideration, an arbitrary selection of one solution from these candidates yields the same post-decoding error correction performance. This is exactly what is done by the traditional ML decoder[14].

However, if further consideration for error resilience using other measures such as in this application of error concealment based on the residual redundancy of the digital video signal, then the traditional ML decoder prevents the chance to examine other equally good candidates. In this sense, the proposed MCL decoding strategy with all shortest-distance slice candidates will definitely enhance the performance because all equally good slice candidates (from the channel coding viewpoint) are further considered using additional metrics based on the digital video signal properties.

At a low BER the number of slice candidates with the shortest distance is mostly 1. In this case, the MCL decoder with shortest distance is equivalent to the traditional ML decoder with one solution. The performance of the proposed scheme is expected to be the same as that of the traditional FEC scheme. At a high BER, the number of slice candidates with shortest distance increases and the proposed scheme is expected to outperform the traditional FEC scheme.

Figure 3.7 shows the simulated number of slice candidates in $G_0$ for each slice of the sequence “Table Tennis” at BER of $1.2 \times 10^{-2}$ using a $(16,8)$ quasi-cyclic code.

Though for most slices, there are only one slice candidate in $G_0$, there are about 5% of the total 21600 slices that have more than one slice candidates in $G_0$.

By allowing slice candidates with a shortest distance plus $a$ where $a > 0$ to be considered in further error concealment, more trust is essentially put on the further error concealment. As “$a$” increases, the number of slice candidates is increased. However, by allowing a larger value of “$a$”, the “trust” on the channel decoder is reduced intentionally. This may degrade the performance of the proposed scheme.
3.3.3 Size of slice candidate groups

Consider all the slice candidates in $G_0$. Let $N_i$ be the number of possible slice candidates in $G_i$. For these slice candidates each of their constituent message portions gets a message portion candidate from $B_0$. Given a received slice $S = x_1|x_2|......|x_L$, suppose there are $f_0(x_i)$ message portion candidates in $B_0$ for $x_i$, then $N_0$ is

$$N_0 = \prod_{i=1}^{L} f_0(x_i)$$  \hspace{1cm} (3.17)

If $G_1$ is considered, to generate a slice candidate, any one message gets one message portion candidate from $B_1$. Suppose there are $f_1(x_i)$ candidates in $B_1$ for $x_i$, then $N_1$ is:

$$N_1 = \sum_{j=1}^{L} \left[ f_1(x_j) \prod_{i=1, i \neq j}^{L} f_0(x_i) \right]$$  \hspace{1cm} (3.18)

When $G_2$ is considered, to generate a slice candidate, there are two possibilities: any two message portions get message portion candidates from $B_1$ or any one message portion gets the message portion candidate from $B_2$. Suppose there are $f_2(x_i)$
message portion candidates in $B_2$ for $x_i$, then $N_2$ is:

$$N_2 = \sum_{j=1}^{L} [f_2(x_j) \prod_{i=1, i \neq j}^{L} f_0(x_i)] + \sum_{i=1}^{L-1} \sum_{j=i+1}^{L} [f_1(x_i) f_1(x_j) \prod_{k=1, k \neq i, j}^{L} f_0(x_k)] \quad (3.19)$$

For $(a + 1)$ groups, $G_0, G_1, ..., G_a$, given by the MCL decoder, there are

$$N = \sum_{i=0}^{a} N_i \quad (3.20)$$

slice candidates to the syntax checker.

### 3.3.4 Syntax checker

As discussed in Section 3.1.1, syntax/semantic based redundant information can be used to conceal bit error occurrence in the compressed bitstream. The objective of the syntax checker is to check the slice candidates given by the channel decoder for syntax/semantic errors before they are sent to be evaluated by the discontinuity measure.

For the $N$ slice candidates given by the MCL decoder, the syntax checker fully decompresses each slice candidate and checks for syntax/semantic errors. Because many errors are not recognizable as such until the end of the slice, the decisions are made on a slice by slice basis. The slice candidates that result in an MPEG syntax/semantic error are rejected and only those slice candidates which have no syntax/semantic violation are sent to the discontinuity measure selector to calculate their discontinuity measures. The "no syntax/semantic violation" condition occurs when the decompression arrives at the end of the current slice without finding a syntax/semantic error.

### 3.3.5 Discontinuity measure selector

After syntax checker, the outright syntax violation has been removed by the syntax checker. The slice candidates passed from the syntax checker are then sent to the discontinuity measure selector.
The discontinuity measure as defined in Section 3.1.2 indicates the change in pixel values as compared to neighbor slices. Since a large change unlikely occurs, a slice candidate with the smallest value of discontinuity measure is considered as the best choice.

When slice candidates in the same group \((G_0, G_1, \text{ or } G_2)\) are considered, discontinuity measure is used to evaluate their "picture" goodness. But if slice candidates in different groups are evaluated, their discontinuity measure cannot be treated equally because these slice candidates have different performance in view of channel coding. This factor must also be taken into consideration as follows.

The distance of a slice candidate from the received slice is an evaluation of how likely that slice candidate is, in view of channel coding, e.g., a slice candidate \(S_0 \in G_0\) has a better performance in terms of channel coding than a slice candidate \(S_1 \in G_1\). When they are further considered in terms of discontinuity measure, this should be included. For this, a modified discontinuity measure for a slice candidate \(S_i\) belonging to group \(G_a\) is introduced:

\[
M_i \triangleq M_i + \alpha_a
\]  

where \(M_i\) is the discontinuity measure of slice \(S_i\), \(\alpha_a\) is a constant offset. Since a slice candidate \(S_i\) belonging to group \(G_a\) has a better performance than a slice candidate \(S_j\) of group \(G_b\) where \(a < b\), it is logical to select \(\alpha_a < \alpha_b\). For \(a = 0\) (i.e., slice with the shortest distance), \(\alpha_0 = 0\) is selected.

The objective of \(\alpha_a\) is therefore to balance between candidates in \(G_0\) and those in \(G_a\) so that the PSNR is maximized. The search for \(\alpha_a\) will be presented in Section 4.2.

3.3.6 Video Decompressor

The decompressor module have the ability to perform in the presence of errors. Also it can pass the uncompressed video to the syntax checker and discontinuity measure
selector. The MPEG2 decompressor V1.2a is used in this work. The diagram of a video compressor is shown in Figure 2.2.

3.4 Complexity of the proposed scheme

The complexity of the scheme depends on two factors:

- The complexity of the FEC decoder
- The complexity of the syntax checker and discontinuity measure selector

Complexity of the FEC decoder depends on the selected code. Complexity of the syntax checker and discontinuity measure selector is mainly related to two processes:

- Slice-by-slice decompression of the MPEG2 sequence
- Discontinuity measure calculation for each slice

Discontinuity measure calculation is relatively simple. The major contributor to the complexity of the proposed scheme is in the repeated decompression of a number of slice candidates. Therefore, the order of complexity can be represented by:

(i) Number of slice candidates to be decompressed
(ii) Number of bits extracted from the bitstream

3.4.1 Number of slice candidates to be decompressed

The number of slice candidates to be decompressed can be estimated as follows. Consider a (16,8) quasi-cyclic code with \(d_{\text{min}} = 5\) defined in Appendix A.

The \(2^{16} = 65536\) possible received words are split into classes according to how far each is away from the closest valid codeword. Specifically, Class \(i\) contains all received words that are \(i\) away from the closest valid codeword. Thus Class 0 contains the valid codewords since they are 0 away from themselves, and there are \(2^8 = 256\) members in Class 0. For the minimum distance of 5, the numbers of members in Classes 1, 2 can also be calculated. For each valid codeword, there
Table 3.1: The distance profile $\gamma(\tau, j)$ for the received words, with probability calculated for BER of $1.2 \times 10^{-2}$

<table>
<thead>
<tr>
<th>Type</th>
<th>Class</th>
<th>$\pi_F$</th>
<th>Distance Profile $\gamma(\tau, j)$ for $j=0,1, \ldots, 16$</th>
<th>Probability $P_r$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$\begin{bmatrix} 256 &amp; 0 &amp; 0 &amp; 24 &amp; 44 &amp; 40 &amp; 49 &amp; 40 &amp; 38 &amp; 34 &amp; 36 &amp; 35 &amp; 36 &amp; 38 &amp; 36 &amp; 36 &amp; 38 &amp; 36 &amp; 36 \end{bmatrix}$</td>
<td>$0.824349014819$</td>
</tr>
<tr>
<td>1a</td>
<td>1</td>
<td>0</td>
<td>$\begin{bmatrix} 2048 &amp; 0 &amp; 1 &amp; 0 &amp; 0 &amp; 16 &amp; 32 &amp; 52 &amp; 48 &amp; 37 &amp; 32 &amp; 22 &amp; 8 &amp; 2 &amp; 0 &amp; 0 &amp; 0 \end{bmatrix}$</td>
<td>$0.00009986486209$</td>
</tr>
<tr>
<td>1b</td>
<td>1</td>
<td>0</td>
<td>$\begin{bmatrix} 2048 &amp; 0 &amp; 1 &amp; 0 &amp; 0 &amp; 7 &amp; 17 &amp; 36 &amp; 48 &amp; 42 &amp; 43 &amp; 35 &amp; 16 &amp; 7 &amp; 3 &amp; 0 &amp; 0 &amp; 0 \end{bmatrix}$</td>
<td>$0.00009728270566$</td>
</tr>
<tr>
<td>2a</td>
<td>2</td>
<td>0</td>
<td>$\begin{bmatrix} 2048 &amp; 0 &amp; 0 &amp; 1 &amp; 3 &amp; 4 &amp; 15 &amp; 38 &amp; 46 &amp; 46 &amp; 33 &amp; 15 &amp; 8 &amp; 3 &amp; 0 &amp; 0 &amp; 0 \end{bmatrix}$</td>
<td>$0.0010089078913$</td>
</tr>
<tr>
<td>2b</td>
<td>2</td>
<td>0</td>
<td>$\begin{bmatrix} 8120 &amp; 0 &amp; 0 &amp; 1 &amp; 2 &amp; 5 &amp; 18 &amp; 35 &amp; 44 &amp; 44 &amp; 35 &amp; 18 &amp; 5 &amp; 2 &amp; 1 &amp; 0 &amp; 0 \end{bmatrix}$</td>
<td>$0.00246312111376$</td>
</tr>
<tr>
<td>2c</td>
<td>2</td>
<td>0</td>
<td>$\begin{bmatrix} 4096 &amp; 0 &amp; 0 &amp; 1 &amp; 2 &amp; 6 &amp; 17 &amp; 31 &amp; 48 &amp; 52 &amp; 38 &amp; 51 &amp; 22 &amp; 6 &amp; 1 &amp; 1 &amp; 0 \end{bmatrix}$</td>
<td>$0.0019476428500$</td>
</tr>
<tr>
<td>2d</td>
<td>2</td>
<td>0</td>
<td>$\begin{bmatrix} 3120 &amp; 0 &amp; 0 &amp; 1 &amp; 2 &amp; 6 &amp; 18 &amp; 30 &amp; 44 &amp; 56 &amp; 44 &amp; 25 &amp; 18 &amp; 10 &amp; 2 &amp; 0 &amp; 0 \end{bmatrix}$</td>
<td>$0.00026395965853$</td>
</tr>
<tr>
<td>2e</td>
<td>2</td>
<td>0</td>
<td>$\begin{bmatrix} 4096 &amp; 0 &amp; 0 &amp; 1 &amp; 2 &amp; 5 &amp; 19 &amp; 34 &amp; 40 &amp; 50 &amp; 50 &amp; 29 &amp; 14 &amp; 9 &amp; 3 &amp; 0 &amp; 0 \end{bmatrix}$</td>
<td>$0.00194654394874$</td>
</tr>
<tr>
<td>2f</td>
<td>2</td>
<td>0</td>
<td>$\begin{bmatrix} 4096 &amp; 0 &amp; 0 &amp; 1 &amp; 2 &amp; 6 &amp; 18 &amp; 35 &amp; 44 &amp; 44 &amp; 35 &amp; 18 &amp; 5 &amp; 2 &amp; 1 &amp; 0 \end{bmatrix}$</td>
<td>$0.0019476428500$</td>
</tr>
<tr>
<td>2g</td>
<td>2</td>
<td>0</td>
<td>$\begin{bmatrix} 2048 &amp; 0 &amp; 0 &amp; 1 &amp; 1 &amp; 7 &amp; 21 &amp; 27 &amp; 42 &amp; 58 &amp; 42 &amp; 37 &amp; 21 &amp; 7 &amp; 1 &amp; 1 &amp; 0 \end{bmatrix}$</td>
<td>$0.8571644810-04$</td>
</tr>
<tr>
<td>2h</td>
<td>2</td>
<td>0</td>
<td>$\begin{bmatrix} 4096 &amp; 0 &amp; 0 &amp; 0 &amp; 2 &amp; 9 &amp; 19 &amp; 29 &amp; 40 &amp; 50 &amp; 50 &amp; 24 &amp; 14 &amp; 6 &amp; 3 &amp; 1 &amp; 0 \end{bmatrix}$</td>
<td>$4.9915146030-05$</td>
</tr>
<tr>
<td>3a</td>
<td>3</td>
<td>0</td>
<td>$\begin{bmatrix} 6144 &amp; 0 &amp; 0 &amp; 0 &amp; 3 &amp; 9 &amp; 15 &amp; 28 &amp; 46 &amp; 54 &amp; 46 &amp; 28 &amp; 15 &amp; 9 &amp; 3 &amp; 0 &amp; 0 \end{bmatrix}$</td>
<td>$1.1030003050-04$</td>
</tr>
<tr>
<td>3b</td>
<td>3</td>
<td>0</td>
<td>$\begin{bmatrix} 4096 &amp; 0 &amp; 0 &amp; 0 &amp; 3 &amp; 9 &amp; 14 &amp; 29 &amp; 50 &amp; 50 &amp; 40 &amp; 34 &amp; 19 &amp; 5 &amp; 3 &amp; 1 &amp; 0 \end{bmatrix}$</td>
<td>$7.525991135-05$</td>
</tr>
<tr>
<td>3c</td>
<td>3</td>
<td>0</td>
<td>$\begin{bmatrix} 5120 &amp; 0 &amp; 0 &amp; 0 &amp; 2 &amp; 10 &amp; 18 &amp; 35 &amp; 44 &amp; 55 &amp; 44 &amp; 30 &amp; 18 &amp; 6 &amp; 3 &amp; 1 &amp; 0 \end{bmatrix}$</td>
<td>$6.278154825-05$</td>
</tr>
<tr>
<td>3d</td>
<td>3</td>
<td>0</td>
<td>$\begin{bmatrix} 3048 &amp; 0 &amp; 0 &amp; 0 &amp; 3 &amp; 7 &amp; 16 &amp; 36 &amp; 43 &amp; 42 &amp; 48 &amp; 36 &amp; 17 &amp; 7 &amp; 0 &amp; 0 &amp; 1 \end{bmatrix}$</td>
<td>$3.648155792-05$</td>
</tr>
<tr>
<td>3e</td>
<td>3</td>
<td>0</td>
<td>$\begin{bmatrix} 2048 &amp; 0 &amp; 0 &amp; 0 &amp; 4 &amp; 8 &amp; 11 &amp; 22 &amp; 32 &amp; 52 &amp; 48 &amp; 42 &amp; 32 &amp; 18 &amp; 8 &amp; 3 &amp; 0 \end{bmatrix}$</td>
<td>$4.843237113-05$</td>
</tr>
<tr>
<td>3f</td>
<td>3</td>
<td>0</td>
<td>$\begin{bmatrix} 4096 &amp; 0 &amp; 0 &amp; 0 &amp; 3 &amp; 8 &amp; 15 &amp; 33 &amp; 46 &amp; 44 &amp; 46 &amp; 38 &amp; 15 &amp; 4 &amp; 3 &amp; 1 &amp; 0 \end{bmatrix}$</td>
<td>$5.662356732-05$</td>
</tr>
<tr>
<td>3g</td>
<td>3</td>
<td>0</td>
<td>$\begin{bmatrix} 512 &amp; 0 &amp; 0 &amp; 0 &amp; 4 &amp; 7 &amp; 12 &amp; 26 &amp; 48 &amp; 42 &amp; 48 &amp; 36 &amp; 12 &amp; 7 &amp; 4 &amp; 0 &amp; 0 \end{bmatrix}$</td>
<td>$1.202767053-05$</td>
</tr>
<tr>
<td>3h</td>
<td>3</td>
<td>0</td>
<td>$\begin{bmatrix} 2048 &amp; 0 &amp; 0 &amp; 0 &amp; 3 &amp; 8 &amp; 16 &amp; 32 &amp; 42 &amp; 48 &amp; 52 &amp; 32 &amp; 11 &amp; 8 &amp; 4 &amp; 0 &amp; 0 \end{bmatrix}$</td>
<td>$3.6424688237-05$</td>
</tr>
<tr>
<td>3i</td>
<td>3</td>
<td>0</td>
<td>$\begin{bmatrix} 2048 &amp; 0 &amp; 0 &amp; 0 &amp; 2 &amp; 8 &amp; 20 &amp; 32 &amp; 42 &amp; 48 &amp; 52 &amp; 32 &amp; 16 &amp; 8 &amp; 0 &amp; 0 &amp; 1 \end{bmatrix}$</td>
<td>$2.481855519-05$</td>
</tr>
</tbody>
</table>

are $\binom{16}{1} = 16$ received words that are 1 away from it. So there are total $16 \times 256 = 4096$ members in Class 1. Similarly, the number of received words in Class 2 is $\binom{16}{2} \times 256 = 30720$. For Class 3 there may be one received word that are 3 away from two different valid codewords, so the number of received words after Class 2 cannot be predicted. For the (16,8) quasi-cyclic code in use, there are no received words in Class 5 or higher, as expected for a well designed code.

What is of interest is not only how far a certain received word is to the closest valid codeword but also how far it is to each of the 256 valid codewords. This is done for each of the 65536 possible received words and 22 distinct types is found as shown in Table 3.1, in which each row represents one type.
Assume the valid codewords are transmitted with equal probability over a BSC with a given BER. The probability of each received word in one type can be calculated as shown in the final column of Table 3.1.

Specifically this probability is calculated as follows. Let $BER = p$, $q = 1 - p$. Let $n_\tau$ denote the number of received words in type $\tau$, and $\gamma(\tau, j)$ be the number of valid codewords that are $j$ bits away from the received word in type $\tau$, i.e., the element $(\tau, j)$ in Table 3.1. The probability of each received word in type $\tau$ is:

$$P_\tau = \frac{n_\tau}{256} \sum_{j=0}^{16} p^j (1 - p)^{16-j} \gamma(\tau, j) \quad (3.22)$$

Given a BER of $1.2 \times 10^{-2}$, the probability for each type is shown in the last column of Table 3.1. The matlab source code to derive this table is given in Appendix B.

The number of slice candidates to be searched in $G_0$ is given by Equation 3.17. Assuming independently transmitted codewords and independent errors, the mean value of $N_0$ is:

$$E\{N_0\} = \prod_{i=1}^{L} E\{f_0(x_i)\} \quad (3.23)$$

where $E\{\}$ is the expectation operator. Since $x_i$ are identically distributed,

$$E\{N_0\} = [E\{f_0(x_i)\}]^L \quad (3.24)$$

Taking the expectation of $N_1$ and $N_2$ using their derivatin in Equation 3.18 and 3.19,

$$E\{N_1\} = E\left\{ \sum_{j=1}^{L} f_1(x_j) \prod_{i=1, i \neq j}^{L} f_0(x_i) \right\}$$

$$= LE\{f_1(x_i)\} [E\{f_0(x_i)\}]^{L-1} \quad (3.25)$$
The calculation of the variance for $N_0$, $N_1$, and $N_2$ is given in Appendix C. Here only the results are given.

\[\begin{align*}
E\{N_2\} &= \sum_{j=1}^{L} E\{f_2(x_j) \prod_{i=1, i \neq j}^{L} f_0(x_i)\} \\
&\quad + \sum_{i=1}^{L-1} \sum_{j=i+1}^{L} E\{f_1(x_i)f_1(x_j) \prod_{k=1, k \neq i,j}^{L} f_0(x_k)\} \\
&= LE\{f_2(x_i)\}(E\{f_0(x_i)\})^{L-1} \\
&\quad + \frac{L(L-1)}{2}[E\{f_1(x_i)\}]^2 \times [E\{f_0(x_i)\}]^{L-2} \\
&= (3.26)
\end{align*}\]

\[\begin{align*}
\text{var}\{N_0\} &= E\{N_0^2\} - E^2\{N_0\} = [E\{f_0^2(x_i)\}]^L - [E\{f_0(x_i)\}]^{2L} \\
&= (3.27)
\end{align*}\]

\[\begin{align*}
\text{var}\{N_1\} &= E\{N_1^2\} - E^2\{N_1\} \\
&= (3.28)
\end{align*}\]

where

\[\begin{align*}
E\{N_1^2\} &= LE\{f_1^2(x_j)\}[E\{f_0^2(x_i)\}]^{L-1} \\
&\quad + (L^2 - L)[E\{f_1(x_j)f_0(x_j)\}]^2[E\{f_0(x_i)\}]^{L-2} \\
&= (3.29)
\end{align*}\]

\[\begin{align*}
\text{var}\{N_2\} &= E\{N_2^2\} - E^2\{N_2\} \\
&= (3.30)
\end{align*}\]
where

\[
E\{N_2^2\} = LE\{f_2^2(x_j)\}[E\{f_0^2(x_i)\}]^{L-1} + (L^2 - L)[E\{f_2(x_j) f_0(x_j)\}]^2[E\{f_0^2(x_i)\}]^{L-2} + \frac{(L^2 - L)}{2}[E\{f_1^2(x_j)\}]^2[E\{f_0^2(x_i)\}]^{L-3} + \frac{L(L - 1)(L - 2)}{4}[E\{f_1(x_j) f_0(x_i)\}]^4[E\{f_0^2(x_i)\}]^{L-4} + 2(L^2 - L)[E\{f_1(x_i) f_2(x_i)\} E\{f_1(x_i) f_0(x_i)\}] [E\{f_0^2(x_i)\}]^{L-2} + L(L - 1)(L - 2)[E\{f_1(x_i) f_0(x_i)\}]^2[E\{f_2(x_i) f_0(x_i)\}[E\{f_0^2(x_i)\}]^{L-3}
\]

(3.31)

The values of \(E\{f_a(x_i)\}\) are calculated as follows. Consider the distribution of distance profile \(\gamma(\tau, j)\). For each type \(\tau\), the element \(\gamma(\tau, j)\) in row \(\tau\) indicates the number of valid codewords that are \(j\) bits away from a received word of type \(\tau\). Recall that a received word of type \(\tau\) is \(i\) bits away from a valid codeword where \(i\) is the class number corresponding to type \(\tau\). Therefore, the first non-zero element in row \(\tau\) appearing at column \(J\), \(\gamma(\tau, J)\), represents the number of candidates (i.e., valid codewords) with the shortest distance from the received word of type \(\tau\). The element in row \(\tau\) at column \((J + a)\), \(\gamma(\tau, J + a)\), indicates the number of candidates with the shortest distance plus \(a\) from the received word of type \(\tau\) (i.e., belonging to set \(B_a\)). It follows that

\[
E\{f_a(x_i)\} = \sum_{\text{all types } \tau} \gamma(\tau, J + a) \cdot P_\tau
\]

(3.32)

Using the value in Table 3.1 for the \((16, 8)\) quasi-cyclic code and \(p = 1.2 \times 10^{-2}\), the calculations for \(E\{f_a(x_i)\}\) are summarized in Table 3.2.

To estimate the confidence limits of the number of slice candidates in \(G_0\), change
<table>
<thead>
<tr>
<th>$\tau$</th>
<th>$J$</th>
<th>$\gamma(\tau, J)$</th>
<th>$\gamma(\tau, J+1)$</th>
<th>$\gamma(\tau, J+2)$</th>
<th>$P_\tau$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0.82434900147619</td>
</tr>
<tr>
<td>1a</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0.08009986458629</td>
</tr>
<tr>
<td>1b</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0.08009972289769</td>
</tr>
<tr>
<td>2a</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td>0.00100890786913</td>
</tr>
<tr>
<td>2b</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>0.00249310113476</td>
</tr>
<tr>
<td>2c</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>6</td>
<td>0.00199476428500</td>
</tr>
<tr>
<td>2d</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>6</td>
<td>0.00249345965853</td>
</tr>
<tr>
<td>2e</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>0.00199448434964</td>
</tr>
<tr>
<td>2f</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>0.0020180991544</td>
</tr>
<tr>
<td>2g</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>6</td>
<td>0.00197114951919</td>
</tr>
<tr>
<td>2h</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>7</td>
<td>9.8571644819e-04</td>
</tr>
<tr>
<td>3a</td>
<td>3</td>
<td>2</td>
<td>9</td>
<td>19</td>
<td>4.9915146030e-05</td>
</tr>
<tr>
<td>3b</td>
<td>3</td>
<td>3</td>
<td>9</td>
<td>15</td>
<td>1.1030003050e-04</td>
</tr>
<tr>
<td>3c</td>
<td>3</td>
<td>3</td>
<td>9</td>
<td>14</td>
<td>7.3529911835e-05</td>
</tr>
<tr>
<td>3d</td>
<td>3</td>
<td>2</td>
<td>10</td>
<td>18</td>
<td>6.2748154025e-05</td>
</tr>
<tr>
<td>3e</td>
<td>3</td>
<td>3</td>
<td>7</td>
<td>16</td>
<td>3.6481557826e-05</td>
</tr>
<tr>
<td>3f</td>
<td>3</td>
<td>4</td>
<td>8</td>
<td>11</td>
<td>4.8432371139e-05</td>
</tr>
<tr>
<td>3g</td>
<td>3</td>
<td>3</td>
<td>8</td>
<td>15</td>
<td>3.6623267323e-05</td>
</tr>
<tr>
<td>3h</td>
<td>3</td>
<td>4</td>
<td>7</td>
<td>12</td>
<td>1.2072670636e-05</td>
</tr>
<tr>
<td>3i</td>
<td>3</td>
<td>3</td>
<td>8</td>
<td>16</td>
<td>3.6624988237e-05</td>
</tr>
<tr>
<td>3j</td>
<td>3</td>
<td>2</td>
<td>8</td>
<td>20</td>
<td>2.4815863519e-05</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>10</td>
<td>24</td>
<td>28</td>
<td>1.8469887415e-07</td>
</tr>
</tbody>
</table>

Table 3.2: Values of $E(f_a(x_i))$ for $p = 1.2 \times 10^{-2}$ and (16,8) quasi-cyclic code
$N_0$ to log scale,

$$\log N_0 = \sum_{i=1}^{L} \log f_0(x_i) \quad (3.33)$$

From the above assumptions, $f_0(x_i)$ are discrete independent random variables with identical distribution, and $\log f_0(x_i)$ defines a one-to-one transformation between the values of $f_0(x_i)$ and $\log f_0(x_i)$, thus $\log f_0(x_i)$ are identical distributed independent random variables with distributions derived from Table 3.2. Consequently, the expectation of $\log N_0$ can be calculated as:

$$E\{\log N_0\} = \sum_{i=1}^{L} E\{\log f_0(x_i)\} = LE\{\log f_0(x_i)\} \quad (3.34)$$

From the central limit theorem[11], the sum of statistically independent and identically distributed random variables with finite mean and variance approaches a Gaussian cdf as $L \to \infty$. In practice, when $L > 30$, the distribution of the sum can be looked as Gaussian. So $\log N_0$ approaches a Gaussian cdf when $L > 30$.

From Table in [33], the 90% confidence interval of $\log N_0$ is

$$LE\{\log f_0(x_i)\} - 1.645\sqrt{L}var\{\log f_0(x_i)\} < \log N_0 < LE\{\log f_0(x_i)\} + 1.645\sqrt{L}var\{\log f_0(x_i)\} \quad (3.35)$$

Use the Chebyshev’s theorem [33]

$$P(\mu - k\sigma < X < \mu - k\sigma) \geq 1 - \frac{1}{k^2} \quad (3.36)$$

if $1 - \frac{1}{k^2} = 90\%$, then the range of $N_0$ is:

$$E\{N_0\} - \sqrt{10}var\{N_0\} < N_0 < E\{N_0\} + \sqrt{10}var\{N_0\} \quad (3.37)$$

These two limits for 90% confidence interval for different length is plotted in Figure
3.8. Since $N_0$, $N_1$ and $N_2$ are larger than 0, when the limits are less than zero, they are set to zero.

Figure 3.8 shows that the Chebyshev confidence limit is looser than the central limit confidence limit.

Because of the nature of the equations for $N_1$ and $N_2$ only the Chebyshev confidence limit can be calculated. The curves are shown in Figures 3.9, 3.10, respectively:

**3.4.2 Number of bits extracted from the bitstream**

Since all slices do not have the same length, and the slice candidates that cause syntax violation are rejected and are not completely decompressed, a more exact measure of complexity than the number of slice candidates to be decompressed may be formulated.

Such a measure is a running number of bits extracted from the bitstream. If a slice candidate is fully decompressed, then the total length of the slice in bits is added to the bits extracted. But if a slice which is 1000 bits long has a syntax violation which is caught after 400 bits, this slice would not be decompressed fully and would only add 400 bits to the number of bits extracted. Each slice candidate
Figure 3.9: Chebyshev’s limit for 90% confidence interval for \( N_1 \)

Figure 3.10: Chebyshev’s limit of 90% confidence interval for \( N_2 \)
decompressed adds its number of bits extracted to the total number of bits extracted.
The total number of bits extracted can be compared to the size of the bitstream to indicate complexity. This complexity indicator can be measured empirically in the experiments. Unfortunately unlike the number of slice candidates it is harder to predict theoretically.

3.4.3 Effect of FEC code length on complexity

As previously discussed, a slice is divided into $L$ message portions of received words. $L$ decreases as the code length increases. For example, suppose there are 8000 bits in one slice. If a (16,8) binary code is used, then $L = 1000$. For a (26,16) binary code, $L = 500$. As $L$ decreases, the number of slice candidates in $G_a$ decreases for lower complexity.

But longer FEC code increases the complexity of FEC decoder. For example, a table lookup scheme is used for decoding of an $(n,k)$ binary code. A table of $2^n$ is set up and used to decode the FEC codewords. As $n$ increase, the entries of the table increase exponentially. For example, the (26,16) FEC code has a $2^{26}$ entries, which is $2^{10}$ times that of the (16,8) FEC code. The (26,16) FEC code has a larger complexity to set up the table. Thus when FEC codes that have the same error correction capability but different length are selected, there is a trade-off between the complexity of FEC decoder and decompressor.
Chapter 4

Performance Evaluation
In this chapter, the performance of the proposed scheme is objectively and subjectively evaluated. PSNR (introduced in Section 1.3) is used as the objective measure. As an illustrative indication of subjective picture quality, some frames from the sequence and comments on the sequence observed on the computer are given. Performance comparison of the proposed scheme and a classic FEC scheme using the same channel code, as well as a case without FEC is presented.

Two 2Mb/s MPEG2 sequences “Table Tennis”, and “Flower Garden” are used in the simulations, each one consisting of 180 frames at 30 frames/sec. For “Table Tennis”, the frame size is 704 * 480 for the luminance part and subsampled in both spatial directions for the color components (4:2:0 format). So the compression rate is 60:1. For “Flower Garden”, the frame size of the luminance component is also 704 * 480, while the chrominance is subsampled only horizontally but not vertically (4:2:2 format). The compression rate is is 80:1. These two sample sequences are obtained from the homepage of MPEG [12].

The simulations are done with software (C language). It runs under the Sun system (unix). The sequences are played under the Sun system to be observed its subjective quality. All experiments are conducted five times and the results presented are the average of these five trials. Complexity of the proposed scheme is also discussed.

4.1 Performance of classical ML receiver using (16,8) quasi-cyclic code

In this experiment, sequence “Table Tennis” is used. The simulation uses the block diagram shown in Figure 2.4 with a (16,8) quasi-cyclic code described in Appendix A and a binary symmetric channel (BSC). A table look-up ML decoder is implemented. For a 16-bit received word $r$, it produces a valid codeword $c_j$ with the shortest Hamming distance. Figure 4.1 shows the performance comparison between the cases with and without FEC in terms of PSNR versus the channel BER for luminance, blue chrominance and red chrominance.
Figure 4.1: PSNR versus channel BER for "Table Tennis", without and with FEC (from top to bottom: Luminance, Blue Chrominance, Red Chrominance)
Without FEC, the PSNR is not degraded for the channel BER of $10^{-6}$ or better. By using the (16,8) quasi-cyclic code with ML decoder, the PSNR can be kept unchanged for a channel BER as high as $3 \times 10^{-3}$. The results are not surprising since for an input channel BER of $3 \times 10^{-3}$, the (16,8) quasi-cyclic code provides a post-decoding word error probability of $1.6 \times 10^{-5}$ as shown in Figure 4.2 or a post-decoding BER of $10^{-6}$.

As an indication of subjective quality, three frames in each sequence are given in Figures 4.3 - 4.5. As shown in the middle pictures, without FEC, the reconstructed picture is still recognizable for a channel BER of $10^{-4}$ (Figure 4.3) although there is a noticeable black stripe. As the channel BER increases to $2 \times 10^{-3}$ (Figure 4.4) or more (Figure 4.5), the middle picture is not recognized. On the other hand, with a (16,8) quasi-cyclic code and ML decoding, the bottom pictures of Figures 4.3 - 4.5 are recognizable even for a channel BER of $1.2 \times 10^{-2}$ (Figure 4.5). It is also noted that at the channel BER of $1.2 \times 10^{-2}$, there are noticeable black stripes.

When the sequence is played on the computer, an improvement over the corrupted sequence in terms of subjective quality can be seen. In the corrupted sequences, when channel BER is high, the errors cause the discarding of a portion of
Figure 4.3: Frames 120 from compressed "Table Tennis" (2Mb/s), from top to bottom: no error, without FEC and with channel BER of $1 \times 10^{-4}$, with classic FEC and with channel BER of $1 \times 10^{-4}$
Figure 4.4: Frames 69 from compressed ‘Table Tennis’ (2Mb/s): from top to bottom: no error, without FEC and with channel BER of $2 \times 10^{-3}$, with classic FEC and with channel BER of $2 \times 10^{-3}$.
Figure 4.5: Frames 0 from compressed "Table Tennis" (2Mb/s), from top to bottom: no error, without FEC and with channel BER of $1.2 \times 10^{-2}$, with classic FEC and with channel BER of $1.2 \times 10^{-2}$
the slice or a whole slice in many places. There are some shifts in the sequences as well. The sequence is not recognized. But the sequence after error correction is much better. Many black stripes are removed and many shifts are corrected.

Simulation is also conducted for the sequence “Flower Garden”. Figure 4.6 shows the PSNR versus channel BER for the case without and with FEC. Similar results to Figure 4.1 are obtained, i.e., the PSNR remains unchanged for BER upto $10^{-6}$ (without FEC) and $3 \times 10^{-3}$ (with FEC).

Figures 4.30 - 4.9 show the sample pictures for different channel BER’s. The results again indicate that with a (16,8) quasi-cyclic code and a traditional ML decoder, pictures are still recognized for a channel BER upto $1.2 \times 10^{-2}$. Without FEC, recognizable pictures can only be obtained for a channel BER better than $10^{-4}$.

### 4.2 Performance of the proposed combined FEC/EC scheme using a (16,8) quasi-cyclic code

#### 4.2.1 Using slice candidates with shortest distance

All slice candidates in group $G_0$ with shortest distance are examined by the syntax checker and discontinuity measure selector and the one with no syntax violation and the lowest discontinuity measure is selected as the solution. This experiment is called SSG0 (search space group $G_0$). The same (16,8) quasi-cyclic code is used for performance comparison of the classic FEC and proposed scheme.

Figure 4.10 indicates that the proposed SSG0 provides an improvement in PSNR as compared to the classical FEC for a range of channel BER from $2 \times 10^3$ to $1.4 \times 10^{-2}$.

As an illustrative example of subjective equality, Figures 4.11 - 4.13 show the comparison of pictures obtained by the classical ML decoder (classic FEC) and proposed MCL decoder with $G_0$ (SSG0) for a channel BER of $1.2 \times 10^{-2}$. In these
Figure 4.6: PSNR versus channel BER for "Flower Garden", without FEC and with FEC (from top to bottom: Luminance, Blue Chrominance, and Red Chrominance)
Figure 4.7: Frames 0 from compressed "Flower Garden" (2Mb/s), from top to bottom: no error, without FEC and with channel BER of $1 \times 10^{-4}$, with classic FEC and with channel BER of $1 \times 10^{-4}$
Figure 4.8: Frames 136 from compressed "Flower Garden" (2Mb/s), from top to bottom: no error, without FEC and with channel BER of $2 \times 10^{-3}$, with classic FEC and with channel BER of $2 \times 10^{-3}$
Figure 4.9: Frames 14 from compressed "Flower Garden" (2Mb/s), from top to bottom: no error, without FEC and with channel BER of $1.2 \times 10^{-2}$, with classic FEC and with channel BER of $1.2 \times 10^{-2}$
Figure 4.10: PSNR versus BER for "Table Tennis", classic FEC and SSG0, from top to bottom: Luminance, Blue Chrominance and Red Chrominance
pictures, the top pictures are the case of no channel error and can be used as reference. The proposed scheme (bottom pictures) offers a better performance than the classical ML decoder (middle pictures).

When the whole sequence is observed on the computer, the improvement of SSG0 over classic FEC in the subjective picture quality is also noticed. With the classic FEC, there are still some errors that are serious enough to cause the discarding of a portion of the slices. There are still some shifts. The sequence after SSG0 looks better. Most black stripes are removed and most shifts are corrected.

4.2.2 Using slice candidates in $G_0$ and $G_1$

In this experiment, called SSG1, the MCL decoder provides candidates in groups $G_0$ (shortest distance) and $G_1$ (shortest-distance-plus-one).

Let $S_0/s$ and $S_1/s$ denote the slice candidates in $G_0$ and $G_1$, respectively. The syntax/semantic checker consider all of them for validity. Only the slice candidates with valid syntax/semantic are further examined for the discontinuity measure. As discussed in Section 3.3.5, the modified discontinuity measure of a slice candidate $S_{ki}$ defined as

$$M_{ki} \triangleq M_{ki} + \alpha_k$$

where $k = 0$ or $1$ and $\alpha_0 = 0$, $\alpha_1 > 0$. $M_{ki}$ is the discontinuity measure defined in Section 3.1.2. The slice candidate $S_{ki}$ is finally selected as the solution if it satisfied the syntax/semantic rules and has the lowest value $M_{ki}$. It can be seen that $G_1$ has more candidates than $G_0$. The experiment SSG1 can provide a better PSNR than the previous SSG0, only if there are slice candidates in $G_1$, which are finally selected as solutions with larger PSNR. On the other hand, if slice candidates in $G_1$ are more often selected than slice candidates in $G_0$, the overall performance of SSG1 can be worse than SSG0. In this sense, $\alpha_1$ is selected to maximize the PSNR. Figure 4.14 shows the PSNR versus $\alpha_1$ for a channel BER of $1.2 \times 10^{-2}$. It indicates
Figure 4.11: Frames 2 from compressed "Table Tennis" (2Mb/s): from top to bottom: no error, with classic FEC and with channel BER of $1.2 \times 10^{-2}$, with SSG0 and with channel BER of $1.2 \times 10^{-2}$
Figure 4.12: Frames 135 from compressed "Table Tennis" (2Mb/s): from top to bottom: no error, with classic FEC and with channel BER of $1.2 \times 10^{-2}$, with SSG0 and with channel BER of $1.2 \times 10^{-2}$
Figure 4.13: Frames 179 from compressed “Table Tennis” (2Mb/s), from top to bottom: no error, with classic FEC and with channel BER of $1.2 \times 10^{-2}$, with SSG0 and with channel BER of $1.2 \times 10^{-2}$
Table 4.1: PSNR for different $\alpha_1$, at BER of $1.2 \times 10^{-2}$.

<table>
<thead>
<tr>
<th>$\alpha_1$</th>
<th>PSNR(Y)</th>
<th>PSNR(Cb)</th>
<th>PSNR(Cr)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>27.31</td>
<td>36.56</td>
<td>36.09</td>
</tr>
<tr>
<td>30</td>
<td>28.50</td>
<td>37.51</td>
<td>37.04</td>
</tr>
<tr>
<td>50</td>
<td>28.44</td>
<td>37.43</td>
<td>36.78</td>
</tr>
<tr>
<td>75</td>
<td>28.44</td>
<td>37.43</td>
<td>36.78</td>
</tr>
</tbody>
</table>

that for small value of $\alpha_1$, the performance of SSG1 is worse than that of SSG0. This confirms that when $\alpha_1$ is small, more slice candidates from $G_1$ are selected over those from $G_0$ with poorer overall PSNR. As $\alpha_1$ increases, the PSNR offered by SSG1 increases and becomes better than that of SSG0 until the PSNR gets to its peak value at around $\alpha_1 = 30$ in the simulation. For $\alpha_1 > 30$, the PSNR offered by SSG1 drops. For larger $\alpha_1$, the PSNR keeps unchanged.

The characteristics of the "PSNR versus $\alpha_1$" curve can be explained as follows. By increasing $\alpha_1$, the modified discontinuity measure of slice candidates from group $G_1$ is increased intentionally and hence reduce the chance that a slice candidate in group $G_1$ is selected over the one from group $G_0$. For a sufficiently large value of $\alpha_1$, slice candidates from group $G_1$ is selected only when no slice candidate from $G_0$ satisfies the syntax\semantic rules. In this case, the selection algorithm tends to be the following. If there are some slice candidates from group $G_0$ with valid syntax\semantic, select the one with lowest discontinuity measure. Otherwise consider slice candidates from group $G_1$ using the same rule.

The results plotted in Figure 4.14 and summarized in Table 4.1 show that the scheme SSG1 is robust for large $\alpha_1$, i.e., the PSNR remains unchanged and higher than that of SSG0.

Using $\alpha_1 = 30$, Figure 4.15 shows that SSG1 provides an improvement in PSNR as compared to SSG0 for a range of channel BER from $10^{-3}$ to $1.4 \times 10^{-2}$.

As an illustrative indication of subjective quality, Figures 4.16 - 4.18 show the comparison pictures obtained by SSG0 and SSG1. SSG1 (bottom pictures) offers a better performance than SSG0 (middle pictures).

When the whole sequence is observed on the computer, the improvement of
Figure 4.14: PSNR versus $\alpha_1$ for "Table Tennis" at BER of $1.2 \times 10^{-2}$, from top to bottom: Luminance, Blue Chrominance, and Red Chrominance
Figure 4.15: PSNR versus BER for "Table Tennis" (2Mb/s), with SSG0 and SSG1, from top to bottom: Luminance, Blue Chrominance, and Red Chrominance
Figure 4.16: Frames 2 from compressed "Table Tennis" (2Mb/s), from top to bottom: no error, with SSG0 and with channel BER of $1.2 \times 10^{-2}$, with SSG1 and with BER of $1.2 \times 10^{-2}$
Figure 4.17: Frames 46 from compressed "Table Tennis" (2Mb/s), from top to bottom: no error, with SSG0 and with channel BER of $1.2 \times 10^{-2}$, with SSG1 and with channel BER of $1.2 \times 10^{-2}$
Figure 4.18: Frames 169 from compressed "Table Tennis" (2Mb/s), from top to bottom: no error, with SSG0 and with channel BER of $1.2 \times 10^{-2}$, with SSG1 and with channel BER of $1.2 \times 10^{-2}$
SSG1 over SSG0 in the subjective picture quality can be seen. There is no black stripes and only some black blocks remained. Also some shifts and wrong DC values are left.

### 4.2.3 Using the slice candidates in $G_0$, $G_1$ and $G_2$

In this experiment, called SSG2, the MCL decoder provides slice candidates in $G_0$, $G_1$ and $G_2$. Based on the previous results, set $\alpha_0 = 0$, $\alpha_1 = 30$ and search for the best value for $\alpha_2$.

Figure 4.19 and Table 4.2 show the simulation results in terms of PSNR versus $\alpha_2$ for a channel BER of $1.2 \times 10^{-2}$ and for the “Table Tennis” sequence. The results indicate the same tendency: as $\alpha_2$ increases, the PSNR offered by SSG2 increases and becomes better than that of SSG1. The PSNR reaches the peak and then drops to an unchanged value as $\alpha_2$ keeps increasing. However, compared to Figure 4.14, the peak (at $\alpha_2 = 60$) and unchanged value (at $\alpha_2 = 80$) of the PSNR for large $\alpha_2$ in Figure 4.19 are closer.

Figure 4.20 shows that SSG2 (with $\alpha_1 = 30$ and $\alpha_2 = 60$) provides an improvement in PSNR as compared to classic FEC, SSG0 and SSG1 for a range of channel BER from $3 \times 10^{-3}$ to $1.4 \times 10^{-2}$.

As an illustrative indication of subjective quality, Figures 4.21 - 4.23 show the comparison pictures obtained by SSG2 and SSG1. SSG2 (bottom pictures) offers a better performance than SSG1 (middle pictures).

When the whole sequence is observed on the computer, the improvement in the subjective picture quality over SSG1 is also noticed. Almost all black stripes are removed and shifts are corrected. There are only checker board blocks on the
Figure 4.19: PSNR versus $\alpha_2$ for "Table Tennis" (2Mb/s), at BER of $1.2 \times 10^{-2}$, from top to bottom: Luminance, Blue Chrominance, Red chrominance.
Figure 4.20: PSNR versus BER for "Table Tennis" (2Mb/s), from top to bottom: Luminance, Blue Chrominance and Red Chrominance
Figure 4.21: Frames 2 from compressed "Table Tennis" (2Mb/s), from top to bottom: no error, with SSG1 and with BER of $1.2 \times 10^{-2}$, with SSG2 and with BER of $1.2 \times 10^{-2}$
Figure 4.22: Frames 75 from compressed "Table Tennis" (2Mb/s), from top to bottom: no error, with SSG1 and with BER of $1.2 \times 10^{-2}$, with SSG2 and with BER of $1.2 \times 10^{-2}$
Figure 4.23: Frames 169 from compressed "Table Tennis" (2Mb/s), from top to bottom: no error, with SSG1 and with BER of $1.2 \times 10^{-2}$, with SSG2 and with BER of $1.2 \times 10^{-2}$
pictures (which are not detected well by our discontinuity measure).

This experiment is also conducted with the sequence "Flower Garden". Figure 4.24 shows the PSNR versus channel BER for the cases with classic FEC and SSG2. Similar results to Figure 4.20 are obtained i.e., SSG2 provides an improvement over the classic FEC in PSNR.

Figures 4.32 - 4.27 show the sample pictures. The results again indicate the better performance of SSG2 over classic FEC.

4.3 Simulation results with a higher-rate (21,16) code

In this section, the performance of the proposed scheme is examined using a shortened BCH (21,16) code, derived from the single error correcting BCH (31,26) code. The minimum distance of this code is 3 so it can correct 1 error. The code rate is $\frac{16}{21}$ and the corresponding bandwidth expansion is $\frac{21}{16} = 1.3125$. The sequence used in the simulation is the same "Table Tennis" sequence used in the previous simulation.

4.3.1 Performance of a classical ML decoder

Figure 4.28 shows the performance comparison between the cases with and without FEC in terms of PSNR versus the channel BER for luminance, blue chrominance and red chrominance.

Without FEC, the PSNR is not degraded for the channel BER of $10^{-8}$ or better as previously indicated in Section 4.1. By using the (21,16) code with ML decoder, the PSNR can be kept unchanged for a channel BER as high as $3 \times 10^{-4}$. The results are not surprising since for an input channel BER of $3 \times 10^{-4}$, the (21,16) code provides a post-decoding word error probability of around $2.1 \times 10^{-5}$ (equivalent to a post-decoding BER of $10^{-6}$) as shown in Figure 4.29. This performance is inferior to the case using the (16,8) quasi-cyclic code (Section 4.1) because this (21,16) code has a weaker error correction capability than the (16,8) code.

Figure 4.30 shows the comparison pictures between the case with and without the (21,16) FEC code. The results indicate that with a (16,8) quasi-cyclic code and
Figure 4.24: PSNR versus BER for "Flower Garden" (2Mb/s) with classic FEC and SSG2, from top to bottom: Luminance, Blue Chrominance, and Red Chrominance for "Flower Garden"
Figure 4.25: Frames 25 from compressed "Flower Garden" (2Mb/s), from top to bottom: no error, with classic FEC and with BER of $1.2 \times 10^{-2}$, with SSG2 and with BER of $1.2 \times 10^{-2}$
Figure 4.26: Frames 100 from compressed “Flower Garden” (2Mb/s), from top to bottom: no error, with classic FEC and with BER of $1.2 \times 10^{-2}$, with SSG2 and with BER of $1.2 \times 10^{-2}$
Figure 4.27: Frames 159 from compressed "Flower Garden" (2Mb/s), from top to bottom: no error, with classic FEC and with BER of $1.2 \times 10^{-2}$, with SSG2 and with BER of $1.2 \times 10^{-2}$
Figure 4.28: PSNR versus BER for "Table Tennis" (2Mb/s), without FEC and with (21,16) FEC code, from top to bottom: Luminance, Blue Chrominance and Red Chrominance
a traditional ML decoder, the picture is much better than the one without FEC for the BER of $2 \times 10^{-3}$.

As evidence of subjective quality, the sequences are observed on the computer. In the corrupted sequences, when channel BER is high, the errors are serious and cause the discarding of a portion of the slice or a whole slice in many places. There are some shifts in the sequences as well. The sequences are not recognized. But the sequences after error correction are much better. Many black stripes are removed and many shifts are corrected.

4.3.2 Performance of the proposed scheme

Using $\alpha_0 = 0$, $\alpha_1 = 30$, $\alpha_2 = 60$ for the SSG2 with the shortened BCH (21,16) code, Figure 4.31 shows the simulation results in terms of PSNR versus channel BER. It indicates that SSG2 provides an improvement over the classic FEC using ML decoder in PSNR for the channel BER of $2 \times 10^{-4}$ or higher.

Figure 4.32 shows the sample pictures obtained from classic FEC and SSG2. The results again indicate the better performance of SSG2 over classic FEC.

As an indication of subjective quality, the sequence after this experiment are observed on the computer. There is an improvement in the subjective picture quality
Figure 4.30: Frames 2 from compressed "Table Tennis" (2Mb/s), from top to bottom: no error, without FEC and with channel BER of $2 \times 10^{-3}$, with (21,16) FEC and with channel BER of $2 \times 10^{-3}$
Figure 4.31: PSNR versus BER for "Table Tennis" (2Mb/s), with (21,16) FEC code and with SSG2, from top to bottom: Luminance, Blue Chrominance and Red Chrominance
Figure 4.32: Frame 2 from compressed "Table Tennis" (2Mb/s), from top to bottom: no error, with (21,16) FEC and with BER of $2 \times 10^{-3}$, with SSG2 and with BER of $2 \times 10^{-3}$
over that of the classic FEC scheme. Almost all black stripes are removed and shifts are corrected. There are only some checker board blocks on the pictures.

4.4 Complexity of the proposed scheme

This thesis uses two measures for complexity: number of slice candidates to be decompressed and number of bits extracted. They are discussed in the next sections.

4.4.1 Number of slice candidates to be decompressed

Given BER of $1.2 \times 10^{-2}$, numbers of slice candidates for different slice lengths are collected and plotted in Figures 4.33 - 4.35. The sequence has a total of $120 \times 180 = 21600$ slices.

Let divide the slice length into a number of bins. Each bin has a size of 10. In each bin, the average number of slice candidates is calculated. The results based on the statistical collections in Figures 4.33 - 4.35 are plotted in Figures 4.36 - 4.38, respectively. They indicate that the average values are in a good agreement with the theoretical prediction.

The average simulation results are also compared with the confidence limits for
Figure 4.34: Statistical collection of numbers of slice candidates in $G_1$ versus slice length, at BER of $1.2 \times 10^{-2}$

Figure 4.35: Statistical collection of numbers of slice candidates in $G_2$ versus slice length, at BER of $1.2 \times 10^{-2}$
Figure 4.36: Average number of slice candidates in $G_0$ versus slice length

Figure 4.37: Average number of slice candidates in $G_1$ versus slice length
Figure 4.38: Average number of slice candidates in $G_2$ versus slice length

$G_0$ (calculated in Section 3.4.1) in Figure 4.39, 4.40.

To compare these two limits, they are plotted together in Figure 4.41.

It shows that the Chebyshev confidence limit is much looser than the central limit confidence limit.

The confidence limits for $N_1$ and $N_2$ are shown in Figure 4.42 and 4.43.

These figures indicates that the simulation results conform to the theoretic results well.

4.4.2 Number of bits extracted from the bitstream

A second proxy for complexity is number of bits extracted as explained in Section 3.4.2.

Figure 4.44 shows the number of bits extracted versus BER for different search spaces. As can be seen the cost of expanding the search space from $G_0$ to $G_0,G_1$ and then to $G_0, G_1$ and $G_2$ is very dramatic as the BER increases.

To see the effect of expanding search space on the number of bit extracted more clearly, the BER is fixed to $1.2 \times 10^{-2}$, and the numbers of bits extracted for different search space are listed in the Table 4.3.

Table 4.3 shows the increase in the bits extracted proxy for complexity as well as
Figure 4.39: Central limit confidence limit of 90% confidence interval for $N_0$

Figure 4.40: Chebyshev limit of 90% confidence interval for $N_0$

<table>
<thead>
<tr>
<th>search space</th>
<th>Number of Bits Extracted</th>
<th>PSNR(Y)</th>
</tr>
</thead>
<tbody>
<tr>
<td>classic FEC</td>
<td>24167808</td>
<td>26.49</td>
</tr>
<tr>
<td>SSG0</td>
<td>25593340</td>
<td>27.47</td>
</tr>
<tr>
<td>SSG1</td>
<td>86515424</td>
<td>28.50</td>
</tr>
<tr>
<td>SSG2</td>
<td>510091901</td>
<td>29.61</td>
</tr>
</tbody>
</table>

Table 4.3: Number of bits extracted for different search space
Figure 4.41: Central limit and Chebyshev limits of 90% confidence interval for $N_0$.

Figure 4.42: Chebyshev's limit for 90% confidence interval for $N_1$. 
Figure 4.43: Chebyshev's limit for 90% confidence interval for $N_2$

Figure 4.44: Number of bits extracted versus BER for different search space
the PSNR resulting from various search spaces. As can be seen there are diminishing
returns in terms of PSNR as greater and greater costs are paid in complexity. With
this in mind the search space is not further expanded beyond $G_2$. This decision may
be less valid at higher BER's.

4.5 Further discussions

There are a number of questions that arise from the proposed technique: How big
should the search space be? How do $\alpha_d$ and discontinuity affect the selection of
slice candidates? Some discussions of these questions are presented in the following
sections.

4.5.1 Value of expanding the search space

Already in Section 4.4.2 the value of expanding the search space was examined by
looking at overall complexity and overall PSNR. In this section the size of the search
space is examined by looking at the decisions made for individual slices. Here two
questions are discussed: is it necessary to expand the search space; how big should
the search space be?

To examine the decisions by the proposed scheme for an individual slice some
definitions are necessary.

First, let $S_{oi}$ be the $ith$ slice of the original bitstream, i.e., the correct slice
candidate, and let $S_{ci}$ be slice $i$ of the corrupted bitstream. Note that these two
terms include both the information bits and the parity bits used to send a slice. Of
course only the information bits carry video information.

The Hamming distance between them is the true Hamming distance $d_{ti}$, i.e.,

$$d_{ti} = d_H(S_{oi}, S_{ci})$$

(4.2)

Then let $S_{mi}$ be the shortest distance slice candidate for slice $i$. The shortest distance
$\text{d}_{mi}$ can be defined as
Table 4.4: Number of slices whose correct slice candidate $S_{oi}$ is in each group, for different BER's

<table>
<thead>
<tr>
<th>BER</th>
<th>$8 \times 10^{-3}$</th>
<th>$1 \times 10^{-2}$</th>
<th>$1.2 \times 10^{-2}$</th>
<th>$1.4 \times 10^{-2}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>number of $S_{oi}$s in $G_0$</td>
<td>21460</td>
<td>21320</td>
<td>21122</td>
<td>20834</td>
</tr>
<tr>
<td>number of $S_{oi}$s in $G_1$</td>
<td>135</td>
<td>268</td>
<td>444</td>
<td>695</td>
</tr>
<tr>
<td>number of $S_{oi}$s in $G_2$</td>
<td>4</td>
<td>9</td>
<td>28</td>
<td>58</td>
</tr>
<tr>
<td>number of $S_{oi}$s in $G_3$</td>
<td>1</td>
<td>3</td>
<td>6</td>
<td>12</td>
</tr>
<tr>
<td>number of $S_{oi}$s in $G_4$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 4.5: Number of slices selecting slice candidates correctly, at BER $1.2 \times 10^{-2}$

<table>
<thead>
<tr>
<th>SSG0</th>
<th>SSG1</th>
<th>SSG2</th>
</tr>
</thead>
<tbody>
<tr>
<td>number of slices choosing $S_{oi}$s in $G_0$</td>
<td>20942</td>
<td>20946</td>
</tr>
<tr>
<td>number of slices choosing $S_{oi}$s in $G_1$</td>
<td>0</td>
<td>130</td>
</tr>
<tr>
<td>number of slices choosing $S_{oi}$s in $G_2$</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

The difference between these two distances ($d_{ti} - d_{mi}$) is the extra errors of the correct slice candidate $S_{oi}$ over the shortest distance slice candidate $S_{mi}$. Then the correct slice candidate for slice $i$ is in group $(d_{ti} - d_{mi})$, i.e., $G(d_{ti} - d_{mi})$.

Out of a total of 21600 slices examined and for various BER's, Table 4.4 shows the number of slices whose correct slice candidate $S_{oi}$ is in $G_0$, $G_1$, and $G_2$.

It shows that at each BER, possibly correct slice candidates $S_{oi}$s lie outside of $G_0$. When the BER is high, one slice candidate $S_{oi}$ even lies in $G_4$. This table indicates the value of expanding the search space as BER increases.

Next examine the number of slices for which the scheme correctly chooses the correct slice candidate $S_{oi}$. These numbers are shown in Table 4.5, at BER of $1.2 \times 10^{-2}$.

It can be seen from the table that as the search space is expanded, the number of slices that choose the correct slice candidates $S_{oi}$s in each group increases too. It's interesting that the number that chooses $S_{oi}$s in $G_0$ increases as search space increases. This is thought to be because of the fact that the discontinuity measure is done across slice boundaries: thus a better decoding of one slice can improve the

103
 decoding of the slice below it.

The above discussion shows that as the search space is expanded, the video quality is improved. But as discussed in Section 4.4, the complexity increases fast as the search space is expanded. Then another question is how big should the search space be?

In Table 4.4, at the BER of $1 \times 10^{-2}$, the number of slices in $G_2$ is relatively large (28/21600). The number of slices in $G_2$ is even larger (58/21600) at $1.4 \times 10^{-2}$, so it's necessary to expand the search space to $G_2$ for these BER's.

Then let's consider if it is necessary to consider groups outside $G_2$.

At BER $8 \times 10^{-3}$, there is only one slice candidates in $G_3$ and its PSNR after SSG2 is 33.01 dB. The PSNR of the original sequence is 30.35 dB. The degradation caused by the slices outside of $G_2$ is not serious.

At BER $1 \times 10^{-2}$, there are 3 slice candidates in $G_3$. The PSNR of that slices after SSG2 is shown in Table 4.6: It can be seen from the table that the degradation is not serious for the slices outside of the search space compared to the original PSNR 30.35 dB. Also it is not necessary to consider the slice candidates in $G_3$.

At BER $1.2 \times 10^{-2}$, there are 6 slice candidates in $G_3$. The PSNR of those slices after the simulation of SSG2 is shown in Table 4.7. It indicates that for most of the slices outside of the search space, the degradation is not serious except for

<table>
<thead>
<tr>
<th>Slice</th>
<th>PSNR</th>
</tr>
</thead>
<tbody>
<tr>
<td>frame 23 slice 112</td>
<td>38.60</td>
</tr>
<tr>
<td>frame 100 slice 71</td>
<td>32.38</td>
</tr>
<tr>
<td>frame 168 slice 88</td>
<td>32.92</td>
</tr>
</tbody>
</table>

Table 4.6: PSNR of slices out of search space, at BER $1 \times 10^{-2}$

<table>
<thead>
<tr>
<th>Slice</th>
<th>PSNR</th>
</tr>
</thead>
<tbody>
<tr>
<td>frame 13 slice 59</td>
<td>29.88</td>
</tr>
<tr>
<td>frame 23 slice 112</td>
<td>38.46</td>
</tr>
<tr>
<td>frame 100 slice 71</td>
<td>32.45</td>
</tr>
<tr>
<td>frame 103 slice 83</td>
<td>20.39</td>
</tr>
<tr>
<td>frame 118 slice 23</td>
<td>28.27</td>
</tr>
<tr>
<td>frame 168 slice 88</td>
<td>33.30</td>
</tr>
</tbody>
</table>

Table 4.7: PSNR of slices out of search space, at BER $1.2 \times 10^{-2}$
frame 103 slice 83). The result from Section 4.2.3, the PSNR of the whole sequence is degraded less than 1 dB (0.67 dB).

The PSNR's of the slices in $G_3$ at different BER $8 \times 10^{-3}, 1 \times 10^{-2}$, and $1.2 \times 10^{-3}$ indicate that the slices in $G_3$ don't cause serious problem when BER is not higher than $1.2 \times 10^{-2}$. So it's unnecessary to expand the search space to $G_3$ for these BER's.

When the sequences after SSG2 (BER: $1.2 \times 10^{-2}$) are observed, which have been shown in Figure 4.21 - 4.23, it can be seen that the pictures are good and the slices in $G_3$ don't cause serious problem.

From the above discussion, two conclusions can be drawn:
1. It's necessary to expand the search space beyond $G_0$. The reconstructed video quality can be improved by expanding search space.
2. When BER $p \leq 1.2 \times 10^{-2}$, the search space does not need to be bigger than $G_0, G_1$ and $G_3$.

Simulations should be done across more sequences to see if these observations hold.

### 4.5.2 Evaluate discontinuity measure and $\alpha_a$

The discontinuity measure and $\alpha_a$ play different roles in selecting slice candidates. The discontinuity measure is responsible for selection both within one group and between different groups, while $\alpha_a$ is only used to select slice candidates between different groups. So the selection of an incorrect slice candidate but in the correct group is the fault of discontinuity measure and not the $\alpha_a$. The selection of slice candidates in an incorrect group may be due to both discontinuity measure and $\alpha_a$.

The numbers of slices that select wrong groups and correct groups but wrong slice candidates at BER $1.2 \times 10^{-2}$ are shown in Table 4.8.

As stated in Section 4.2.2 and 4.2.3, the $\alpha_a$ is selected based on PSNR. Table 4.8 indicates that as expected, the value that maximizes PSNR namely $\alpha_2 = 60$, also maximizes (by a thin margin) the number of slice candidates that are selected.

105
<table>
<thead>
<tr>
<th>α₂</th>
<th>correct</th>
<th>incorrect group</th>
<th>correct group but incorrect can.</th>
</tr>
</thead>
<tbody>
<tr>
<td>35</td>
<td>21187</td>
<td>213</td>
<td>200</td>
</tr>
<tr>
<td>60</td>
<td>21197</td>
<td>204</td>
<td>199</td>
</tr>
<tr>
<td>80</td>
<td>21192</td>
<td>207</td>
<td>201</td>
</tr>
</tbody>
</table>

Table 4.8: Slices selecting wrong groups and correct groups but wrong slice candidates

correctly, and minimizes the number of slices that select a candidate in an incorrect group.

Therefore, it can be concluded that it is hard to improve the video quality further by revising α₂. But further improvement may be achieved by revising the discontinuity measure.
Chapter 5

Conclusions and Future Work
5.1 Contribution

This thesis presents a joint channel/source error resilience scheme which combines the soft outputs and reliability information given by the channel decoder together with the source information. How to make use of the information provided by the decompressor and FEC decoder is the main contribution of this work.

The traditional ML decoder does an arbitrary selection of one solution from the slice candidates with the shortest distance $D^s$. But with further consideration for error resilience using other measures, there may be some slice candidates that are equivalent or even worse from channel coding viewpoint but have other better measures. The traditional ML decoder prevents the chance to examine such slice candidates. Thus, an MCL channel decoding scheme is considered. Unlike the traditional ML decoder, the proposed MCL decoding strategy gives out all shortest-distance slice candidates and the shortest distance plus $a$ ($a > 0$) slice candidates along with their reliability information. This enhances the performance because all equally good slice candidates with distance of $D^s$ and some worse slice candidates (from the channel coding viewpoint) with distance ($D^s + a$) are further considered using additional metrics based on the digital video signal properties.

The slice candidates given by MCL decoder are first selected using the syntax information of the digital video signal. Then the discontinuity property of the digital video is combined with the reliability information of the slice candidates to select among the candidates without syntax violation.

When the slice candidates that are not equally good from the channel coding viewpoint are considered, there is a balance between the discontinuity measure and the distance of the slice candidates. Thus an offset $\alpha_a$ based on the distance ($D^s + a$) is intentionally added to the discontinuity measure. The performance of the scheme is sensitive to the value of $\alpha_a$. A best value for $\alpha_a$ is obtained empirically. With this value, PSNR gets the highest and it has an improvement over the traditional FEC and the scheme with a smaller search space.
5.2 Conclusions

The following conclusions are drawn:

- Objective and Subjective video quality can be improved by using residual redundancy in the video to improve classical FEC
- The worst type of degradations both subjectively and objectively is due to syntax errors
- The worst type of degradations both subjectively and objectively aside from syntax errors is due to shifts in slices and errors in the DC terms
- The interslice nature of the discontinuity measure means that better performance in one slice improves performance in slices below. This same observation can likely be made of many error concealment or resilience techniques that depend on the residual redundancy of the video.

5.3 Future work

The discontinuity measure can be improved by investigating a more systematic approach to define its formula. This can be part of the work in the future.

As BER increases the search space should be increased. However this can result in very large increases in complexity. To avoid the complexity cost unnecessarily the size of the search space can be adaptively decided by monitoring the channel BER.

Suboptimum schemes which save on complexity should be evaluated. Since most of the gain in PSNR from increasing the search space comes from slices who have no slice candidates that are syntax error free in the smaller search space, then one scheme would be to only search group $G_i$ if there are no candidates without syntax errors in slice $G_j$ where $j < i$.

Soft FEC decoding can give more than one slice candidates and can give reliability information for each slice candidate. Thus it is suitable to be used in this
scheme. In the future, soft decoding such as Viterbi decoding can be applied into the proposed scheme.

The proposed scheme just uses the forward error correcting scheme. There is no feedback and iteration in the scheme. In the future, one can use the idea of Turbo coding that corrects the errors iteratively.

There are some other jobs that can be left as the future work such as considering of burst errors and using different compressed scheme such as MPEG4.
Bibliography


Appendix A

(16,8) quasi-cyclic code

The generator matrix of the systematic (16,8) quasi-cyclic code is given as

\[ G = [I, P] \]  \hspace{1cm} (A.1)

where \( I \) is an 8 \times 8 identity matrix and \( P \) is an 8 \times 8 binary circulant matrix of the form

\[
P = \begin{bmatrix}
1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\
1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\
1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\
0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\
1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 & 1 & 1 & 1
\end{bmatrix} \]  \hspace{1cm} (A.2)

The valid codewords for this FEC code are given as follows:
Appendix B

Matlab source code to derive the distance profile for the (16,8) quasi-cyclic code

Let \( \text{val} \) contain the valid codewords. Codewords are 16 bits long (each array entry is “0” or “1”). and there are 256 valid codewords. Generate all \( 2^{16} \) received words in the “1”, “0” format:

\[
i=1;
\text{allword} = \text{zeros}(65536,16);
\text{for } j_{15}=0:1 \\
\text{for } j_{14}=0:1 \\
\text{for } j_{13}=0:1 \\
\text{for } j_{12}=0:1 \\
\text{for } j_{11}=0:1 \\
\text{for } j_{10}=0:1 \\
\text{for } j_{9}=0:1 \\
\text{for } j_{8}=0:1 \\
\text{for } j_{7}=0:1 \\
\text{for } j_{6}=0:1
\]
for j5=0:1
for j4=0:1
for j3=0:1
for j2=0:1
for j1=0:1
for j0=0:1
    allword(i,:)=[j15 j14 j13 j12 j11 j10 j9 j8 j7 j6 j5 j4 j3 j2 j1 j0];
    i=i+1;
end
end
end
end
end
end
end
end
end
For the received words that are 0 bit away from the closest valid codeword
(Class 0 in Table 3.1), its distance to each valid codeword is calculated.
dist0=zeros(256,17);
for i=1:256
for j=1:256
There are only one possible distance profiles for the 256 received words in Class 0. This can be tested as follows:

for i=1:256
    hold = sum(abs(distO(1,:)-distO(i,:)));  
    if hold ~= 0
        i
    end
end

Nothing returns. So there are only one possible distance profiles for the 256 received words in Class 0.

For each of the possible received words, calculate how far it is to each valid codeword.

distance=zeros(65536,17);
for i=1:65536
    for j=1:256
        d = sum(xor(val(j,:),allword(i,:)));  
        distance(i,d+1) = distance(i,d+1) + 1;
    end
end

All the received words that are 1 bit away from the closest codeword (Class 1 in Table 3.1) are sort out. There should be \(\begin{pmatrix} 16 \\ 1 \end{pmatrix} = 16\) for each valid codeword, thus 16*256 = 4096 of these.

j=1;
dist1=zeros(4096,18);
for i=1:65536
if distance(i, 1) == 0
    if distance(i, 2) == 1
        dist1(j, 1) = i;
        dist1(j, 2:18) = distance(i, 1:17);
        j = j + 1;
    end
end
end

The possible types of distance profile for Class 1 is obtained as follows:

dist1prof = [1 dist1(1, 2:18)];
i = 1;
for j = 2:4096
    flag1 = 0;
    for k = 1:i
        flag2 = 0;
        for l = 2:18
            if dist1(j, l) == dist1prof(k, l)
                flag2 = 1;
            end
        end
        if flag2 == 0
            flag1 = 1;
            dist1prof(k, 1) = dist1prof(k, 1) + 1;
        end
    end
    if flag1 == 0
        i = i + 1;
        dist1prof(i, 1) = 1;
        dist1prof(i, 2:18) = dist1(j, 2:18);
There are only two possible distance profiles for the 4096 received words in Class 1. They are split half/half between the two profiles. (Note number in first column is number of received words with this profile, subsequent columns are the distance profile. Column 2 is the number of valid code words 0 bit away from the received words with this profile, column 3 is the number of valid codewords that are 1 bit away, etc).

Now for the received words in Class 2, which are 2 bits away from the closest valid codewords, calculate how far it is to each valid codeword. Note that there should be \( \binom{16}{2} = 120 \) for each valid codeword, so there are \( 120 \times 256 = 30720 \) of these.

```matlab
j=1;
dist2=zeros(30720,18);
for i=1:65536
    if distance(i,1) == 0
        if distance(i,2) == 0
            if distance(i,3) == 1
                dist2(j,1)=i;
                dist2(j,2:18)=distance(i,1:17);
                j=j+1;
            end
        end
    end
end
end
end
The possible types of distance profile for Class 2 is obtained as follows:
dist2prof=[1 dist2(1,2:18)];
i=1;
```
for j=2:30720
    flag1 = 0;
    for k=1:i
        flag2 = 0;
        for l=2:18
            if dist2(j,l) \neq dist2 prof(k,l)
                flag2 = 1;
            end
        end
        if flag2 == 0
            flag1 = 1;
            dist2 prof(k,1) = dist2 prof(k,1) \div 1;
        end
    end
    if flag1 == 0
        i = i+1;
        dist2 prof(i,1) = 1;
        dist2 prof(i,2:18) = dist2(j,2:18);
    end
end

Of the received words in Class 2 there are 8 types.

Now to those received codewords in Class 3. Note here a received word may be 3 bits away from 2 or more valid codewords. So the code is changed. Also how many received words in Class 3 can no longer be calculated before hand since the correction capabilities of the code is exceeded.

j=1;

dist3=zeros(31000,18); (the number is not know so we guess high)
for i=1:65536
    if distance(i,1) == 0

if distance(i,2) == 0
if distance(i,3) == 0
if distance(i,4) == 0
    dist3(j,1)=i;
    dist3(j,2:18)=distance(i,1:17);
    j=j+1;
end
end
end
end
end

The possible distance profiles for received words in Class 3 is obtained as follows:
dist3prof=[1 dist3(1,2:18)];

i=1;
for j=2:30208
    flag1 = 0;
    for k=1:i
        flag2 = 0;
        for l=2:18
            if dist3(j,l) == dist3prof(k,l)
                flag2 = 1;
            end
        end
        if flag2 == 0
            flag1 = 1;
            dist3prof(k,1) = dist3prof(k,1) + 1;
        end
    end
if flag1 == 0

There are 10 distinct types in Class 3. For those received words in Class 4, calculate how far it is to each valid codeword.

\[ j = 1; \]
\[ \text{dist4} = \text{zeros}(356, 18); \text{ (the number is not known so be guessed high)} \]

for \( i = 1:65536 \)
\[ \text{if distance}(i, 1) == 0 \]
\[ \text{if distance}(i, 2) == 0 \]
\[ \text{if distance}(i, 3) == 0 \]
\[ \text{if distance}(i, 4) == 0 \]
\[ \text{if distance}(i, 5) == 0 \]
\[ \text{dist4}(j, 1) = i; \]
\[ \text{dist4}(j, 2:18) = \text{distance}(i, 1:17); \]
\[ j = j + 1; \]
end
end
end
end
end
end
end
end
end
end

The possible distance profile for Class 4 is obtained as follows:
\[ \text{dist4prof} = \left[ 1 \quad \text{dist4}(1, 2:18) \right]; \]
\[ i = 1; \]
for \( j = 2:256 \)
\[ \text{flag1} = 0; \]
for k=1:i
    flag2 = 0;
    for l=2:18
        if dist4(j,l) ~= dist4prof(k,l)
            flag2 = 1;
        end
    end
    if flag2 == 0
        flag1 = 1;
        dist4prof(k,1) = dist4prof(k,1) + 1;
    end
end
if flag1 == 0
    i=i+1;
    dist4prof(i,1) = 1;
    dist4prof(i,2:18) = dist4(j,2:18);
end
end
All possible received words have been enumerated and there are no received words that are more than 4 away from all possible valid codewords. Put all the results in prof.
for i=1:18
    prof(1,i)=dist0prof(1,i);
end
for j=2:3
    prof(j,i)=dist1prof(j-1,i);
end
for j=4:11
    prof(j,i)=dist2prof(j-3,i);
end
for j=12:21
    prof(j,i)=dist3prof(j-11,i);
end
prof(22,i)=dist4prof(j-21,i);
end

To calculate the probability for each type of distance profile, i.e., for each line in Table 3.1, assume BER=1.2 × 10^{-2}.

ber =0.012;
for k=1:22
    prof(k,19)=0;
    for i=2:18
        prof(k,19) = prof(k,19) + prof(k,i)*ber^(-i-2)*(1-ber)^(-18-i);
    end
    prof(k,19) = prof(k,19)*prof(k,1)/256;
end
That is the last column of Table 3.1.
Appendix C

Variances of $N_0$, $N_1$ and $N_2$

From Table 3.1, the expectation $E\{f_0^2(x_j)\}$, $E\{f_1^2(x_j)\}$, $E\{f_2^2(x_j)\}$ and $E\{f_0(x_j)f_1(x_j)\}$, $E\{f_0(x_j)f_2(x_j)\}$, $E\{f_1(x_j)f_2(x_j)\}$ can be calculated. The variance $\text{var}\{N_i\}$ is:

$$\text{var}\{N_0\} = E\{N_0^2\} - E^2\{N_0\} = (E\{f_0^2(x_i)\})^L - (E\{f_0(x_i)\})^{2L} \quad (C.1)$$

For $N_1^2$,

$$N_1^2 = \sum_{j=1}^{L} f_1(x_j) \prod_{i=1,i\neq j}^{L} f_0(x_i) \sum_{k=1}^{L} f_1(x_k) \prod_{m=1,m\neq k}^{L} f_0(x_m)$$

$$= \sum_{j=1}^{L} \sum_{k=1}^{L} f_1(x_j)f_1(x_k) \prod_{i=1,i\neq j}^{L} \prod_{m=1,m\neq k}^{L} f_0(x_i)f_0(x_m) \quad (C.2)$$

Taking the expectation of $N_1^2$, the result is:

126
\[ E\{N_i^2\} = \sum_{j=1}^{L} \sum_{k=1}^{L} E\{f_1(x_j) f_1(x_k) \prod_{i=1, i \neq j}^{L} \prod_{m=1, m \neq k}^{L} f_0(x_i) f_0(x_m)\} \]

\[ = \sum_{j=1}^{L} E\{f_1(x_j) f_1(x_j)\} \prod_{i=1, i \neq j}^{L} \prod_{m=1, m \neq j}^{L} E\{f_0(x_i) f_0(x_m)\} \]

\[ + \sum_{j=1}^{L} \sum_{k=1}^{L} E\{f_1(x_j) f_1(x_k) \prod_{i=1, i \neq j}^{L} \prod_{m=1, m \neq k}^{L} f_0(x_i) f_0(x_m)\} \]

\[ = L(E\{f_1^2(x_j)\})(E\{f_2^2(x_i)\})^{L-1} \]

\[ + (L^2 - L)(E\{f_1(x_j) f_0(x_j)\})^{2}(E\{f_0^2(x_i)\})^{L-2} \]

\[ \text{var}\{N_i\} = E\{N_i^2\} - E^2\{N_i\} \quad \text{(C.3)} \]

\[ N_2 = \sum_{j=1}^{L} (f_2(x_j) \prod_{i=1, i \neq j}^{L} f_0(x_i)) + \sum_{i=1}^{L-1} \sum_{j=i+1}^{L} (f_1(x_i) f_1(x_j) \prod_{k=1, k \neq i, j}^{L} f_0(x_k)) \quad \text{(C.4)} \]

Call the first term of \(N_2\) \(a\), and call the second term \(b\), \(i.e.,\)

\[ a = \sum_{j=1}^{L} f_2(x_j) \prod_{i=1, i \neq j}^{L} f_0(x_i) \]

\[ b = \sum_{i=1}^{L-1} \sum_{j=i+1}^{L} f_1(x_i) f_1(x_j) \prod_{k=1, k \neq i, j}^{L} f_0(x_k) \]

then

\[ E\{N_2^2\} = E\{a^2\} + E\{b^2\} + 2E\{ab\} \quad \text{(C.5)} \]

where

127
The relationship between $i$, $j$, $k$, $Z$ is shown in Figure C.1. Fixing $i$, $j$, each term in the two inner summations ($k$, $l$ summations) can be represented by a point in the $(k, l)$ plane. Since $l = k + 1$ to $L$, points inside the triangular area that is bounded by the three solid lines represent the terms of the inner summation. $i = k$, $j = l$:

In Figure C.1, this is the cross point (point A in Figure C.1) with $l = j$. Let $E\{b_1^2\}_i$ be the portion of $E\{b_1^2\}$ in this case,

$$
E\{b_1^2\} = \sum_{i=1}^{L-1} \sum_{j=i+1}^{L} \sum_{k=1}^{L} \sum_{l=k+1}^{L} E\{f_1(x_i)f_1(x_j)f_1(x_k)f_1(x_l)\} \prod_{m=1,m\neq i,j}^{L} \prod_{n=1,n\neq k,l}^{L} f_0(x_n)f_0(x_m) \quad (C.7)
$$

The relationship between $i$, $j$, $k$, $l$ is shown in Figure C.1. Fixing $i$, $j$, each term in the two inner summations ($k$, $l$ summations) can be represented by a point in the $(k, l)$ plane. Since $l = k + 1$ to $L$, points inside the triangular area that is bounded by the three solid lines represent the terms of the inner summation. $i = k$, $j = l$:

In Figure C.1, this is the cross point (point A in Figure C.1) with $l = j$. Let $E\{b_1^2\}_i$ be the portion of $E\{b_1^2\}$ in this case,

$$
E\{b_1^2\} = \sum_{i=1}^{L-1} \sum_{j=i+1}^{L} E\{f_1(x_i)f_1(x_j)f_1(x_i)f_1(x_j)\} \prod_{m=1,m\neq i,j}^{L} \prod_{n=1,n\neq k,l}^{L} f_0(x_n)f_0(x_m) \quad (C.8)
$$

$k = i, l \neq j$: This case means the curve $k = i$ except the cross point (point A) in Figure C.1 with $l = j$. Let $E\{b_1^2\}_2$ be the portion of $E\{b_1^2\}$ in this case, it can be
Figure C.1: Calculation figure for $E\{b^2\}$

\[ E\{b^2\}_2 = \sum_{i=1}^{L-1} \sum_{j=i+1}^{L} \sum_{l=i+1, l \neq j}^{L} E\{f_1(x_i)f_1(x_j)f_1(x_l)\} \prod_{m=1, m \neq i, j}^{L} f_0(x_n) \prod_{n=1, n \neq i, l}^{L} f_0(x_m) \]

\[ = \sum_{i=1}^{L-1} \sum_{j=i+1}^{L} \sum_{l=i+1, l \neq j}^{L} E\{f_1^2(x_i)\}(E\{f_1(x_i)f_0(x_i)\})^2(E\{f_0^2(x_i)\})^{L-3} \]

\[ = \sum_{i=1}^{L-1} \sum_{j=i+1}^{L} (L - i - 1)E\{f_1^2(x_i)\}(E\{f_1(x_i)f_0(x_i)\})^2(E\{f_0^2(x_i)\})^{L-3} \]

\[ = \sum_{i=1}^{L-1} (L - i)(L - i - 1)E\{f_1^2(x_i)\}(E\{f_1(x_i)f_0(x_i)\})^2(E\{f_0^2(x_i)\})^{L-3} \]

\[ = \frac{L(L-1)(L-2)}{3} E\{f_1^2(x_i)\}(E\{f_1(x_i)f_0(x_i)\})^2(E\{f_0^2(x_i)\})^{L-3} \]

\[ (C.9) \]

$l = j, k \neq i$: This case means the curve $l = j$ except the cross point with $k = i$ (point A) in Figure C.1. Let $E\{b^2\}_3$ be the portion of $E\{b^2\}$ in this case, it can be

129
calculated as:

\[
E\{b^2\}_3 = \sum_{i=1}^{L-1} \sum_{j=i+1}^{L} \sum_{k=1, k \neq i}^{j-1} E\{f_1(x_i)f_1(x_j)f_1(x_k)f_1(x_j) \} \prod_{m=1, m \neq i, j}^{L} \prod_{n=1, n \neq k, j}^{L} f_0(x_n)f_0(x_m) \}
\]

\[
= \sum_{i=1}^{L-1} \sum_{j=i+1}^{L} \sum_{k=1, k \neq i}^{j-1} E\{f_1^2(x_i)\}(E\{f_1(x_i)f_0(x_i)\})^2(E\{f_0^2(x_i)\})^{L-3}
\]

\[
= \sum_{i=1}^{L-1} \sum_{j=i+1}^{L} (j-2)E\{f_1^2(x_i)\}(E\{f_1(x_i)f_0(x_i)\})^2(E\{f_0^2(x_i)\})^{L-3}
\]

\[
= \frac{L(L-1)(L-2)}{3} E\{f_1^2(x_i)\}(E\{f_1(x_i)f_0(x_i)\})^2(E\{f_0^2(x_i)\})^{L-3}
\]

(C.10)

\[l = i, k \neq j: \text{In Figure C.1, when } i = l, \text{ since } j = i + 1, \text{ and } k = l - 1, \text{ so } k \text{ is impossible to equal to } j, \text{i.e., there is no cross point for the two curves } l = i \text{ and } k = j \text{ within the triangle area. Let } E\{b^2\}_4 \text{ be the portion of } E\{b^2\} \text{ in this case. It can be calculated as:}
\]
\[ E\{b^2\}_4 = \sum_{i=1}^{L-1} \sum_{j=i+1}^{L} \sum_{k=1}^{i-1} E\{f_1(x_i)f_1(x_j)f_1(x_k)f_1(x_j) \}
\prod_{m=1, m \neq i, j}^{L} \prod_{n=1, n \neq k, i}^{L} f_0(x_m)f_0(x_n) \}
= \sum_{i=1}^{L-1} \sum_{j=i+1}^{L} \sum_{k=1}^{i-1} E\{f_1^2(x_i)\}(E\{f_1(x_i)f_0(x_i)\})^2(E\{f_0^2(x_i)\})^{L-3}
= \sum_{i=1}^{L-1} \sum_{j=i+1}^{L} (i-1)E\{f_1^2(x_i)\}(E\{f_1(x_i)f_0(x_i)\})^2(E\{f_0^2(x_i)\})^{L-3}
= \sum_{i=1}^{L-1} (L-i)(i-1)E\{f_1^2(x_i)\}(E\{f_1(x_i)f_0(x_i)\})^2(E\{f_0^2(x_i)\})^{L-3}
= \frac{L(L-1)(L-2)}{6} E\{f_1^2(x_i)\}(E\{f_1(x_i)f_0(x_i)\})^2(E\{f_0^2(x_i)\})^{L-3}
\]
(C.11)

\[ k = j, l \neq i: \text{In Figure C.1, when } k = j, \text{since } i = j - 1, \text{and } l = k + 1, \text{so } l \text{ is impossible to equal to } i, \text{i.e., there is no cross point for the two curves } l = i \text{ and } k = j \text{ within the triangle area. Let } E\{b^2\}_5 \text{ be the portion of } E\{b^2\} \text{ in this case. It} \]
can be calculated as:

\[ E\{b^2\}_6 = \sum_{i=1}^{L-1} \sum_{j=i+1}^{L} \sum_{l=i+1}^{L} E\{f_i(x_i)f_i(x_j)f_i(x_k)f_i(x_l)\} \prod_{m=1, m\neq i, j}^{L} f_0(x_n)f_0(x_m) \]

\[ = \sum_{i=1}^{L-1} \sum_{j=i+1}^{L} \sum_{l=i+1}^{L} E\{f_i^2(x_i)\}(E\{f_i(x_i)f_0(x_i)\})^2(E\{f_0^2(x_i)\})^{L-3} \]

\[ = \sum_{i=1}^{L-1} \sum_{j=i+1}^{L} (L - j)E\{f_i^2(x_i)\}(E\{f_i(x_i)f_0(x_i)\})^2(E\{f_0^2(x_i)\})^{L-3} \]

\[ = \sum_{i=1}^{L-1} \frac{(L - i)(L - i - 1)}{2} E\{f_i^2(x_i)\}(E\{f_i(x_i)f_0(x_i)\})^2(E\{f_0^2(x_i)\})^{L-3} \]

\[ = \frac{L(L - 1)(L - 2)}{6} E\{f_i^2(x_i)\}(E\{f_i(x_i)f_0(x_i)\})^2(E\{f_0^2(x_i)\})^{L-3} \]

(C.12)

\[ k \neq i, j, l \neq i, j: \text{In Figure C.1, these are the points in the triangle area except for the curves } k = i, k = j, l = i, \text{ and } l = j. \text{ Let } E\{b^2\}_6 \text{ be the portion of } E\{b^2\} \text{ in this case. It can be calculated as:} \]

\[ E\{b^2\}_6 = \sum_{i=1}^{L-1} \sum_{j=i+1}^{L} \sum_{k=1, k \neq i, j}^{L} \sum_{l=k+1, l \neq i, j}^{L} E\{f_i(x_i)f_i(x_j)f_i(x_k)f_i(x_l)\} \prod_{m=1, m \neq i, j}^{L} f_0(x_n)f_0(x_m) \]

\[ = \sum_{i=1}^{L-1} \sum_{j=i+1}^{L} \sum_{k=1, k \neq i, j}^{L} \sum_{l=k+1, l \neq i, j}^{L} (E\{f_i(x_j)f_0(x_i)\})^4(E\{f_0^2(x_i)\})^{L-4} \]

\[ = \frac{L(L - 1)(L - 2)(L - 3)}{4} (E\{f_i(x_j)f_0(x_i)\})^4(E\{f_0^2(x_i)\})^{L-4} \]

(C.13)

Add all the results, the expectation of \(b^2\) can be calculated as:
Next the expectation of $ab$ is calculated.

$$E\{ab\} = \sum_{j=1}^{L} \sum_{k=1}^{L-1} \sum_{l=k+1}^{L} E\{f_2(x_j)f_1(x_k)f_1(x_l)\} \prod_{i=1,i\neq j}^{L} \prod_{m=1,m\neq k,l}^{L} f_0(x_i)f_0(x_m)$$

(C.15)

It also can be divided into several cases. The relationship between $j, k, l$ is shown in Figure C.2. As C.1, since $l = k + 1$ to $L$, the triangle area that is bounded by the three solid lines represents the possible points here.

$k = j$: In this case, $l$ is impossible to equal to $j$. Let $E\{ab\}_1$ be the portion of $E\{ab\}$ in this case. It is calculated as:

$$E\{b^2\} = (L^2 - L)/2(E\{f_1^2(x_i)\})^2(E\{f_0^2(x_i)\})^{L-2}$$

$$+ L(L - 1)(L - 2)E\{f_1^2(x_i)\}(E\{f_1(x_i)f_0(x_i)\})^2(E\{f_0^2(x_i)\})^{L-3}$$

$$+ \frac{L(L - 1)(L - 2)(L - 3)}{4}(E\{f_1(x_i)f_0(x_i)\})^4(E\{f_0^2(x_i)\})^{L-4}$$

(C.14)
\[ E\{ab\}_1 = \sum_{j=1}^{L} \sum_{l=j+1}^{L} E\{f_2(x_j)f_1(x_k)f_1(x_l) \prod_{i=1,i\neq j}^{L} \prod_{m=1,m\neq k,l}^{L} f_0(x_i)f_0(x_m)\} \]

\[ = \sum_{j=1}^{L} \sum_{l=j+1}^{L} E\{f_1(x_l)f_2(x_i)\}E\{f_1(x_i)f_0(x_l)\}(E\{f_0^2(x_i)\})^{L-2} \]

\[ = (L^2 - L)/2 E\{f_1(x_l)f_2(x_i)\}E\{f_1(x_i)f_0(x_l)\}(E\{f_0^2(x_i)\})^{L-2} \]

(C.16)

\[ l = j: \text{In this case, } k \text{ is impossible to equal to } j. \text{ Let } E\{ab\}_2 \text{ be the portion of } E\{ab\} \text{ in this case. It is calculated as:} \]

\[ E\{ab\}_2 = \sum_{j=1}^{L} \sum_{k=1}^{j-1} E\{f_2(x_j)f_1(x_k)f_1(x_l) \prod_{i=1,i\neq j}^{L} \prod_{m=1,m\neq k,l}^{L} f_0(x_i)f_0(x_m)\} \]

\[ = \sum_{j=1}^{L} \sum_{k=1}^{j-1} E\{f_1(x_k)f_2(x_i)\}E\{f_1(x_i)f_0(x_k)\}(E\{f_0^2(x_i)\})^{L-2} \]

\[ = (L^2 - L)/2 E\{f_1(x_k)f_2(x_i)\}E\{f_1(x_i)f_0(x_k)\}(E\{f_0^2(x_i)\})^{L-2} \]

(C.17)

\[ k \neq j, l \neq j: \text{In this case, the points are the triangle area in Figure C.2 except for the two curves } k = j, l = j. \text{ Let } E\{ab\}_3 \text{ be the portion of } E\{ab\} \text{ in this case. It is calculated as:} \]

\[ E\{ab\}_3 = \sum_{j=1}^{L} \sum_{k=1,k\neq j}^{L-1} \sum_{l=k+1}^{L} E\{f_2(x_j)f_1(x_k)f_1(x_l) \prod_{i=1,i\neq j}^{L} \prod_{m=1,m\neq k,l}^{L} f_0(x_i)f_0(x_m)\} \]

\[ = \frac{L(L-1)(L-2)}{2} (E\{f_1(x_l)f_0(x_i)\})^2E\{f_2(x_i)f_0(x_i)\}(E\{f_0^2(x_i)\})^{L-3} \]

(C.18)
Add these three cases, then the expectation of \( ab \) is:

\[
E\{ab\} = (L^2 - L)E\{f_1(x_i)f_2(x_i)\}E\{f_1(x_i)f_0(x_i)\}(E\{f_0^2(x_i)\})^{L-2} + \frac{L(L-1)(L-2)}{2}(E\{f_1(x_i)f_0(x_i)\})^2E\{f_2(x_i)f_0(x_i)\}(E\{f_0^2(x_i)\})^{L-3}
\]  

(C.19)

Using Equation C.5, the expectation of \( N_2^2 \) can be calculated.