Video partitioning for wireless applications

This item was submitted to Loughborough University's Institutional Repository by the/an author.

Additional Information:


Metadata Record: https://dspace.lboro.ac.uk/2134/27109

Publisher: © Christopher Ian Richards

Rights: This work is made available according to the conditions of the Creative Commons Attribution-NonCommercial-NoDerivatives 2.5 Generic (CC BY-NC-ND 2.5) licence. Full details of this licence are available at: http://creativecommons.org/licenses/by-nc-nd/2.5/

Please cite the published version.
This item was submitted to Loughborough University as a PhD thesis by the author and is made available in the Institutional Repository (https://dspace.lboro.ac.uk/) under the following Creative Commons Licence conditions.

For the full text of this licence, please go to:
http://creativecommons.org/licenses/by-nc-nd/2.5/
Video Partitioning for Wireless Applications

by

Christopher Ian Richards

A Doctoral Thesis

Submitted in partial fulfilment of the requirements for the award of Doctor Of Philosophy of Loughborough University

September, 1998

© by Christopher Ian Richards, 1998
Acknowledgements

I would like to express my appreciation to :-

- my parents for their continued support and encouragement, without which this research would not have been possible; and to Dr Michael Richards for verifying the occasional bit of mathematics...

- my supervisor, Professor Hamid Gharavi, for his guidance during this research.

- Professor Roger Goodall and Dr David Parish for assistance during the writing of this thesis.

- the EPSRC for funding this research, and to the Faculty of Engineering for their generous financial support.

- Alan Schwarzenberger for computer / network support.

Lastly, I would like to thank all those in the department who have assisted in some way during the last four years, particularly to those of you who have been unwitting sounding boards for ideas during this time. You may not have known what I was talking about, but hopefully this thesis proves that I did...
Abstract

One of the key aspects of digital broadcast television is the need to compress the digital video to reduce the transmission bandwidth requirement. Numerous video coding standards have been defined with properties that depend upon the targeted application. For example, H.263 is primarily designed for low bit-rate applications, and MPEG-II is used for applications where quality is the most important aspect. These coding standards are primarily models for how to efficiently code video. They, in general, do not consider how the coded video is broadcast, and how the compressed video bitstream responds to transmission errors. In this thesis, the properties of the MPEG-II coding standard are investigated (although many of the results are extensible to the other frequency transform based video codecs).

For every transmission scheme, a new codec is required to achieve optimal source-channel coding. This is a particularly expensive and inflexible solution. The alternative, however, of completely separating the source coding and channel coding is less efficient. This thesis presents the concept of a transcoder operating on the compressed video bitstream, in order to:

i) create a compromise between the costs/efficiency of joint source-channel coding

ii) target the properties of the transmission link

iii) improve the response of the coded video bitstream to transmission errors.

This thesis primarily considers transmission through a mobile radio link, as this is one of the most severe environments likely to be used for video applications. The suitability of various M bits-per-symbol modulation schemes, with different error rates for the bits that constitute a given symbol, is investigated.

Results are presented in terms of the error rates of the properties of the transmission model, the quality of decoded video subject to noise in the compressed bitstream, and in terms of video transmitted through the overall system.
# Table Of Contents

Thesis Access Form .................................................................................................................... i
Video Partitioning for Wireless Applications .......................................................................... ii
Certificate of Originality .......................................................................................................... iii
Acknowledgements .................................................................................................................. iv
Abstract .................................................................................................................................... v
Table Of Contents ................................................................................................................... vi
Table Of Figures .................................................................................................................... ix
Table Of Tables ....................................................................................................................... xv
Glossary ......................................................................................................................................... xvii
Scope Of Thesis ........................................................................................................................ xviii
  Organisation of Thesis .......................................................................................................... xviii
  Original Contribution .......................................................................................................... xx

Chapter One: Introduction to Digital Data Coding ................................................................... 2
  Data Compression .................................................................................................................. 3
    Lossless Compression Algorithms .................................................................................... 5
    Lossy Compression Algorithms ...................................................................................... 12
  Still Image Coding .............................................................................................................. 12
    JPEG .................................................................................................................................. 13
  Video Coding ....................................................................................................................... 17
    DPCM loop ....................................................................................................................... 18
    Intra- / Inter-frame coding ............................................................................................ 18
    Motion Estimation .......................................................................................................... 19
    DCT Filtering .................................................................................................................. 24
    Buffering ......................................................................................................................... 25
  Video Coding Standards ..................................................................................................... 25
    H.261 ................................................................................................................................. 25
    MPEG-I,II ........................................................................................................................ 26
    H.263 ................................................................................................................................. 26
  Comparison of Features ....................................................................................................... 26
  The MPEG-II Video Coding Standard .............................................................................. 27
Summary ..................................................................................................................................... 32

Chapter Two: Transmission of Compressed Video ................................................................ 34
  Multi-bit per symbol with equalising error correcting codes .......................................... 34
  Retransmission of damaged data ........................................................................................ 35
  Variable-rate transmission .................................................................................................. 37
  Prioritised video bitstreams via sub-band coding ............................................................ 38
  Hierarchical encoding via fixed zonal mask ..................................................................... 41
Summary ..................................................................................................................................... 42
Chapter Four: Error Resilience in MPEG-II .......................................................... 76

- MPEG-II Statistics .................................................................................................. 76
  - Decoder Error Resilience and Synchronisation .................................................... 79
  - Persistence of Errors ............................................................................................. 84

- VLC Synchronisation ............................................................................................. 84

- MPEG-II Transcoder ............................................................................................ 87
  - DCT Interleaving .................................................................................................. 88
  - Syntax Protection ................................................................................................. 96
  - EREC Transcoding ............................................................................................. 97
  - Development of Compression Standards ............................................................ 99

- Error Concealment ............................................................................................... 99

Summary .................................................................................................................. 101

Chapter Five: Partitioning for Wireless Applications ........................................... 104

- Targeting of Error-Rate Prioritised Channels ....................................................... 105
  - Matched Error-Rates via ECC ............................................................................. 105
  - Error-Rate Based Priority Partitioning ................................................................. 105

- Partitioning ............................................................................................................ 106
  - Partitioning Equations ......................................................................................... 108
  - Recombining Equations ....................................................................................... 109
  - Implementation .................................................................................................... 109

- Error Detection ...................................................................................................... 110

- Partitioning Algorithms ........................................................................................ 112

- Partitioning Performance ....................................................................................... 116

- Partitioning Efficiency .......................................................................................... 139
  - Transcoding Efficiency and Bit-rate Control ....................................................... 140
  - Partitioning B-frames to improve efficiency ....................................................... 143

Summary .................................................................................................................. 150

Chapter Six: MPEGII Transmission System ......................................................... 153

- Overall System Model ........................................................................................... 153
- Choice of System Parameters ................................................................................ 154

Efficiency .................................................................................................................. 157
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Example of Run-length coding</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>Example of Huffman coding</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>Example of Arithmetic coding</td>
<td>9</td>
</tr>
<tr>
<td>4</td>
<td>Baseline JPEG encoder</td>
<td>13</td>
</tr>
<tr>
<td>5</td>
<td>Uniform symmetric quantizers : mid-tread and mid-riser</td>
<td>15</td>
</tr>
<tr>
<td>6</td>
<td>Default MPEG-II Quantization matrix (W) (Intra coded blocks)</td>
<td>16</td>
</tr>
<tr>
<td>7</td>
<td>Quantizers used in MPEG-II : intra coded data, non-intra coded data</td>
<td>16</td>
</tr>
<tr>
<td>8</td>
<td>Default Coefficient Scanning Pattern (non-interlaced frames)</td>
<td>17</td>
</tr>
<tr>
<td>9</td>
<td>Alternate Coefficient Scanning Pattern (interlaced frames) and scanning pattern within one interlaced field</td>
<td>17</td>
</tr>
<tr>
<td>10</td>
<td>Basic DPCM loop for video compression</td>
<td>18</td>
</tr>
<tr>
<td>11</td>
<td>Block matching motion estimation</td>
<td>19</td>
</tr>
<tr>
<td>12</td>
<td>Frame differences from block matching motion estimation algorithms (MAD, MSD) for Mobile &amp; Calendar, prediction of frame 2 from 1, 16x16 pixel blocks, ±15 pixel search window</td>
<td>20</td>
</tr>
<tr>
<td>13</td>
<td>Linear interpolation of pixel intensity</td>
<td>21</td>
</tr>
<tr>
<td>14</td>
<td>Pel-recursive motion calculation using 5x2 block correlation coefficient : Mobile and Calendar, frame 1 to 2, location {400,100}</td>
<td>22</td>
</tr>
<tr>
<td>15</td>
<td>Pel-recursive motion calculation using 4x4 block MAD : Mobile and Calendar, frame 1 to 2, location {400,100}</td>
<td>22</td>
</tr>
<tr>
<td>16</td>
<td>Hybrid motion estimation using MAD of 16x16 pixel blocks, motion restricted to ±15 pixels. Difference between frame 2 and its estimate</td>
<td>23</td>
</tr>
<tr>
<td>17</td>
<td>Zonal mask applied to DCT coefficients (non-interlaced frames)</td>
<td>24</td>
</tr>
<tr>
<td>18</td>
<td>Hierarchical content of an MPEG-II bitstream</td>
<td>28</td>
</tr>
<tr>
<td>19</td>
<td>Sequence and Group Of Pictures layers of an MPEG-II bitstream</td>
<td>28</td>
</tr>
<tr>
<td>20</td>
<td>Picture, slice and macroblock layers of an MPEG-II bitstream</td>
<td>29</td>
</tr>
</tbody>
</table>
Figure 21: Re-ordering of frames to enable bi-directional prediction ........................................ 30
Figure 22: Macroblock layer, showing block structure for 4:2:0 format video .................................. 31
Figure 23: Error-rate equalising error-correction for hierarchical modulation .................................. 35
Figure 24: Video encoding showing revised frame order after errors are reported (storage space reduced by halving the intra-frame rate) ................................................................. 36
Figure 25: Variable rate QAM modulation scheme: from 1 bit/symbol to 6 bits/symbol .................. 37
Figure 26: Selection of modulation scheme against fading level .................................................... 38
Figure 27: Two-dimensional (separable) sub-band filtering ............................................................. 39
Figure 28: Sub-band decomposition of Mobile & Calendar, frame #1 ................................................ 40
Figure 29: H.263 compatible sub-band coding hierarchical video encoder ...................................... 40
Figure 30: Hierarchical image encoding for hierarchical modulation ............................................... 41
Figure 31: Multipath (Mobile Radio) Scenario .................................................................................. 47
Figure 32: Rayleigh Distribution for N=10 ...................................................................................... 48
Figure 33: Fading effect of a Rayleigh channel ($f_d T_s = 0.01$) ...................................................... 50
Figure 34: PSK ............................................................................................................................... 51
Figure 35: QPSK with Gray-coded bit-symbol assignments ............................................................. 52
Figure 36a: 16-QAM (Type I) ........................................................................................................... 53
Figure 37: $P(e)$ for PSK, QPSK and QAM16 Type-III ($B=3A$) via a Rayleigh-fading Channel .......... 56
Figure 38: BER Effect of Modulation Depth for QAM16 Type III via a Rayleigh Fading Channel at 18 dB ..................................................................................................................... 56
Figure 39: Comparison of QAM16 Types II and III ....................................................................... 57
Figure 40: Simple Transmission Model ............................................................................................. 57
Figure 41: Optimal Filter Arrangement for AWGN channel ............................................................. 58
Figure 42a: Frequency Response of Raised Cosine Filter ................................................................. 59
Figure 43a: Frequency Response of Square-Root Raised Cosine Filter ........................................... 60
Figure 44: Eye diagrams after square-root raised cosine filter (applied once and twice). 
(α=0.35, 41 filter taps : timebase reflects at edges of graph) ........................................ 61

Figure 45: Automatic Gain Control .................................................................................. 62

Figure 46: Bit-error rate (bits c, d) vs. f_dT_s for AGC (γ=∞) ................................................. 62

Figure 47: Pilot Assisted Fade Tracking ......................................................................... 63

Figure 48: Bit-error rate vs. f_dT_s for Pilot Assisted AGC (γ=∞) ........................................ 64

Figure 49: Pilot Assisted Fade Tracking with 2nd Order (Spatial) Diversity ................. 65

Figure 50: Comparison of Diversity and Non-Diversity Schemes (f_dT_s=0.001, Pilot Spacing = 10) .......................................................... 66

Figure 51: Comparison of Non-diversity and Diversity schemes in terms of distance travelled between received pilots at 20dB (no differential phase encoding) ........... 67

Figure 52: Error Correcting Capability of BCH (63, k, t) code ........................................ 70

Figure 53: Error Correcting Capability of BCH (255, k, t) code ........................................ 70

Figure 54: Error Correcting Capability of RS (15, k, t) code (4-bit symbols) ................. 71

Figure 55: Error Correcting Capability of RS (63, k, t) code (6-bit symbols) .................... 72

Figure 56: Error Correcting Capability of RS (255, k, t) code (8-bit symbols) ................. 72

Figure 57: Effect of Errors in MPEG-II Coded Bitstreams (Mobile & Calendar coded at 4Mbits/s) ................................................................................................................ 79

Figure 58: Effect of Errors in MPEG-II Coded Bitstreams (Football coded at 4Mbits/s) .... 80

Figure 59: Effect of Errors in an MPEG-II Coded Bitstream (Mobile & Calendar coded at 4Mbits/s, Error Rate = 0.005%) ................................................................. 81

Figure 60a: PSNR vs. BER for different bitstream synchronisation points (Intra-frames) ................................................................. 84

Figure 61: PSNR vs. Number of Retained DCT Coefficients (Frame #1 of Mobile & Calendar) ........................................................................................................ 85

Figure 62: Image Detail with Indicated Number of DCT Coefficients Retained .............. 86

Figure 63: Mobile & Calendar with 5 DCT Coefficients Retained .................................. 86

Figure 64: Transcoding .................................................................................................. 87

Figure 65: Existing macroblock layout .............................................................. 88
Figure 66: Interleaved macroblock layout ................................................................. 88
Figure 67: Macroblocks interleaved across a slice ..................................................... 89
Figure 68: Probability of Decoder Failure at/before bit $j$ ......................................... 90
Figure 69: Effect of Errors on Interleaved MPEG-II Bitstreams (Mobile & Calendar
coded at 4Mbits/s) ............................................................................................... 91
Figure 70: Effect of Errors on Interleaved MPEG-II Bitstreams (Football coded at
4Mbits/s) .................................................................................................................. 91
Figure 71: Effect of Errors in an Interleaved MPEG-II Coded Bitstream (Mobile &
Calendar coded at 4Mbits/s, Error Rate = 0.005%) ............................................... 93
Figure 72a: EREC coding ......................................................................................... 97
Figure 73: Bitstream Partitioning ............................................................................... 104
Figure 74: Partitioning of interleaved DCT coefficients ............................................ 113
Figure 75: Partitioning of interleaved DCT coefficients ............................................ 113
Figure 76: Slice partitioning of interleaved DCT coefficients .................................... 115
Figure 77: Average PSNR vs. Partition 2 bit-error rate [1:1 partitioning] (Mobile &
Calendar coded at 4Mbits/s, Mode 0 Reconstruction, various Macroblock
Groupings) ............................................................................................................... 116
Figure 78: Average PSNR vs. Partition 2 bit-error rate [1:1 partitioning] (Mobile &
Calendar coded at 4Mbits/s, Mode 1 Reconstruction, various Macroblock
Groupings) ............................................................................................................... 117
Figure 79: PSNR vs. BER with validation pattern in P2 Dead-Time. (Mobile &
Calendar coded at 4Mbits/s, 1:1 partitioning, Mode 1 Reconstruction, various
Macroblock Groupings) ......................................................................................... 118
Figure 80: PSNR vs. BER Comparison of Slice Interleaving and Macroblock Grouping
of the entire slice. (Mobile & Calendar coded at 4Mbits/s, 1:1 partitioning) .......... 118
Figure 81: Effect of Partition 2 Errors on Partitioned MPEG-II Coded Bitstreams
(frames and difference from error free sequence) (Mobile & Calendar coded at
4Mbits/s, P2 Error Rate = 0.1%, 1:1 Partitioning, Mode 0 Reconstruction) .......... 121
Figure 82: Effect of Partition 2 Errors on Partitioned MPEG-II Coded Bitstreams
(frames and difference from original sequence) (Mobile & Calendar coded at
4Mbits/s, P2 Error Rate = 0.01%, 1:1 Partitioning, Mode 1 Reconstruction) .......... 124
Figure 83: Effect of Partition 2 Errors on Partitioned MPEG-II Coded Bitstreams (frames and difference from original sequence) (Mobile & Calendar coded at 4Mbits/s, P2 Error Rate = 0.1%, 1:1 Partitioning, Mode 1 Reconstruction) ........... 127

Figure 84: Effect of Partition 2 Errors on Partitioned MPEG-II Coded Bitstreams (Mobile & Calendar coded at 4Mbits/s, P2 Error Rate = 1%, 1:1 Partitioning, Mode 1 Reconstruction) ................................................................. 130

Figure 85: Effect of Partition 2 Errors on Partitioned MPEG-II Coded Bitstreams (frames and difference from original sequence) (Mobile & Calendar coded at 4Mbits/s, P2 Error Rate = 0.1%, 1:1 Partitioning, Mode 1 Reconstruction with Dead-Time Validation) ........................................................................................... 133

Figure 86: Effect of Partition 2 Errors on Partitioned MPEG-II Coded Bitstreams (frames and difference from original sequence) (Mobile & Calendar coded at 4Mbits/s, P2 Error Rate = 0.1%, 1:1 Partitioning, Slice Interleaving) .................... 136

Figure 87: Efficiency of Partitioning for various Macroblock groupings (Mobile & Calendar, frames 1-51, coding with B-frames). Contrasted with efficiency of BCH error correcting code on partition 2.................................................................. 139

Figure 88: Efficiency of Partitioning for various Macroblock groupings (HDTV sequence GreenLeaf, frames 1-20, coded with B-frames). Contrasted with efficiency of BCH error correcting code on partition 2....................................................... 140

Figure 89: Fluctuation in ε and overheads vs. Frame Number (Mobile and Calendar, coded at 4Mbits/s, κ = 1.00, Slice Interleaving) ................................................................. 142

Figure 90: Effect of controlling the transmission bit-rate outside the DPCM loop (frames 1-51 of Mobile and Calendar, coded at 4Mbits/s) .............................................. 143

Figure 91: Average PSNR vs. Partition 2 bit-error rate [1:1 partitioning, Mode 0 reconstruction] (frames 1-51 of Mobile and Calendar coded at 4Mbits/s, B-frame headers partitioned)................................................................. 145

Figure 92: Average PSNR vs. Partition 2 bit-error rate [1:1 partitioning, Mode 1 reconstruction] (frames 1-51 of Mobile and Calendar coded at 4Mbits/s, B-frame headers partitioned)........................................................................ 146

Figure 93: Efficiency of Partitioning vs. Compression Ratio with B-frame headers partitioned (frames 1-51 of Mobile and Calendar, coded at 4Mbits/s) .................... 148

Figure 94: Effect of Partition 2 Errors on Partitioned MPEG-II Coded Bitstreams (frames and difference from original sequence) (Mobile & Calendar coded at 4Mbits/s, P2 Error Rate = 0.1%, 1:1 Partitioning, B-frame headers partitioned) ... 147

Figure 95: Fluctuation in ε and overheads vs. Frame Number (Mobile and Calendar, coded at 4Mbits/s, κ = 1.00, B-frame headers split, 44 macroblocks interleaved)........................................................................... 150
Figure 96: Complete transmission system............................................................................ 153

Figure 97: Effect on Bit-error rate of BCH code interleaving period................................. 155

Figure 98: Effect on Partition 1 Bit-error rate of various error correcting codes (with interleaving). ............................................................................................................ 156

Figure 99: Effect on Partition 2 Bit-error rate of various error correcting codes (without interleaving). ............................................................................................................ 156

Figure 100: PSNR vs. Channel quality for frames 1-50 of Mobile & Calendar, BCH(63,39,4) on P1, BCH(63,57,1) on P2, slice interleaved, κ=1, transmitted at 1.33Msymbols/sec............................................................................................................ 158

Figure 101: PSNR vs. Channel quality for frames 1-50 of Football, BCH(63,39,4) on P1, BCH(63,57,1) on P2, slice interleaved, κ=1, transmitted at 1.33Msymbols/sec............................................................................................................ 159

Figure 102: End-to-end system results (Mobile & Calendar transmitted at 1.33Mbits/s, γ=20dB, P1 BCH(63,39,4), P2 BCH(63,57,1), slice interleaving with mode 1 reconstruction)............................................................................................................ 160

Figure 103: End-to-end system results (Mobile & Calendar transmitted at 1.33Mbits/s, γ=24dB, P1 BCH(63,39,4), P2 BCH(63,57,1), slice interleaving with mode 1 reconstruction)............................................................................................................ 161

Figure 104: PSNR vs. Channel quality for frames 1-50 of Mobile & Calendar, BCH(63,30,6) on P1, BCH(63,57,1) on P2, slice interleaved, κ=1, transmitted at 1.33Msymbols/sec............................................................................................................ 162

Figure 105: PSNR vs. Channel quality for frames 1-50 of Mobile & Calendar, BCH(63,39,4) on P1, no P2 error correction, slice interleaved, κ=1, transmitted at 1.33Msymbols/sec............................................................................................................ 163

Figure 106: PSNR vs. Channel quality for frames 1-50 of Mobile & Calendar, BCH(63,39,4) on P1, BCH(63,57,1) on P2, 44 macroblocks interleaved, κ=1.05, transmitted at 1.33Msymbols/sec............................................................................................................ 163

Figure 107: PSNR vs. Channel quality for frames 1-50 of Mobile & Calendar, BCH(63,39,4) on P1, BCH(63,57,1) on P2, 11 macroblocks interleaved, κ=1.05, transmitted at 1.33Msymbols/sec............................................................................................................ 164

Figure 108: Comparison of images with near identical PSNR (from frame 11 of Mobile & Calendar). ............................................................................................................ 188
Table Of Tables

Table 1 : Improving the efficiency by coding two bits per symbol ........................................... 5
Table 2 : Improving the efficiency by coding three bits per symbol ......................................... 5
Table 3 : Average bits per symbol for a Huffman code ............................................................. 8
Table 4 : Result of arithmetic coding ...................................................................................... 10
Table 5 : Common image file formats and compression techniques utilised .......................... 12
Table 6 : Comparison of JPEG, H.261, MPEG-I, MPEG-II and H.263 encoders ................. 26
Table 7 : Example error correcting code ................................................................................. 68
Table 8 : MPEG-II Bitstream Contents ................................................................................... 77
Table 9 : Affect of MPEG-II Bitstream Errors ........................................................................ 78
Table 10 : Summary of error concealment strategies .............................................................. 100
Table 11 : Knowledge of bitstream properties in the partitioning and re-combining processes .................................................................................................................. 108
Table 12 : Percentage of EOB codes occurring in partition 1, for frames 1-51 of Mobile & Calendar coded at 4Mbits/s ................................................................. 114
Table 13 : Summary of included video sequence figures ........................................................ 120
Table 14 : Average Partitioning Overheads, for frames 1-51 of Mobile & Calendar coded at 4Mbits/s ................................................................. 144
Table 15 : Distribution of bits in B-frame macroblock headers (Mobile & Calendar, frames 1-51, coded at 4Mbits/s) .............................................................................. 144
Table 16 : Bandwidth efficiency for various systems ................................................................. 157
Table 17 : MPEG-II bitstream structure .................................................................................. 172
Table 18 : Sequence header fields .......................................................................................... 173
Table 19 : Group of Pictures data fields .................................................................................. 173
Table 20 : Picture header fields .............................................................................................. 174
Table 21 : Picture data fields ................................................................................................. 174
**Glossary**

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWGN</td>
<td>Additive White Gaussian Noise</td>
</tr>
</tbody>
</table>
| BCH | Bose-Chaudhuri-Hocquenghem (error correcting code)  
Also referred to in \( (n,k,t) \) notation: \( k \) data bits, encoded as \( n \) bits, capable of correcting \( t \) or fewer errors |
| BER | Bit Error Rate |
| DAT | Digital Audio Tape |
| DCT | Discrete Cosine Transform |
| EC | Error Correction (use of error correction information) |
| EOB | End Of Block |
| FEC | Forward Error Correcting Code (addition of error correction information) |
| ISI | Inter-Symbol Interference |
| JPEG | Joint Picture Experts Group |
| MB | Macroblock |
| MB Grouping \( n \) | A collection of \( n \) macroblocks interlaved together as a whole |
| MPEG | Motion Picture Experts Group |
| PSNR | Peak Signal to Noise Ratio |
| PSTN | Public Switched Telephone Network |
| QAM | Quadrature Amplitude Modulation |
| RS | Reed-Solomon (error correcting code)  
Also referred to in \( (n,k,t) \) notation: \( k \) data symbols, encoded as \( n \) symbols, capable of correcting \( t \) or fewer symbol errors |
| VLC | Variable Length Code |
Scope Of Thesis

This thesis is concerned with the transmission of compressed video bitstreams through noisy communication channels. To achieve optimal performance, it is usually necessary to develop a compression / coding algorithm that matches the properties of the source data to the channel (joint ‘source-channel’ coding), but this is not always practical. Certain coding models are becoming accepted as world-side standards, and real-time (silicon) implementations are already beginning to appear. There are advantages to be gained from being able to use the output from these existing standards as the basis for transmission in scenarios for which they were not specifically designed, for example:-

- the cost of developing entirely new coding models is high.

- only the compressed form of the video need be sent to the transmitter, not the complete unencoded video. This reduces the bandwidth of this link significantly.

- compatibility with existing systems using these coding models is retained.

Although dedicated solutions are always likely to perform better, this thesis considers the compromise solution of transformation of the pre-compressed video bitstream into a form more suited to transmission. With careful design of the transformation process, it is possible to match the properties of the transmitted data to the properties of the channel (which, for this thesis, is the particularly harsh mobile radio environment).

Organisation of Thesis

Chapter One provides an introduction to digital video coding, starting from the principles of digital data compression, and ending with an outline of the techniques used in video coding algorithms. A brief outline of the major international standards for video encoding is included, before the specific structure of MPEG-II is considered.
Chapter Two presents a review of approaches to compressed video transmission that already exist in the literature. The majority of these approaches require feedback from the receiver to the video encoder, thus making broadcast applications impossible.

Chapter Three develops a model for the simulation of a severe transmission channel. Multi-level modulation schemes, filters, fade-tracking algorithms and error correcting codes are all considered as part of the simulation of efficient transmission to a mobile radio receiver. The error-rates from the simulation are compared with theoretical expressions, in the assumption that the receiver tracks the fading perfectly, before realistic algorithms for fade-tracking are discussed.

Chapter Four considers how the video decoding process reacts to errors in the bitstream, and how the error-resilience of the compressed video can be improved upon without affecting the data-rate. The concept of coefficient interleaving is introduced to effectively increase the number of decoder synchronisation points in the bitstream, without adding to the amount of data transmitted. An alternative solution (developed in parallel by other researchers) is described for completeness.

Error-concealment strategies that already exist are outlined, though they are not utilised in this work.

Chapter Five introduces an algorithm for dividing the more error-resilient version of the video bitstream into sub-channels of different levels of visual significance, thus making it more suited to the transmission channels outlined in Chapter Three. The ability to divide the data into an arbitrary ratio enables the use of different levels of error correction on the different sub-channels. Three different algorithms for detecting errors in the received data are investigated, and methods for improving the efficiency of the algorithm are investigated. Finally, algorithms for making the output bit-rate constant are considered.

Chapter Six outlines an overall transmission system for broadcast of MPEG-II coded bitstreams. The video is first made more resilient to errors, and then divided into two constant rate sub-channels. These are then protected to different degrees by error correcting
codes, before being transmitted through a Rayleigh fading mobile radio environment, using a 16 level modulation scheme and fade tracking.

Based on the bit-error rate requirements discussed in Chapter Four, and from simulations of the transmission model, error correcting codes are chosen to provide intelligible video for a channel noise ratio (CNR) > 18dB.

Chapter Seven presents the conclusions of this thesis, and provides a number of suggestions as to how the system described could be improved upon.

Original Contribution

The fundamental concepts introduced in this thesis are those of:-

i) Transcoding an MPEG-II video bitstream to match the properties associated with transmission, without requiring the development of an entirely new video encoder/decoder pair.

ii) Interleaving the DCT coefficients to significantly increase the resilience of the compressed bitstream to transmission errors, without adding to the amount of data transmitted.

These are both introduced in Chapter Four, before being expanded upon in Chapter Five where the properties inherent in DCT interleaving are shown to be those required for partitioning the video bitstream for transmission through channels comprising of different error-resilience sub-channels. After investigating efficiency considerations associated with the partitioning process, and introducing effective solutions that enable the transmitted data-rate to be matched to the bit-rate of the original video bitstream, an end-to-end video transmission system is developed in Chapter Six. This system makes more efficient use of the bandwidth available by reducing the amount of error-correcting information required. The quality of the reconstructed video is shown to decrease steadily as the channel quality worsens, until the critical syntactic structure of the bitstream is damaged, making the system suited to environments where the channel quality fluctuates.
Chapter One: Introduction to Digital Data Coding
Chapter One: Introduction to Digital Data Coding

A digital signal is inherently less susceptible to noise than an analogue signal, as re-quantization of the signal to the original digital levels removes large amounts of the noise. Consequently, there are significant advantages in representing, storing and transmitting signals in digital form. Recent advances in the very large scale integration (VLSI) of electronic circuits has led to the mass production of digital circuits consisting of millions of transistors, which is enabling the development of consumer electronic products for:

- digital audio (CD, DAT)
- digital radio
- digital television

Of these, digital audio in the form of compact discs (CDs) has been available for several years, digital radio has recently been launched, and digital television is due in the United Kingdom in the next couple of years. The major problem with the digital representation of these signals (which is the primary reason why digital television has not been in existence for some time) is the transmission bandwidth requirement:

For existing broadcast quality video: image resolution ~ 720x576,

16 bits per pixel\(^1\),

25 frames per second,

giving a bandwidth of about 160Mbits/s per channel [1].

This does not compare well with the bandwidth currently assigned in the UK for analogue television broadcasting (c. 5.5MHz [2]). Modern VLSI devices are capable of supporting

---

\(^1\) Due to the way the human visual system works, it is sufficient to represent video with luminance levels sampled to 8-bit accuracy, and chrominance values sampled at 8-bit accuracy but with only half the spatial resolution. On average, this gives 8+4+4=16 bits per pixel.
complex algorithms which can be used to process such vast quantities of data in real-time, which makes two important information theoretic concepts available for data transmission:

- compression
- error correction / detection

This chapter discusses the techniques that are available for compression of the information which makes digital broadcasting a reality: error correction is discussed in Chapter Three where properties of transmission systems are discussed.

To make digital television an efficient reality requires consistent compression in the order of 30 times: put in other terms, each pixel of each frame needs to be represented (on average) by 0.5 bits. This is a particularly strict requirement which is achieved through a combination of:

- lossless compression (where the uncompressed data is an exact copy of the original data)
- lossy compression (where the uncompressed data is an approximation to the original data)

This chapter begins by discussing the mathematical properties and the various techniques that exist for lossless data compression. Techniques for lossy compression are outlined with particular reference to digital video, and finally, video compression standards that are already in existence are outlined.

Data Compression

If there are $n$ discrete levels transmitted at intervals of $\lambda$s, the number of possible signal combinations transmitted in $T$ seconds is $n^{\lambda T}$. Since the amount of information transmitted must be proportional to the time of transmission, the amount of information transmitted is defined as the logarithm of the number of possible signal combinations:
information \propto \log\left(n^\frac{1}{n}\right) = \frac{T}{\lambda} \log(n)

If each of the \( n \) symbols are equally probable, then the total amount of information transmitted per symbol period is given by :-

\[
\text{information} = \sum p_i \log_k (n) = -\sum p_i \log_k (p_i)
\]

The base of the logarithm is chosen so that the unit of information is the bit. If there are two equiprobable symbols, '0' and '1', then:-

\[
\text{information} = -(\frac{1}{2} \log_2 \frac{1}{2} + \frac{1}{2} \log_2 \frac{1}{2}) = -\log_2 \frac{1}{2} = \log_2 2 = 1
\]

*Therefore, k = 2, so information = \(-\sum p_i \log_2 (p_i)\) ................. Equation 1*

Noting the similarity of equation 1 with the expression for entropy defined in statistical mechanics ([3] for example), the amount of information conveyed is also known as the entropy of the data, \( H \), defined as :-

\[
H = -\sum p_i \cdot \log_2 p_i
\]

For example, if '0' occurred 75% of the time, then the entropy, \( H \), is given by:-

\[
H = -(\frac{3}{4} \log_2 \frac{3}{4} + \frac{1}{4} \log_2 \frac{1}{4}) = 0.81
\]

and so there is less information conveyed than when the symbols are equally probable.

Encoding this with a single bit code would be inherently inefficient: each symbol would require a single bit resulting in (defining the efficiency, \( \eta = \frac{H}{\text{ave bits}} \)) \( \eta = 0.81 \). Encoding pairs of bits leads to:-
Table 1: Improving the efficiency by coding two bits per symbol

which is closer to the entropy of the original data (η=0.96). Encoding 3 bits as a single code gives (η=0.98):

Table 2: Improving the efficiency by coding three bits per symbol

These examples show that as the length of the string of bits encoded is increased, so the efficiency of the code can be improved. As the number of bits encoded is increased, the average number of bits will approach the entropy, but cannot improve upon it.

Lossless Compression Algorithms

The principle of lossless compression is to reduce the number of data bits transmitted whilst retaining the property that the decoded data is exactly the same as the original data. This can only be achieved because the input data contains some intrinsic redundancy. In most cases, lossless compression can only be achieved efficiently if the symbol probabilities are

---

2 The choice of how these are encoded is not important for now, though these tables have been produced using Huffman coding (see later in this chapter).
known beforehand (when the code is designed) but this is not always the case as will be shown later on.

**Run-Length Coding**

In many situations, data to be transmitted contains a single value repeated several times. One of the simplest methods to achieve compression of this data is to store a code signifying how many times the value repeats (the run-length), and a code representing the actual value itself.

\[
\begin{array}{ccccccc}
0000 & 1111111111 & 00 & 1 & \ldots
\end{array}
\]

*run of 4 zeros run of 11 ones run of 2 zeros run of 1 one*

\[\text{Figure 1: Example of Run-length coding}\]

In figure 1, the sequence \(000011111111110010\ldots\) would be encoded as \(<\text{run of 4}>0, <\text{run of 11}>1, <\text{run of 2}>0, <\text{run of 1}>1, \ldots\). In fact, when there are only two possible levels, only the initial level needs to be transmitted - the subsequent levels toggle for each run-length code. Provided that the run-lengths are encoded efficiently, compression of the bitstream results. This is the basic compression technique employed for facsimile transmission (Group III fax, [4]), where the run-length is variable length encoded based on its probability of occurrence in a set of typical documents.

The form of run-length encoding described above is known as 1-dimensional run-length encoding, as there is one degree of freedom. Some of the video coding standards described later in this chapter utilise two- or three-dimensional run-length encoding, where the other degrees of freedom represent:

- level after a run of 'length' zeros (MPEG-II)
- whether the code represents the end of transmission of this item (H.263).

The choice of whether to use these more complex run-length encoding schemes is determined by the statistics of the data being compressed. For instance, in H.263 encoding it is quite probable that a code will represent the last piece of information for this 8x8 pixel block, and
so it is efficient to encode this information as part of the code. This is not true for MPEG-II encoders, and so two-dimensional run-length encoding is used.

**Prefix Coding**

In order to decode any form of compressed data, the compressed form must be uniquely decodable. A prefix code extends this a little further: *a prefix code is defined as a code in which no codeword is the prefix of any other codeword*. Note that a prefix code is always uniquely decodable, but the converse is not necessarily true. From the definition, a prefix code can be decoded as soon as the bit sequence representing an end of code is received, or alternatively, when the bits received so far form a word that is part of the codebook.

**Huffman Coding**

Huffman codes result in the shortest average code length of all statistical coding techniques. They are prefix codes, and so are uniquely decipherable. The basic algorithm, outlined in figure 2, is a two stage process:

1) sort the symbols in decreasing order of probability of occurrence

2) combine the lowest two symbols, assigning the bits '1' and '0' to differentiate between these two symbols (the choice of which symbol is indicated by '0' is arbitrary).

This is repeated until there is only one symbol left. The only subtlety with this algorithm is when the probabilities at a node are equal. In this case, it makes no difference which symbol is placed first as the average number of bits will be the same. Typically, the Huffman code with the least variance\(^3\) in the code length is chosen. This is produced when the symbols are primarily sorted by probability, and when the probabilities are the same, they are sorted by the length of the code assigned so far [5].

---

\(^3\) The variance in the code length is given by: \[ \text{var} = \sum_{\text{codes}} p \cdot (\text{code len} - \text{ave bits per symbol})^2 \]
The resulting codewords for the Huffman code of figure 2 are given in table 3 below. This table also shows the average number of bits per symbol required for this code, which compares well with the entropy, $H=1.78$. The average bits per symbol can only equal the entropy when the symbol probabilities are $2^{-n}$ (integer $n$).

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Probability</th>
<th>Code</th>
<th>Length</th>
<th>Ave. bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X_1$</td>
<td>0.50</td>
<td>0</td>
<td>1</td>
<td>0.50</td>
</tr>
<tr>
<td>$X_2$</td>
<td>0.20</td>
<td>11</td>
<td>2</td>
<td>0.40</td>
</tr>
<tr>
<td>$X_3$</td>
<td>0.18</td>
<td>100</td>
<td>3</td>
<td>0.54</td>
</tr>
<tr>
<td>$X_4$</td>
<td>0.12</td>
<td>101</td>
<td>3</td>
<td>0.36</td>
</tr>
</tbody>
</table>

As the number of symbols is increased, so the length of (some of) the Huffman codewords increases. Although optimal, the Huffman code so produced may not be desirable from an implementation point of view. To restrict the maximum length codeword, the codebook is usually modified to include a code which combines several low probability codes which are encoded by a fixed length code that follows. As this modification is usually reserved for low probability symbols, the effect on the average number of bits per symbol is small.

**Arithmetic Coding**

Arithmetic coding is a statistical compression technique that can improve upon Huffman coding when the symbol probabilities are not integer powers of $\frac{1}{2}$. Each symbol is assigned an interval between 0 and 1, whose width equals the probability of that symbol occurring. The second symbol of the message is encoded by further subdividing this interval,
resulting in the entire message being represented by a single floating point number (which is any value that lies within the interval at the end of the message).

As figure 3 shows, the accuracy required of the floating point number that represents the entire message increases as the message length increases, which does not lend the algorithm to implementation. As a result, this approach remained no more than an interesting theoretical technique until implementations based on integer arithmetic were developed in [6]. These implementations are based on outputting the part of the representation that is common to all values within the interval, and then re-expanding the interval (making sure that multiple symbols do not map to the same integer value). In the example above⁴, all possible values of the message $X_4, X_2, X_5$ lie between 0.504 and 0.520, and so the “0.5” can now be output, leaving a new range of 0.04 to 0.20.

After $n$ symbols of the message the width of the interval, $I$ (which is the probability of this message occurring, $p_{message}$), is given by $I = \prod_{i=1}^{n} p_i$. The number of bits required to

---

⁴ This example uses decimal arithmetic for clarity. Implementations of this algorithm are usually based on binary fractions.
represent this message is the number of bits required to uniquely define a number within this
interval. This is given (depending on where the interval begins and ends\textsuperscript{5}) by :-

\[ \text{bits} \leq 1 + \text{ceil}(\log_2 I) \] \hspace{1cm} \text{Equation 2}

where \text{ceil}(x) is defined to be the nearest integer value greater than or equal to \( x \).

When the message length \( n \) increases, the \text{ceil} function has less effect\textsuperscript{6}, the "1+" term of
equation 2 becomes less significant as \( -\log_2 I \) increases, and so the average number of bits
is given by :-

\[ \sum_{\text{all messages}} p_{\text{message}} \cdot \text{bits(message)} \xrightarrow{n \to \infty} \sum_{\text{all messages}} p_{\text{message}} \cdot -\log_2 p_{\text{message}} \]

which is the entropy of the original message.

To illustrate how this algorithm works, the 3-bit code of table 2 (page 5) is arithmetic
coded in table 4 (for clarity, the lower and upper bounds are shown in decimal notation, not as
binary fractions) :-

<table>
<thead>
<tr>
<th>Code</th>
<th>Probability</th>
<th>Lower bound</th>
<th>Upper bound</th>
<th>Coded as</th>
<th>Ave bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>27/64</td>
<td>0</td>
<td>0.421875</td>
<td>00</td>
<td>2x27/64</td>
</tr>
<tr>
<td>001</td>
<td>9/64</td>
<td>0.421875</td>
<td>0.5625</td>
<td>100</td>
<td>3x9/64</td>
</tr>
<tr>
<td>010</td>
<td>9/64</td>
<td>0.5625</td>
<td>0.703125</td>
<td>1010</td>
<td>4x9/64</td>
</tr>
<tr>
<td>011</td>
<td>3/64</td>
<td>0.703125</td>
<td>0.75</td>
<td>10111</td>
<td>5x3/64</td>
</tr>
<tr>
<td>100</td>
<td>9/64</td>
<td>0.75</td>
<td>0.890625</td>
<td>110</td>
<td>3x9/64</td>
</tr>
<tr>
<td>101</td>
<td>3/64</td>
<td>0.890625</td>
<td>0.9375</td>
<td>11101</td>
<td>5x3/64</td>
</tr>
<tr>
<td>110</td>
<td>3/64</td>
<td>0.9375</td>
<td>0.984375</td>
<td>11110</td>
<td>5x3/64</td>
</tr>
<tr>
<td>111</td>
<td>1/64</td>
<td>0.984375</td>
<td>1</td>
<td>111111</td>
<td>6x1/64</td>
</tr>
</tbody>
</table>

Total (per bit) 1.016

Table 4: Result of arithmetic coding

\textsuperscript{5} The exact value depends on where the interval begins. For example, an interval of 0.1 from 0.25 to 0.35
requires 2 digits to uniquely specify it. An interval of the same width from 0.2 to 0.3 would only require 1
digit.

\textsuperscript{6} \text{ceil}(x) - x < 1, so as \( x \to \infty \), \( \frac{\text{ceil}(x)}{x} \to 1 \)
There are two codes highlighted that are inefficient (they could be uniquely coded with one fewer bit, but the result of this would be a representation that did not fall uniquely within the bounds set), which is purely due to the length of the original message being short. Although less efficient than Huffman coding for messages this short, as the message length increases the efficiency improves, and arithmetic coding has the significant advantage of not requiring a codebook at the decoder: the symbol probabilities are all that is required. This enables compression of much longer messages than Huffman coding could realistically provide.

**Lempel-Ziv Compression**

Although Huffman and Arithmetic coding can in theory approach the ideal case where the code lengths are equal to the amount of information carried by the code, they require *a priori* knowledge of the statistics of the data being coded. The Huffman codes are usually constrained to be local codes (else the codebook becomes prohibitively large) and so can miss higher-order relationships in the data being compressed.

In order to overcome these deficiencies, the Lempel-Ziv algorithm [7, 8] is an adaptive process that builds the codebook based on the data being compressed. Starting with an initial codebook containing the symbols 0 and 1, new symbols are created consisting of an existing symbol and an innovation bit. A new codebook entry is created which consists of the index in the codebook of the existing symbol and the innovation bit. This is assigned a unique, fixed-length index until the codebook is complete. So, starting with a codebook containing the entries 0 and 1, the bitstream 00010111001010010... is split into codes 00, 01, 011, 10, 010, 100, 101 (this process is outlined in more detail in [9]) which are then assigned codebook indexes of 0010; 0011; 1001; 0100; 1000; 1101 in the case of a 16-entry index.

LZ-compression is able to outperform Huffman coding provided that the codebook can grow sufficiently large - in the popular implementations of this compression scheme (which work on bytes rather than bits), the initial codebook contains all 256 8-bit characters with 4096 entries allocated for the index. This is inefficient at first, but as the number of
known codes increases, the more bits are encoded as one 12-bit code, and so the higher the efficiency.

**Lossy Compression Algorithms**

In lossy compression, the encoder is permitted to send an *approximation* to the original data instead of an exact copy. This inherently depends on the compression algorithm knowing the properties of the data being compressed in order to remove some of the fine detail. Such a scheme can obviously only be used where the nature of the data is such that accuracy is not critical. The performance of lossy compression schemes depends solely on the acceptable performance of the system in terms of quality of received data.

The techniques used to achieve lossy compression are best discussed in terms of particular applications. Their use in still image and video compression are outlined below, with particular reference to their use in international coding standards.

**Still Image Coding**

There are numerous still image coding techniques in everyday usage: table 5 shows an outline of some of the more common image coding standards and the compression techniques that are used in each [10, 11, 12]. The majority of these image coding techniques are lossless, with one notable exception, JPEG.

<table>
<thead>
<tr>
<th>Image Format</th>
<th>Compression Techniques</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bitmap (BMP)</td>
<td>Run-length (optional)</td>
</tr>
<tr>
<td>Metafile</td>
<td>Vector description of picture</td>
</tr>
<tr>
<td>Graphics Interchange Format (GIF)</td>
<td>Lempel-Ziv (modified)</td>
</tr>
<tr>
<td>Tagged Image (TIFF)</td>
<td>Various (Huffman, Lempel-Ziv, Fax, etc.)</td>
</tr>
<tr>
<td>JPEG</td>
<td>Frequency transform, quantization</td>
</tr>
</tbody>
</table>

*Table 5: Common image file formats and compression techniques utilised*

The JPEG coding standard [13, 14, 15] has many similarities with the video coding standards that are used throughout this thesis, though only operates on a single frame. In order to simplify the discussion of the video coding standards, the techniques used by JPEG are outlined first, before the inherent temporal redundancy of video is considered.
JPEG

JPEG specifies four algorithms for the compression of still images [16]:-

i) Sequential encoding (the 'baseline model'): the image is encoded in a raster scan (top-left to bottom-right) based on a frequency transform of the image.

ii) Progressive encoding: the image is encoded in multiple scans (at the same spatial resolution) for applications where the transmission time may be long, but the viewer needs to view the image contents in multiple coarse-to-fine stages.

iii) Lossless encoding: the decoded image is an exact replica of the original.

iv) Hierarchical encoding: the image is encoded at multiple spatial resolutions, allowing thumbnail images to be decoded without needing to transfer and decode the higher resolution image.

Of the algorithms listed above, sequential coding (the 'baseline model') is the most widely used. The encoder, shown in figure 4, is based on frequency transform, quantization, and zigzag scanning of the AC coefficients which are outlined below. Finally variable length entropy coding (usually Huffman, although extensions for using arithmetic coding exist) is used to further increase the compression ratio achieved. Compression ratios in excess of fifty times are possible, whilst still producing images of acceptable quality.

**Figure 4 : Baseline JPEG encoder**

*Transform coding has proven to be one of the most effective tools for image coding applications, though transforming a signal into the frequency domain does not in itself lead to compression of the information. What it does achieve, however, is compaction of the energy*
of the signal into a small number of coefficients. This in turn will lead to lossy compression when coefficients containing little of the energy of the original signal are coarsely quantized or even lost altogether.

Although there are a large number of discrete transforms\(^7\), almost all image / video compression schemes are based on the 2-Dimensional Discrete Cosine Transform (DCT) ([17, 18] for example), defined in equation 3 [19].

\[
X_{u,v} = \frac{2}{N} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} C_u C_v \cdot x_{i,j} \cdot \cos \left[ \frac{(2i+1)\pi\cdot u}{2N} \right] \cdot \cos \left[ \frac{(2j+1)\pi\cdot v}{2N} \right] \quad \text{Equation 3}
\]

where \( u, v = 0, 1, \ldots, N - 1 \)

\[
C_k = \begin{cases} 
\frac{1}{\sqrt{2}} & \text{if } k = 0, \\
1 & \text{otherwise.}
\end{cases}
\]

This transform has the following key properties that have established it as the de facto choice for image processing applications:-

i) the transform uses real arithmetic, is orthogonal and separable; extension to multiple dimensions is simple

ii) fast implementations of both the transform and its inverse exist

iii) it performs almost as well as the statistically optimum Karhunen-Loeve transform (KLT) [20, 21], without the need for transmission of the basis vectors. In fact, for conventional image data with a reasonably high inter-element correlation, the performance of DCT is virtually indistinguishable from that of the KLT.

iv) the energy of the signal is packed into a few coefficients.

Quantization

Having concentrated the energy of the signal into a few coefficients in the frequency domain, lossy compression is achieved primarily by quantizing the resulting coefficients that

---

\(^7\) Walsh-Hadamard, Harr, Slant, Cosine, Sine, etc.
contain significant energy, and coarsely quantizing (or even discarding) coefficients where the energy is low.

Figure 5 shows two basic quantizers, with uniformly spaced decision \((d_i)\) and representation \((r_i)\) levels. In both diagrams, an input value in the range \(0 \rightarrow d_1\) is mapped to level \(r_1\), \(d_1 \rightarrow d_2\) is mapped to \(r_2\), etc.

![Uniform symmetric quantizers: mid-tread and mid-riser.](image)

There is no fundamental reason why the decision levels should be uniformly spaced, nor that the quantizer should be symmetrical. Ideally, the design of quantizers should be based around the minimisation of a distortion measurement [16, 22, 23], based on the probability distribution of the original data. Quantizers that minimise the mean-square-error for various probability distributions (uniform, Gaussian, Laplacian, etc.) are well documented ([22, 24, 25] for example).

In reality, uniform quantizers perform adequately and are somewhat simpler to implement, so are used in existing coding standards. JPEG uses a uniform mid-tread quantizer, and MPEG-II uses two versions of the mid-tread uniform quantizer, with different values of \(d_1\), but with the same spacing between levels. There is no reason for the quantization to be the same for each of the frequency coefficients, and for intra coded data MPEG uses the matrix given in figure 6 by default (for non-intra coded data, the default value of 16 is used for every coefficient).
<table>
<thead>
<tr>
<th>8</th>
<th>16</th>
<th>19</th>
<th>22</th>
<th>26</th>
<th>27</th>
<th>29</th>
<th>34</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>16</td>
<td>22</td>
<td>24</td>
<td>27</td>
<td>29</td>
<td>34</td>
<td>37</td>
</tr>
<tr>
<td>19</td>
<td>22</td>
<td>26</td>
<td>27</td>
<td>29</td>
<td>34</td>
<td>34</td>
<td>38</td>
</tr>
<tr>
<td>22</td>
<td>22</td>
<td>26</td>
<td>27</td>
<td>29</td>
<td>34</td>
<td>37</td>
<td>40</td>
</tr>
<tr>
<td>22</td>
<td>26</td>
<td>27</td>
<td>29</td>
<td>32</td>
<td>35</td>
<td>40</td>
<td>48</td>
</tr>
<tr>
<td>26</td>
<td>27</td>
<td>29</td>
<td>32</td>
<td>35</td>
<td>40</td>
<td>48</td>
<td>58</td>
</tr>
<tr>
<td>26</td>
<td>27</td>
<td>29</td>
<td>34</td>
<td>38</td>
<td>46</td>
<td>56</td>
<td>69</td>
</tr>
<tr>
<td>27</td>
<td>29</td>
<td>35</td>
<td>38</td>
<td>46</td>
<td>56</td>
<td>69</td>
<td>83</td>
</tr>
</tbody>
</table>

**Figure 6:** Default MPEG-II Quantization matrix (W) (Intra coded blocks)

\[ QF_{u,v} = \frac{\text{round}(32 \times F_{u,v} / W_{u,v}) + \text{round}(0.75 \times q\_scale)}{2 \times q\_scale} \]

for intra coded data

\[ QF_{u,v} = \frac{\text{round}(32 \times F_{u,v} / W_{u,v})}{2 \times q\_scale} \]

for non-intra coded data

Figure 7 shows part of these quantizers for typical values of \( W(=16) \) and \( q\_scale(=4) \) (the full range of possible values of \( F \) is -2048 to 2047). Also shown are the results of applying the inverse quantizer to the data: each integer value of \( QF \) mapping to a single integer value of \( F \).

**Figure 7:** Quantizers used in MPEG-II: intra coded data, non-intra coded data.

**Coefficient scanning**

Once the 8x8 matrix of frequency coefficients has been produced and quantized, the coefficients need to be arranged into a suitable 1-dimensional form for entropy coding (lossless compression) and subsequently transmission. There are essentially two scanning
patterns (figures 8 and 9, from the MPEG-II coding standard) that are used, depending on whether the frames are interlaced or not.

![Default Coefficient Scanning Pattern](image)

*Figure 8: Default Coefficient Scanning Pattern (non-interlaced frames)*

![Alternate Coefficient Scanning Pattern](image)

*Figure 9: Alternate Coefficient Scanning Pattern (interlaced frames) and scanning pattern within one interlaced field*

The purpose of zigzag scanning the coefficients is that the DCT tends to compact the energy of the image into the lowest frequency coefficients \( f = \sqrt{f_x^2 + f_y^2} \), and (after quantization) scanning in this pattern leads to long runs of consecutive zeros. This in turn is ideal for run-length coding, leading to further compression of the information.

**Video Coding**

The compression of video uses all the techniques outlined for the compression of still images, but also makes use of the similarity and predictability of sequential frames to further reduce the amount of information required for transmission.
**DPCM loop**

In the basic DPCM (Differential Pulse Coded Modulation) loop of figure 10, the input frame $s$ and the estimate of this frame from the previous frames, $\hat{s}$, are used to calculate the error signal, $e = s - \hat{s}$. This exploits the inherent spatial redundancy of video by not requiring the coding of the image when there is no change from one frame to the next. The error signal is then frequency transformed and the resulting coefficients quantized, before being entropy coded to produce the compressed bitstream. Since the decoder only knows $\hat{e}$, and since $\hat{e}$ and $e$ are not identical due to the quantization of the data, the encoder contains a local version of the decoder to stop errors accumulating.

![Figure 10: Basic DPCM loop for video compression](image)

**Intra- / Inter-frame coding**

Although not strictly necessary, it is advantageous to occasionally encode a frame without reference to the estimate of the previous frame. Despite this requiring a significant increase in the amount of data required to code the image accurately, the quality of the reconstructed video is improved, as errors that are accumulating in the receiver are discarded. These "intra-frames" will have very different properties to the frames ("inter-frames") that are encoded with reference to (predictions of) other frames. It is likely, therefore, that

---

8. It is also essential to encode the very first frame of a sequence without reference to any other frames.
compression of these two image types will differ by more than the existence of a prediction of the frame.

**Motion Estimation**

In order to minimise the error signal within the DPCM coding loop, an estimate of the frame being encoded is formed from the previously transmitted frame(s). In order to achieve this, the movement in an image needs to be estimated and made available to the decoder. It is important to note that the motion estimation is not necessarily trying to accurately model the motion in the scene, rather it attempts to reduce the difference between the current frame and its estimate. This in turn leads to less energy in the frequency transform, and hence fewer coefficients to encode. The existing algorithms for estimating the motion fall into two categories, block matching and pel-recursive algorithms.

![Block Matching Motion Estimation](image)

*Figure 11: Block matching motion estimation*

The block matching algorithm compares each block (usually 16x16 pixels) in the current frame with locations in the previous frame, usually restricted to the vicinity of the current image block (figure 11). As it does so, it attempts to find the minimum value of a distortion measurement, for example:

\[ \text{Mean Absolute Difference, } MAD = \sum_{\text{block}} |s_i - \hat{s}_i| \]
Mean Square Difference, \[ MSD = \sum_{\text{block}} (s_i - \hat{s}_i)^2 \]

Each of these measurements of the difference requires the summation to be taken over the entire block for every location that is considered. It would be prohibitively time consuming to consider every possible location within the image for movement to have occurred from, and so motion estimation is usually constrained to be within the vicinity of the original block. This does not cause problems, as the motion between successive frames is usually relatively small, and the difference between the original frame and the estimate is transmitted to the decoder.

![Figure 12: Frame differences from block matching motion estimation algorithms (MAD, MSD) for Mobile & Calendar, prediction of frame 2 from 1, 16x16 pixel blocks, ±15 pixel search window](image)

Figure 12 shows the error signal after motion estimation using MAD (PSNR of frame and estimate: 23.02dB) and MSD (PSNR 23.13dB) algorithms used to predict frame 2 of Mobile and Calendar from frame 1. In both cases, the large scale features of the image are predicted well, the errors are concentrated in areas uncovered by motion and in areas of fine detail.

**Pel-recursive Motion Estimation**

Pel-recursive motion estimation is an iterative process that attempts to estimate a pixel in the current frame from information that is currently known to the decoder. The decoder can then use the same algorithm, without the need for transmission of motion vectors, to reconstruct the image. Equation 4 shows the calculation of the \((i+1)th\) iteration of the motion vector, \(\vec{D}_i\). The update term, \(\vec{U}_i\), is evaluated so as to attempt to minimise a measure of the difference between the actual frame and its estimate.
\[ \tilde{D}_{t+1} = \tilde{D}_t + \tilde{U}_t \] \hspace{1cm} \text{Equation 4}

The simplest method of estimating the minimum difference between the current frame, \( s_k \), and its estimate, \( \hat{s}_k \), from the previous frame \( s_{k-1} \), is based on finding the maximum value of the cross-correlation coefficient of a region (M) of the current frame with information in the previous frame. This is achieved by using a fraction of the gradient to the correlation coefficient surface as the update term (equation 5).

\[ \tilde{U}_i = \frac{1}{\varepsilon} \cdot \nabla_p \left( \sum_M s_k(r) \cdot s_{k-1}(r - D_i) \right) \] \hspace{1cm} \text{Equation 5}

Obviously \( \tilde{D}_i \) will not always have integer components. The intensity of the pixel in the frame \( s_{k-1} \) is obtained by linear interpolation of the four neighbouring pixels: figure 13, equation 6.

\[
I(x + \Delta x, y + \Delta y) = (1 - \Delta x)(1 - \Delta y) \cdot I(x, y) \\
+ (1 - \Delta x) \cdot \Delta y \cdot I(x, y + 1) \\
+ \Delta x \cdot (1 - \Delta y) \cdot I(x + 1, y) \\
+ \Delta x \cdot \Delta y \cdot I(x + 1, y + 1) 
\] \hspace{1cm} \text{Equation 6}

Figure 14 shows a contour plot of the correlation coefficient, and the iteration of the motion estimation for a single point, showing the algorithm converging to a local maximum in the cross-correlation coefficient. This is a particularly naïve algorithm which converges very slowly (for \( \varepsilon \) large), or tends not to converge (if \( \varepsilon \) is too small). Other algorithms have been proposed in [12, 26, 27] for example, that converge in many fewer iterations, whilst retaining stability.
Figure 14: Pel-recursive motion calculation using 5x2 block correlation coefficient:
Mobile and Calendar, frame 1 to 2, location {400,100}

All these algorithms suffer from two major disadvantages - they will all converge on
local maxima in the cross-correlation coefficient, not necessarily the optimum value; and the
cross correlation coefficient is not necessarily a good measure of the difference of a single
pixel (as it tends to lead to low-pass filtering of the image).

Figure 15: Pel-recursive motion calculation using 4x4 block MAD:
Mobile and Calendar, frame 1 to 2, location {400,100}

Figure 15 shows the same algorithm using MAD as the distortion measurement.
Convergence in this case is to a point in the opposite direction, but this gives a better
reflection of the true motion in the scene (in this sequence, the camera pans to the left and so
motion of the image is to the right). The MAD field is less smooth than the cross-correlation
field, and so there is a greater chance of converging to a local turning point (in this case, a
local minimum at about \(\{0.9,-0.1\}\) has been chosen, ignoring a deeper minimum at about \(\{0,2.5\}\).

There is no fundamental reason why these two techniques cannot be combined into a hybrid scheme. The major principle of the pel-recursive technique is prediction of a pixel from the pixels transmitted so far in this image, and the previous image, using an equation of the form:

\[
f(s_k(x), s_{k-1}(x) - D(x))
\]

evaluated over a region \(M\) of the current image above and to the left of the current pixel. The block matching technique finds a value of \(D\) for a block of image, by finding the minimum value of a distortion measurement. If this search technique is applied to a small image block (which the current pixel is the bottom-right corner of), motion vectors need not be transmitted as the receiver can find the same estimate from information it already possesses. Figure 16 shows the error signal from predicting frame 2 of Mobile and Calendar from frame 1, using MAD as the distortion measurement and 16x16 pixel blocks. The energy in this error frame is almost identical to that obtained by conventional block matching motion estimation, figure 12.

![Figure 16: Hybrid motion estimation using MAD of 16x16 pixel blocks, motion restricted to ±15 pixels. Difference between frame 2 and its estimate.](image)

Motion estimation for each pixel of the image inherently increases the amount of calculation required to form a prediction of a frame. The amount of calculation can be reduced by considering smaller blocks of image (4x4 pixels instead of 16x16), but this leads to an increase in the energy of the error signal. The algorithm remains computationally
intensive and so is unlikely to be incorporated into video coding standards until computation speeds increase or the algorithm can be simplified.

Existing video coding standards are based on block-matching motion estimation, as this provides a good estimation of the motion field, with low energy in the error signal. Further developments of the pel-recursive algorithm may mean that it is capable of rivalling the performance of block-matching, with the significant advantage of not needing to transmit motion vectors.

DCT Filtering

Figure 17 shows contours of constant frequency in coefficients that result from taking a DCT. Also shown is a typical "zonal mask" which only retains a selection of all the DCT coefficients. The use of a mask following a zigzag pattern approximates to the contours of constant frequency, and so this mask represents a low-pass filter being applied to the image ([28, 29, 30]). The ability to low-pass filter an image has been used to develop hierarchical video encoders offering different levels of service quality ([31, 32, 33] for example), where the usual approach is to use a fixed mask.

![Figure 17: Zonal mask applied to DCT coefficients (non-interlaced frames)](image)

Later in this thesis, adaptive modification to the zonal mask will be used as the fundamental basis for the separation of lesser importance, high frequency information from the remainder of a compressed video bitstream. This will allow the video bitstream to be transmitted through channels of differing error resilience.
**Buffering**

In nearly all applications the output bit-rate from a video encoder needs to be constant over some pre-determined amount of video. This is to ensure that video can be transmitted without the transmitter having to stall when the video data is not available (as in any real system, frames of video are only available at certain times\(^9\)). The output bit-rate can be controlled by feeding back into the compression procedure to vary the amount of compression achieved. This is done by varying the number of quantization steps, so as to produce more or less data at the output of the encoder, although the actual effect of this modification to the quantization cannot easily be predicted as the output of the quantizer is entropy coded.

**Video Coding Standards**

Having established the principles behind still image compression and video compression, some of the international standards for video compression can be considered. Of particular interest in this thesis is the MPEG-II coding standard which is outlined in some detail after brief descriptions of the main standards.

**H.261**

H.261 [34,35] was developed to encode video (CIF\(^{10}\) [36, 37] or QCIF\(^{11}\)) at multiples of 64 kbits/s, with applications such as video-phone and video-conferencing as the primary goal. Compression is achieved in a similar manner to JPEG (outlined earlier) but with the addition of motion-compensated temporal prediction to take account of the inherent temporal redundancy of video.

---

\(^9\) This is an important aspect that always needs to be considered carefully when software simulation is used as the only source of analysis. As the software encoders are rarely real-time implementations, the video will exist on disk beforehand, and so rate control problems might not be evident.

\(^{10}\) Common Intermediate Format, CIF, at a resolution of 352x288 pixels was developed to allow a single video coding standard to operate on different video formats (such as the 625-line PAL and SECAM formats used in Europe, and the 525-line NTSC format used in North America)

\(^{11}\) Quarter CIF : 176x144 pixel resolution.
MPEG-I,II

The MPEG (Motion Picture Experts Group) coding standards are extensions to H.261, intended for high quality video applications [38, 39, 40] with added flexibility (at the expense of encoder complexity). MPEG-I [41] is intended for applications up to about 2Mbits/s, whereas MPEG-II [42] is intended for higher bit-rate (and hence higher-quality) applications. MPEG-II also introduces capability for higher resolution images, different subsampling of the chrominance fields, and modifications for compressing interlaced video.

H.263

The H.263 [43,44] standard is a variation on the MPEG standards, primarily intended for encoding low-resolution video at very low bit-rates (typically 176x144 pixels for luminance, 88x72 pixels for chrominance, encoded at up to 34kbits/s making it suitable for real-time modem transmission via PSTN networks). Compression is achieved using the same techniques as MPEG and the structure of the bitstream is also similar, however the statistics of the quantized data are very different and so the codebooks are replaced. In fact, the quantization of the DCT coefficients leads to very few coefficients, and efficiency is increased through the use of three-dimensional variable length coding (length of run of zeros, subsequent level, and whether this code is the last coefficient).

Comparison of Features

<table>
<thead>
<tr>
<th></th>
<th>JPEG</th>
<th>H.261</th>
<th>MPEG-I</th>
<th>MPEG-II</th>
<th>H.263</th>
</tr>
</thead>
<tbody>
<tr>
<td>DCT</td>
<td>8x8</td>
<td>8x8</td>
<td>8x8</td>
<td>8x8</td>
<td>8x8</td>
</tr>
<tr>
<td>Zigzag scan</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Alt. Scan (interlaced)</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Run-length coding</td>
<td>2D</td>
<td>2D</td>
<td>2D</td>
<td>2D</td>
<td>3D</td>
</tr>
<tr>
<td>Arithmetic coding</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Motion estimation</td>
<td></td>
<td>Forwat</td>
<td>Bi-directional</td>
<td>Bi-directional</td>
<td>Bi-directional</td>
</tr>
<tr>
<td>Motion accuracy</td>
<td>pixel</td>
<td>half-pixel</td>
<td>half-pixel</td>
<td>half-pixel</td>
<td></td>
</tr>
<tr>
<td>Bit-rate control</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Target bit-rate</td>
<td>0.64 Kbit/s</td>
<td>&lt; 2Mbps</td>
<td>&lt; 100Mbps</td>
<td>34 Kbit/s</td>
<td></td>
</tr>
</tbody>
</table>

Table 6: Comparison of JPEG, H.261, MPEG-I, MPEG-II and H.263 encoders

MPEG-II is being adopted as the coding standard for broadcast quality digital video, and so is considered throughout the rest of this thesis. As there is a good deal of similarity in the algorithms for H.261, MPEG-I, MPEG-II and H.263 (see table 6), it is likely that work
based on any particular one of these will be extensible to the others. Where this is not the case, this is highlighted.

The MPEG-II Video Coding Standard

MPEG-II is designed for high quality video compression, providing a constant compression ratio of up to about 50x. This is attained (at the expense of reduced image quality) through a combination of:-

Inter-/intra- frame encoding
Block matching motion estimation
DCT
Quantization
Zigzag coefficient scanning
(Huffman) Variable length encoding
Output buffering, feeding back into the quantizer

The full detail of the MPEG-II coding standard is available as [42], and it is unnecessary to reproduce the entire specification here. Nevertheless, it is important to outline the basic principles of the coding scheme in order to be able to understand the motivation behind this research.

The output of the MPEG-II encoder is a bitstream which comprises of six distinct levels (figure 18), each representing one layer of the encoder. The “block layer” has the least spatial effect on the video sequence: a mere 8x8 pixels. At the other extreme, the “sequence layer” determines fundamental properties of the video that affect the entire decoding process.
The sequence layer defines parameters that are global to the video bitstream, for example the picture size and frame rate (the actual fields present are outlined in Table 18 in Appendix A, page 173). In many applications this data is considered *a priori* knowledge as this information (such as image resolution) will not vary from programme to programme. However, assuming this is restrictive. A major advantage of transmitting this information is that a single television set could be developed to display programmes in a variety of formats, for example PAL and NTSC. This would finally enable world-wide television broadcasting.
Group of Pictures Layer

The Group of Pictures layer is primarily intended for applications where random access into the video sequence is required, for example fast-forward / -reverse playback. In every Group of Pictures, the first frame is encoded without reference to any other frames, and so can always be decoded - thus enabling a decoder to skip all other frames within the GoP. The GoP is the lowest level that a decoder can completely re-synchronise at (for example, when a change of channel is requested by the viewer), and so should occur quite frequently within the bitstream. This highlights a particular problem with the use of compression algorithms based on frame prediction: a decoder cannot complete a request to change channel immediately. On average, the delay will be half the time between successive GoPs, typically about 0.2s for a GoP spacing of ten frames. Although encoding frames without reference to any other frames is less efficient, in a multi-channel system it is vital to transmit this information frequently: realistically no less often than every ten frames.

Picture Layer

The Picture layer (and all layers below it) encompasses all the information (see Tables 20 and 21 in Appendix A, page 174) required for a single frame of any of the three picture types defined in MPEG-II:

I: Intra

Frame is encoded entirely within itself. No predictions exist from other frames.
P : Prediction  Frame is a prediction based on the previous I- or P-frame.

B : Bi-directional prediction  Frame is a prediction formed from the previous I- or P- and the next P-frames.

In order for the prediction of B-frames to occur from the previous and the next P-frames of the sequence, the frame order at the input to the encoder is altered : figure 21. The decoder has already decoded the two reference frames before it attempts to recreate the B-frames, which can then be placed back in the correct order at the decoder output. Re-ordering the frames inherently introduces a delay in the system of two frames at the encoder and a further frame at the decoder.

Original frame order

\[
\text{I B B P B} \quad \text{B P B P}
\]

Re-ordered frames showing predictions

\[
\text{I P B B P B} \quad \text{B P B B}
\]

\text{Figure 21 : Re-ordering of frames to enable bi-directional prediction}

The picture header also provides information about the state of the buffer in the encoder, telling the decoder how long to wait before decoding of the current frame can begin. This is to ensure that the bitstream input buffer in the decoder never overflows (causing the decoder to be unable to cope with the bits arriving) or underflows (where the decoding process has to stall, and in doing so loses time synchronisation on the individual frames).

Slice Layer

A slice is an arbitrary number of 16x16 pixel blocks, contained within a single 16 pixel deep row of a picture. Slices need not occupy a full row of the picture (though in many implementations they do). Also permitted in the standard (but rarely used) is the possibility of multiple slices existing on a single row of the picture, though they cannot overlap.
The slice is the lowest layer in the MPEG-II bitstream that contains synchronisation information in case the bitstream contains errors. The pattern of bits 0000,0000,0000,0000,0000,0001 is unique within the bitstream and is used as a marker indicating the start of a “high-level” object within the bitstream. In the case of a slice, this is combined with eight bits encoding the vertical position of the slice to give a unique 32-bit “start code”. Great care is taken throughout the entire MPEG-II coding standard to ensure that start code emulation (the unintentional creation of the same bit pattern) is not possible, so that a decoder can use this code for synchronisation purposes.

**Macroblock Layer**

![Macroblock Layer Diagram]

*Figure 22: Macroblock layer, showing block structure for 4:2:0 format video*

The macroblock layer is the layer at which motion estimation is introduced into the video encoding. Each 16x16 pixel macroblock may contain motion vectors (forward and/or backward depending on how the picture is encoded) and the remaining picture information that needs to be transmitted in the form of “blocks”. The actual number of blocks contained within the macroblock depends on how the colour information of the video sequence is encoded, but is typically the four luminance blocks and two chrominance blocks (one each for the U- and V-components).

**Block Layer**

The block layer is the fundamental layer that provides video information (in the form of DCT coefficients). The quantized coefficients are encoded using a 2-dimensional run-length code (number of zeros and subsequent level encoded as a single variable length code) according to the zigzag scan defined in figure 8 (page 17). Blocks that contain no data are not transmitted, and this is controlled by a field within the macroblock. As a result, it is not
possible for the first code of a block to be the “End Of Block” marker so the codebook is altered for this first coefficient.

Summary

This chapter has presented the basic principles behind still image and video compression, leading to a description of existing international standards that have been defined. Particular emphasis has been placed on the MPEG-II video coding standard, as this is emerging as the preferred video coding scheme for medium to high bit-rate applications. This popularity as a coding standard is leading to the development of dedicated MPEG-II encoder and decoder integrated circuits, which will be essential if and when MPEG-II becomes the world-wide compression scheme for digital broadcast television.

Having outlined the techniques used in the encoding of digital video, it is important to consider how the video will be delivered to the viewer. Chapter Two provides a review of existing techniques that have been developed for protecting the compressed data, before the rest of this thesis considers this problem in greater depth.
Chapter Two: 
Transmission of Compressed Video
Chapter Two: Transmission of Compressed Video

Chapter One outlines the basic techniques that exist for the compression of video images. However, it is not sufficient to consider the compression of the video in isolation: it is nearly always necessary to consider how the compressed video will be delivered to the viewer, and in turn how this affects the compression process. In the absence of transmission errors, the highest quality decoded video will be the video that has been damaged the least by lossy compression, which for a given compression algorithm is the video with the highest output bit-rate. To maximise the transmission rate, modulation schemes that encode several data bits into each symbol are often used - but this is not without cost. These transmission schemes are much more susceptible to error (as will be shown formally in Chapter Three), and how the video bitstream responds to these errors is of fundamental importance.

This chapter presents a brief review of the main approaches to the transmission of compressed video bitstreams, and the properties associated with each.

Multi-bit per symbol with equalising error correcting codes

In order to lessen the effect of transmission errors when using a multi-bit per symbol modulation scheme to increase the available channel bandwidth, error correcting codes can be added to the data prior to transmission (figure 23). Careful choice of these codes removes the imbalance in error rates associated with the different data bits that comprise a single symbol [45], leading to an approximately constant bit-error rate for all transmitted data bits. This makes the division of the video bitstream purely a process of splitting into a ratio so that the data rates after error correction are identical. Compatibility is retained with the pre-defined video coding standards, as modifications are reserved for the transmitted data and the transmission rate is fixed (so there are no rate-control issues).
The fundamental problem with such a scheme is that the error correcting codes need to be chosen in advance, and as such cannot adapt to the quality of the transmission channel. This inherently makes the scheme inefficient when the channel quality is good, and will still break down when the channel quality is very poor (the performance of error correcting codes falls off rapidly as the channel quality worsens, leading to sudden breakdown in the received video quality).

Although not well suited to the variable fading pattern associated with the mobile radio environment, its performance in stationary broadcast applications is good and this is the technique chosen for the wireless broadcast of High Definition Television in the United States [46].

Retransmission of damaged data

The simplest (conceptually) solution to the data being received in error, is for the receiver to ask the transmitter to re-send the damaged data. This is the standard approach used in many protocols where real-time transmission of data is not critical, and has been shown to be more effective than the use of error correcting codes in fading channels [47]. The majority of video applications are time-critical however, and so the request for retransmission must also be passed to the video encoder in order to control the transmission bitrate. Since the receiver has to communicate back to the transmitter to request re-sending of data, each transmission link requires some form of video encoder at the transmitter.
To retain the real-time property, the receipt of the message indicating that the received data is corrupt can be used to modify future data to minimise the duration of the error at the receiver. In compressed video applications, the majority of the frames are encoded as predictions from previous frames, and so when the receiver signals to the transmitter that an error has occurred the transmitter should replace the next predicted frame with an intra-frame (figure 24: which shows intra-frames created at half the picture rate). The result is that the damaged video is repaired immediately upon receipt of the replacement intra-frame, rather than waiting for the next (scheduled) intra-frame. For MPEG bitstreams this is less significant as the intra-frame period is typically about twelve frames. H.263 however does not require the regular sending of intra-frames after the initial frame, and so benefits greatly from being able to request resending of the damaged information [48, 49].

![Video encoding showing revised frame order after errors are reported](image)

The added intra-frames have to be constructed carefully, as they will be used by the decoder as the starting point for prediction of subsequently transmitted inter-frames. It is important therefore that the intra-frames inserted are as close a match as possible to the frame that the receiver would have produced by decoding the original (undamaged) predicted frames. This requires the intra-frames to be formed from the output of a local decoder operating on the transmitted bitstream\(^\text{13}\) such that \(I_4=I_1+P_2+P_3+P_4+P_5+P_6\).

The intra-frame that is inserted into the bitstream requires a larger number of bits than the inter-frame it replaces, which causes a problem in the buffering at the transmitter. This

\(^{13}\) This is slightly different to [49], where the intra-frames are created as part of the encoding process, but this seems to neglect the fact that, for the second damaged frame onwards, the frame sequence has been altered. That is, with the error shown, \(I_5=I_4+P_8\) rather than \(I_5=I_1+P_2+P_3+P_4+P_5+P_6+P_7+P_8\).
necessitates the provision of feedback into the original video encoder to control the output bit-rate (with loss of compatibility with existing implementations of the encoding standards), or for the subsequent inter-frames to be re-encoded at a lower bit-rate. In either case, it is not practical to fully separate the video encoder from the transmitter. Since the connection required is bi-directional, this requires a separate video encoder for each receiver.

**Variable-rate transmission**

A more bandwidth efficient alternative to the two schemes outlined above is to modify the modulation scheme to take account of the quality of the communication link (via feedback from the receiver) [50, 51]. When the noise level is relatively high, the number of bits encoded per transmitted symbol can be reduced, reducing the bit-error rate in the received bits but at a cost of reduced data being transmitted. The variation in the transmission rate removes compatibility with existing implementations of the video encoding standards, but the actual compression algorithms can remain relatively unchanged. [52, 53] present results of an H.263 compatible video encoder switching from BPSK (in poor channels) through QPSK, 16-star-QAM to 64-star-QAM in high quality channels (figures 25 and 26).

![Variable rate QAM modulation scheme: from 1 bit/symbol to 6 bits/symbol](image)

**Figure 25**: Variable rate QAM modulation scheme: from 1 bit/symbol to 6 bits/symbol
This technique requires that the communication be point-to-point, as feedback on the channel quality is required. Further, the delay in informing the transmitter about the (perceived) quality of the channel must be short compared to the period of the fluctuations in channel quality. This places a *minimum* on the transmission symbol rate which is dependent on the carrier frequency and the maximum speed of the receiver.

**Prioritised video bitstreams via sub-band coding**

The principle of sub-band coding is to divide the original signal's frequency spectrum into several bands (the 'sub-bands') via the application of matched high- and low-pass filters (for example, the Quadrature Mirror Filter, QMF [54, 55]). Primarily introduced for speech coding applications [56, 57, 58], sub-band coding has several properties that are also useful in video coding applications:

- the noise generated within a single sub-band does not spread to the other bands.

- the image is not partitioned into small blocks of pixels (typically 8x8 for DCT) and so sub-band coding does not lead to the blocking artefacts typical of DCT based encoders [59].
• the sub-bands have different visual importance, enabling the encoder to output compressed video bitstreams with different significance levels.

• perfect reconstruction (provided the encoded data is not damaged) of the original data [60] is possible.

The one-dimensional sub-band filter used for speech coding can be extended to a two-dimensional sub-band filter for use in image coding applications by applying the filter in both horizontal and vertical directions [61] (figure 27). This leads to four separate frequency bands (of half the horizontal and vertical resolution of the original image), an example of which is shown in figure 28.

![Figure 27: Two-dimensional (separable) sub-band filtering](image)

The resulting images (figure 28) are then encoded using the techniques outlined in Chapter One, to produce complete video encoders ([62, 63] for example). Only the lowest frequency band retains the high degree of spatial correlation typical of video images, and so only this band benefits from motion estimation and transformation to the frequency domain. The resulting compressed bitstreams are completely separate (though they may be multiplexed back together for transmission purposes) and require their own bit-rate control in the encoder. This results in an inability to re-distribute data between the layers and may lead to coding inefficiencies when some of the frequency bands are 'quiet'.
Obviously these coding schemes are incompatible with existing video coding standards, though some work has been done to try to retain the advantages of sub-band coding within a framework that is compatible with existing coding standards [64].

The resulting encoder (figure 29) is, however, unrealistically complex. Only the base layer is compatible with the existing coding standard, and the complexity (including a complete
decoder coupled with an entirely new video encoder for the enhancement layer) of the enhancement layer encoder does not justify this compatibility, unless compatibility with legacy encoders is a fundamental requirement.

**Hierarchical encoding via fixed zonal mask**

The frequency-space transforms outlined in Chapter One provide a method of dividing the information into different levels of visual importance, by varying the number of frequency coefficients retained. In a similar manner to the sub-band approach outlined above, the low-frequency coefficients are transmitted in the low bit-error rate data bits. With suitable error resilience in the encoding of this data, some information for the picture will always be decoded. The remaining coefficients are transmitted through the more noisy data bits, but errors only affect the 'refinement' layer.

![Diagram](#)

*Figure 30: Hierarchical image encoding for hierarchical modulation*

[65] presents a still image coding algorithm based on this technique (figure 30). The number of coefficients transmitted through each layer is fixed (defined by the zonal mask highlighted), and the coefficients are not entropy coded. Although this makes decoder synchronisation easy, and hence reduces error propagation, the compression ratio achieved is quite low (26% compression). However, the results show that this is likely to be a viable technique for transmission through the severely fading mobile multipath environment, provided that the efficiency can be improved upon. Recovery from the bursts of errors typical of the Rayleigh fading channel is particularly good.
There is no reason why this scheme could not be adapted to work with video\textsuperscript{14} rather than individual frames, although this would require development of an entirely new encoder. Further, inefficiency problems are likely to be severe in low-activity regions of an image (where, for example, frame prediction has been quite successful resulting in little information needing to be transmitted).

\textbf{Summary}

Of the methods outlined in this chapter, some require feedback and so are only suited to point-to-point transmission (i.e. non-broadcast applications), and some are only efficient in the channel for which they were developed. More fundamentally important however, is the fact that only one is directly compatible with the video coding standards that have now been accepted as international standards, and are now beginning to appear as VLSI devices. The complexity of these VLSI devices is such that re-engineering the entire encoder is not a practical solution.

Why the internationally accepted standards do not fully consider transmission aspects within the coding model itself is an interesting question, but is not of real relevance to this thesis. However, the question of whether or not the existing standards could be adapted to accommodate problems introduced by transmission is important. The remainder of this thesis is concerned with answering this question: firstly by developing a simulation of a particularly severe transmission link (through a multipath fading channel), and then by introducing a black-box transcoder that sits between the existing video encoder and the transmitter. This transcoder (and the corresponding inverse transcoder in the receiver) provides the

\textsuperscript{14} If the receiver is situated in a location of a severe fade, and remains stationary, then it is possible that no meaningful data will be received, and consequently the recovered images will be meaningless. Use of this transmission environment for video requires that recovery from such situations is possible (I-frames transmitted periodically).
modification to the existing video coding standard to target the particular problems associated with transmission, whilst retaining the already existing video coding model.
Chapter Three: Transmission
Chapter Three: Transmission

In order to be able to consider an optimal approach to coding digital signals, it is necessary to consider the environment through which the signals will be transmitted, and how that environment will affect the received data. In the case of digital video, five primary ways in which the information can be made available to the viewer can be considered:

i) Wireless broadcast to a mobile receiver.

ii) Wireless broadcast to a stationary receiver.

iii) Satellite

iv) Cable

v) Digital recording media

Wireless broadcast to a mobile receiver is probably the most severe communication channel, as there is potentially severe fading due to there being multiple reflected paths (time variant) and no direct line of sight signal. Stationary wireless broadcast is difficult to model as there are similarities with the multipath effect of broadcast to a mobile receiver, but the fading varies slowly if at all. This makes the choice of parameters for the simulation model difficult. Satellite, cable and digital recordings (such as CD-ROM, etc.) have low error rates, which (with the possible addition of error correcting codes) can be considered error free.

This chapter considers the most severely noise limited channel of wireless broadcast to mobile receivers, given that the other communication channels outlined above will all perform better. The overall aim is to develop a simulation model, and verify its functionality by comparison of the overall error rates with the theoretically obtained equations.

Channels

In all but the most ideal scenarios, the received signal will consist of reflections of the original signal from local geographical features whose surfaces may have very different
reflective properties. As these reflected signals have travelled via different paths to reach the receiver, they will have been delayed by a time $\Delta t$

$$\Delta t = \frac{\text{path length difference}}{c}$$

Assuming that $\Delta t$ is small compared to the time between adjacent symbols, then the effect of inter-symbol interference (ISI) will be negligible, and so the time delay can be thought of purely as a phase shift of $2\pi f \cdot \Delta t$. Typical mean values of $\Delta t$ (for -30dB) are 1.3$\mu$s (urban) and 0.5$\mu$s (suburban) [66 p42]. This corresponds to a symbol rate of about 2MHz (suburban), above which ISI will become a problem. In developing a simulator, it is safe to assume that the input to the receiver is a collection of reflections of the original signal with random amplitudes and phases, and that ISI can be ignored below a signalling rate of about 2MHz.

**Mobile Radio**

In the mobile radio scenario, either the transmitter or the receiver (or both) is not stationary. The effect of this is twofold:

i) it is not possible to guarantee a direct line of sight to the transmitter. Indeed, since the moving aerial has to be near the ground, it is quite likely that the direct line-of-sight will be obscured most of the time.

ii) the receiver moves through the field of reflected signals at speed $v$. The phases of all the reflected terms (except those that are directly perpendicular to the direction of motion) will vary as the receiver moves. In time $t$, the phase of each reflected signal will change by $2\pi \frac{v t}{\lambda} \cos \phi_i$

The signal at the receiver can thus be expressed as:

$$s_0(t) = a_0 e^{j\omega t} \cdot \sum_{i=1}^{N} a_i e^{2\pi f t \cos \phi_i} \text{                      .................................................. Equation 7}$$

where $\phi_i$ is the angle between the incoming wave and the direction of motion.
\( \hat{a}_i \) is a complex variable denoting the random amplitude and phase of the reflected signals.

\( t \) is the time, which in the simulation is replaced by the symbol number multiplied by the symbol duration, \( T_s \).

Figure 31 shows a typical multipath scenario where the direct line-of-sight signal is obscured. It is possible to accurately model the amplitude and phases of the reflected signals using ray-tracing techniques, but this is only really practical on small cell sizes (for example, pico-cells within a single building). In reality, it is not possible to know a priori the precise location of all local features that could be contributing to the reflected signals. Therefore, in the usual model ([66 p172, 67, 68] for example) for describing the mobile radio channel, \( \hat{a}_i = R_i + jS_i \) (where \( R_i \) and \( S_i \) are independent Gaussian random variables of mean 0 and variance 1). It follows that:

\[
E[R_i] = 0, \quad E[S_i] = 0, \quad E[R_i^2] = 1, \quad E[S_i^2] = 1
\]

Rewriting equation 7 as:

\[
s_0(t) = a_0 e^{j\alpha t} \cdot (X_i + jY_i) \quad \text{with} \quad X_i = \sum_{i=1}^{N} (R_i \cos \xi_i - S_i \sin \xi_i)
\]
\[ Y_i = \sum_{i=1}^{N} (R_i \sin \xi_i + S_i \cos \xi_i) \]

\[ \xi_i = 2\pi \frac{\nu_t}{\lambda} \cos \phi_i \]

\( X_i \) and \( Y_i \) are, from the Central Limit Theorem ([69 p143] for example), also independent Gaussian random variables provided \( N \) is large \((\geq 10)\). Thus:-

\[ E[X_i] = 0, \quad E[Y_i] = 0, \]

\[ E[X_i^2] = \sum_{i=1}^{N} \left[ E\left[R_i^2 \cos^2 \xi_i\right] + E\left[S_i^2 \sin^2 \xi_i\right] \right] = \sum_{i=1}^{N} \frac{1}{2} + \frac{1}{2} = N, \text{ ....Equation 8} \]

\[ E[Y_i^2] = N \]

Given that \( X_i \) and \( Y_i \) are independent Gaussian random variables, the probability distribution of the fading term in equation 7 is given by :-

\[ p_{X_iY_i}(X_iY_i) = p(X_i) \cdot p(Y_i) = \frac{1}{2\pi \sigma^2} \cdot e^{-\frac{x^2 + y^2}{2\sigma^2}} \]

where \( \sigma^2 \) is the variance of \( X_i \) and \( Y_i \). From equation 8, therefore, \( \sigma^2 = N \).

Changing variables from \( X_i, Y_i \) to \( r, \theta \) gives:-

\[ p(r, \theta) = \frac{r}{2\pi N} \cdot e^{-r^2/2N} \]

The probability distribution for the amplitude of the fading term is therefore given by:-

\[ p(r) = \int_{0}^{2\pi} \frac{r}{2\pi N} \cdot e^{-r^2/2N} d\theta = \frac{r}{N} \cdot e^{-r^2/2N}, \text{ for } r \geq 0. \text{...............................................Equation 9} \]

---

**Figure 32:** Rayleigh Distribution for \( N=10 \)
\[ p(\theta) = \frac{1}{2\pi} \quad \text{for } 0 \leq \theta < 2\pi \]
\[ p(\theta) = 0 \quad \text{otherwise} \]

This is the classic Rayleigh distribution which is used as the theoretical basis for the simulation of mobile radio transmission. In conjunction with a detailed knowledge of the modulation scheme used for transmission, this will allow a theoretical derivation of the expected error rates for transmission through the mobile channel.

The simulation package is therefore based around the following description of the multipath fading environment:

\[ s_0(t) = a_0 e^{j\omega t} \sum_{i=1}^{N} (R_i + jS_i) \cdot e^{2\pi f_d T_s k \cos \theta} \]

Equation 10

where the assumption is made that the N incoming reflected waves are uniformly distributed.

Equation 10 contains all the variables relating to the motion of the mobile unit in a single term. It makes sense, therefore, to combine these parameters into a single quantity which uniquely defines the properties of the channel. Defining \( f_d \) as the maximum Doppler shift (\( f_d = v/\lambda \)), the channel properties are defined by the dimensionless quantity \( f_d T_s \) (which represents the maximum number of wavelengths travelled between successive symbols).

\[ s_0(t) = a_0 e^{j\omega t} \sum_{i=1}^{N} (R_i + jS_i) \cdot e^{2\pi f_d T_s k \cos \theta} \]

Equation 11

The rate of fading is dependent on \( f_d T_s \) as shown by equation 11. Typical values of \( f_d T_s \) are ([70] for example) in the range \( 10^{-2} \) (shown in figure 33) down to \( 10^{-5} \) (for pedestrians).
Modulation Schemes

It is not within the scope of this thesis to investigate all possible methods of mapping the data bits onto the carrier wave for transmission, as the possibilities are almost endless. Instead, this thesis concentrates on how some existing, well-known modulation schemes can be used efficiently for the transmission of compressed video data. In order to reduce the complexity of the simulation package, the simulations are restricted to phase and/or amplitude modulation in the baseband.

The use of phase and amplitude modulation ([71] for example) can, if constructed correctly, lead to some useful properties in terms of the bandwidth efficiency, and the error probabilities associated with each of the bits. As shall be shown, the bit-error rates can sometimes be constructed to be different for the different bits comprising a given symbol, and can also be tuned by varying properties within the modulator and demodulator. This requires an accurate theoretical description of the bit-error rates for a given modulation scheme. To begin with, the properties of a simple two level phase modulation scheme are derived, and this is then extended into four and then sixteen level schemes. Throughout all the following derivations, it is assumed that each of the possible symbols is equally likely. Obviously, this depends greatly on the form of the data being transmitted, but can be achieved using scramblers.
Coherent Binary Amplitude Modulation (PSK)

For a signal of amplitude ±A, in a channel subject to additive white Gaussian noise (i.e., a non-fading environment), the bit-error probability is given by:

\[ P_e = P\{A + n_c < 0\} = P\{n_c < -A\} \]

with \( n_c \) Gaussian with \( \mu = 0 \)

\[
P_e = \frac{1}{2} - \frac{1}{\sqrt{\pi}} \int_{-\infty}^{-A} e^{-\frac{x^2}{2\sigma^2}} dx
\]

\[
P_e = \frac{1}{2} \left( 1 - \frac{2}{\sqrt{\pi}} \int_{0}^{A} e^{-\frac{x^2}{2\sigma^2}} dx \right) = \frac{1}{2} \left( 1 - \text{erf} \left( \frac{A}{\sqrt{2}\sigma} \right) \right)
\]

\[
P_e = \frac{1}{2} \text{erfc} \left( \frac{A}{\sigma\sqrt{2}} \right) \quad \text{Equation 12}
\]

The average energy (per bit\(^{15}\)) associated with PSK is \( E_b = A^2 \). The noise spectral density, \( N_0 \), is defined by \( N_0 = 2 \cdot \sigma^2 \), and so equation 12 can be rewritten as:

\[
P_e = \frac{1}{2} \text{erfc} \left( \frac{E_b}{\sqrt{N_0}} \right)
\]

In a fading environment, the amplitude of the received signal is reduced, due to the fading, by a factor \( \alpha \). Thus equation 12 becomes:

\[ P_e = P\{\alpha A + n_c < 0\} = P\{n_c < -\alpha A\} \]

\(^{15}\) For PSK, there is no difference between the average energy and the average energy per bit, as only one bit is encoded per symbol. For other modulation schemes, a symbol will be formed from more than one bit and so the use of average energy per bit gives a normalised measure of the energy.
However, in multipath fading, $\alpha$ is not constant (it is the Rayleigh envelope, the probability distribution of which is given by equation 9 (page 48)). The expected value of the error probability (the bit-error rate) is given by:

$$P_e = \frac{1}{2} \text{erfc}\left(\frac{\alpha A}{\sigma \sqrt{2}}\right)$$

Defining and using the integral derived in Appendix B, (see figure 37 on page 56)

$$\langle P_e \rangle = \int_{0}^{\infty} p(\alpha) \cdot \frac{1}{2} \text{erfc}\left(\sqrt{\gamma}\right) \cdot d\alpha$$

$$= \int_{0}^{\infty} \frac{1}{2 \sigma^2 \gamma} e^{-\frac{\gamma}{\sigma^2}} \cdot \text{erfc}\left(\sqrt{\gamma}\right) \cdot \frac{\sigma^2}{A^2} \cdot d\gamma$$

where $\gamma = \frac{\alpha^2 A^2}{2 \sigma^2}$

Defining $\gamma_0 = \frac{A^2 \sigma^2}{\sigma^2} = 2 \sigma^2 \frac{E_b}{N_0} = 2 N_0 \cdot \frac{E_b}{N_0}$, and using the integral derived in Appendix B,

$$\langle P_e \rangle = \int_{0}^{\gamma_0} e^{-\gamma / \gamma_0} \cdot \frac{1}{2} \text{erfc}\left(\sqrt{\gamma}\right) \cdot d\gamma = \frac{1}{2} \left(1 - \frac{1}{\sqrt{1 + \frac{1}{\gamma_0}}} \right)$$

(see figure 37 on page 56)

The PSK result is equally valid for Quadrature-PSK (QPSK) shown below, except for the fact that the constellation points are now closer (by a factor $\sqrt{2}$) to the decision thresholds (see figure 37 on page 56). As there are now four constellation points, each symbol is described by two bits. The bandwidth efficiency is therefore doubled and the average energy per bit ($E_b$) is halved, so for a given $E_b / N_0$ the error rates of PSK and QPSK are identical.

![Figure 35: QPSK with Gray-coded bit-symbol assignments](image-url)
The I and Q components of the signal are statistically independent, and so both the bits have exactly the same bit-error rate (provided that the 2 bits are Gray-code mapped onto the symbol).

16 Level Quadrature Amplitude Modulation

The bandwidth efficiency can be increased still further by introducing more constellation points, but at the cost of increased error susceptibility. Figure 36 shows the three common configurations for 16 level QAM: Types I, II, and III [50, 72, 73, 74]. Each is labelled with corresponding bit patterns to give optimal noise performance (note there are several equally optimum assignments of bits to constellation points). It is important to note that Type-II and Type-III are almost identical - one point in each quadrant is moved so that in Type-II QAM there are only two possible amplitudes. The probability of error should therefore be quite similar in QAM Types II and III (for which it is much simpler to calculate the theoretical value of).

![Figure 36a: 16-QAM (Type I)](image)

![Figure 36b: 16-QAM (Type II)](image)
In Type-III QAM, the constellation points are equally spaced, leading to optimum performance in the presence of additive white gaussian noise [65]. However, to demodulate this requires knowledge of the amplitude of the received signal. In the derivation of the theoretical error rates that follows, this is assumed to be known - methods of achieving this (approximately) are described later in this chapter. As the simplest of the three 16-QAM constellations described above, Type-III QAM is used for the following calculation of the theoretical bit-error rates (BER) for the 4 bits that are encoded in each symbol. This can then be used as an approximation to the error rates of Type-II QAM.

Using signal levels of $\pm A$ and $\pm B$ for both the $I$ and $Q$ components, the probability of bit $a$ (or bit $b$ by symmetry) being incorrect is given by (using the results derived above):

$$P_s(a,b) = \frac{1}{2} P\{\alpha A + n_e < 0\} + \frac{1}{2} P\{\alpha B + n_e < 0\}$$

$$P_s(a,b) = \frac{1}{2} \left[ \frac{1}{2} \text{erfc} \left( \frac{\alpha A}{\sigma \sqrt{2}} \right) + \frac{1}{2} \text{erfc} \left( \frac{\alpha B}{\sigma \sqrt{2}} \right) \right]$$

To work out the error rate for bits $\{c, d\}$ we consider only the two points highlighted (the other points are the same by symmetry). The lines $\kappa$, $-\kappa$ in figure 36c represent the thresholds set within the receiver ($\kappa = \frac{A + B}{2}$) assuming that the receiver tracks the Rayleigh fading signal perfectly.
\[ p'(c) = \frac{1}{2} P[aB + n_c < \alpha x] - \frac{1}{2} P[aB + n_c < -\alpha x] + \frac{1}{2} P[aA + n_c < -\alpha x] + \frac{1}{2} P[aA + n_c > \alpha x] \]

\[ = \frac{1}{2} P[n_c < \alpha(x - B)] - \frac{1}{2} P[n_c < -\alpha(x + B)] + \frac{1}{2} P[n_c < -\alpha(x + A)] + \frac{1}{2} P[n_c > \alpha(x - A)] \]

\[ = \frac{1}{2} P[n_c < -\alpha \frac{x - A}{2}] + \frac{1}{2} P[n_c < -\alpha \frac{x + A}{2}] + \frac{1}{2} P[n_c < -\alpha \frac{x + B}{2}] - \frac{1}{2} P[n_c < -\alpha \frac{x + 3B}{2}] \]

Again, by symmetry arguments this is the same for bit d.

In the Rayleigh fading case, the expected error rates can be evaluated in a similar manner to that for the Binary PSK above, yielding:

\[ \langle P_e(a,b) \rangle = \frac{1}{2} \left( \frac{1}{2} \left( \frac{1}{\sqrt{1 + \frac{1}{\gamma N_0}}} \right) + \frac{1}{2} \left( \frac{1}{\sqrt{1 + \frac{1}{\gamma N_0}}} \right) \right) \]

where \( \gamma = \frac{\sigma_n^2}{\sigma_x^2} = \frac{2N}{N_0} \)

\[ \gamma_0 = 2N \cdot \frac{\bar{E}}{N_0} = \gamma \cdot \bar{E} = \gamma \cdot (A^2 + B^2) \]

\[ \langle P_e(a,b) \rangle = \frac{1}{4} \left( \frac{1}{\sqrt{1 + \frac{A^2 + B^2}{\gamma_0}}} \right) + \frac{1}{4} \left( \frac{1}{\sqrt{1 + \frac{A^2 + B^2}{\gamma_0}}} \right) \]

\[ \langle P_e(c,d) \rangle = \frac{1}{2} \left( \frac{1}{\sqrt{1 + \frac{A^2 + B^2}{k_1 \gamma_0}}} \right) + \frac{1}{4} \left( \frac{1}{\sqrt{1 + \frac{A^2 + B^2}{k_2 \gamma_0}}} \right) - \frac{1}{4} \left( \frac{1}{\sqrt{1 + \frac{A^2 + B^2}{k_3 \gamma_0}}} \right) \]

where \( k_1 = \frac{B - A}{2}, \ k_2 = \frac{3A + B}{2} \) and \( k_3 = \frac{A + 3B}{2} \)

As can be seen from these equations (and from figure 37), the bit-error rates are now different for the pairs of bits \{a, b\} and \{c, d\}, and can be controlled by changing the values of \( A \) and \( B \) whilst keeping the average signal level the same (figure 38). Equations 13, 14 are more complete versions of the approximate equations of Morimoto et al in [75]. Figure 37 compares these theoretical expressions with results from the simulation developed, assuming...
that the receiver is able to track the multipath fading perfectly. This is an unrealistic assumption, which is discussed later in this chapter.

![Diagram showing bit-error rate (BER) vs. mean output SNR for QAM16 Type-III via a Rayleigh-fading Channel.]

*Figure 37: P(e) for PSK, QPSK and QAM16 Type-III (B=3A) via a Rayleigh-fading Channel*

Although figure 38 shows that the bit-error rate can be tuned by varying the properties of the modulation scheme, the modulation depth required to give equal bit-error rates depends on the quality of the channel. This is not a quantity that can be known when a system is being designed and so tuning the modulation to give equal bit-error rates is not a viable option.

![Diagram showing BER vs. modulation depth for QAM16 Type III via a Rayleigh Fading Channel at 18 dB.]

*Figure 38: BER Effect of Modulation Depth for QAM16 Type III via a Rayleigh Fading Channel at 18 dB*

Having noted the similarity between QAM16 types II and III, figure 39 provides a comparison of the QAM16 Type II simulation with the theoretical expressions derived for QAM16 Type III. In order to create a fair comparison, the modulation schemes have been normalised to operate at the same average power level. As the average power for Type II was...
originally less than for Type III, the effect of this is to increase the signal power in Type II. The bit-error rate on bits a and b is therefore reduced, though this is not obtained without cost. The change in the modulation scheme moves the constellation points closer together, and so Type II is more susceptible to errors in bits c and d. Although Type II provides a lower bit-error rate than Type III for bits a and b, it performs over twice as poorly for bits c and d in the $\gamma=20\text{dB}$ region.

![Figure 39: Comparison of QAM16 Types II and III](image)

Filters

In order to keep the transmission bandwidth finite, the output from the modulator needs to be band-pass filtered (which, in a base-band simulator corresponds to low-pass filtering), as shown in figure 40. In order to achieve optimal performance, the channel (assumed to be an AWGN channel) must not exhibit inter-symbol-interference (ISI). This is guaranteed if the filter satisfies the Nyquist Vestigial Symmetry Theorem [76].

![Figure 40: Simple Transmission Model](image)
Matched filters ([9] for example) do not ensure the lack of ISI, but can be designed in such a way as to do so by making them satisfy the Nyquist theorem. Conversely, it is not sufficient to only meet the conditions of a Nyquist channel. Feher [77] shows that, for optimal $P_e$ performance, a Nyquist channel should be used where the Nyquist filter partitioning satisfies the matched filter criteria, that is $H'(f) = H''(f)$. The requirement to have $H'(f) = H''(f)$, and for $H(f) = H'(f) \cdot H''(f)$, means that $H'(f) = H''(f) = \sqrt{H(f)}$.

![Figure 41: Optimal Filter Arrangement for AWGN channel](image)

**Raised Cosine Filter**

The filter that is usually used in mobile radio [67, 78] is the raised cosine filter. This is defined by:

$$ H(f) = T \quad \text{for } 0 \leq |f| \leq \frac{1}{2T} (1 - \alpha) $$

$$ H(f) = T \cdot \left(1 + \cos \left(\pi \cdot \frac{f - \frac{1}{2T} (1 - \alpha)}{\frac{2\alpha}{2T}}\right)\right) \quad \text{for } \frac{1}{2T} (1 - \alpha) \leq |f| \leq \frac{1}{2T} (1 + \alpha) $$

$$ H(f) = 0 \quad \text{for } |f| \geq \frac{1}{2T} (1 + \alpha) $$

This can be seen to satisfy the Nyquist Vestigial Symmetry Theorem (see figure 42a), and thus guarantees ISI-free transmission. The parameter $\alpha$ controls the bandwidth of the filter, the typical value being 0.35 (corresponding to a bandwidth of approximately $1.5 \times T$). The filter taps required for the implementation of this filter are given by:

$$ h(t) = \frac{1}{1 - (2\pi \alpha)^2} \cdot \frac{T}{\pi} \sin \frac{\pi t}{T} \cdot \cos \frac{\pi \alpha t}{T} \quad \text{if } T \neq 2\alpha | $$

$$ h(t = 0) = 1 $$
\[ h(t) = \frac{T}{\pi} \cdot \sin\left(\frac{\pi t}{T}\right) \cdot \frac{\pi}{4} \quad \text{if } T = 2\alpha \]

(which are derived in Appendix B, page 179)

**Square Root Raised Cosine Filter**

In order to achieve optimal error performance, the square root (in frequency space) of the raised cosine filter is required, defined by:

\[ H'(f),H''(f) = \sqrt{T} \]

\[ H'(f),H''(f) = \frac{T}{2} \left(1 + \cos\left(\pi \cdot \frac{f - T^2}{T^2} \cdot (1-\alpha)\right)\right) \]

\[ \text{for } 0 \leq |f| \leq \frac{1}{2T}(1-\alpha) \]

\[ H'(f),H''(f) = 0 \]

\[ \text{for } \frac{1}{2T}(1-\alpha) \leq |f| \leq \frac{1}{2T}(1+\alpha) \]

\[ H'(f),H''(f) = \frac{1}{2T}(1+\alpha) \]

The filter taps required for the implementation of this filter (derived in Appendix B, page 182) are given by:

\[ h(t) = \frac{1}{1-(\frac{4\alpha}{\pi})^2} \cdot \frac{1}{\sqrt{T}} \left\{ \frac{T}{\pi} \sin\left(\frac{\pi (1-\alpha)}{T}\right) + \frac{4\alpha}{\pi} \cdot \cos\left(\frac{\pi (1+\alpha)}{T}\right) \right\} \]

for which two special cases are required:
\[ h(t=0) = \frac{1}{\sqrt{T}} \left\{ \frac{4\alpha}{\pi} + 1 - \alpha \right\} \]

\[ h(t = \pm \frac{T}{4\alpha}) = \frac{\alpha}{\sqrt{T}} \left\{ \frac{2}{\pi} \sin \left( \frac{\pi(1-\alpha)}{4\alpha} \right) + \cos \left( \frac{\pi(1-\alpha)}{4\alpha} \right) \right\} \]

![Figure 43a: Frequency Response of Square-Root Raised Cosine Filter](image1)

![Figure 43b: Filter Taps for Square-Root Raised Cosine Filter](image2)

The square-root raised cosine filter does not, therefore, satisfy the requirements for ISI-free transmission. Thus, ISI is present in the transmitted signal, but the presence of a second (identical) filter in the receiver removes this ISI. This is shown quite clearly by the "Eye" diagrams: figure 44 shows a plot of the I-component of the modulated signal after the transmitter filter and after the combined effects of the transmitter and receiver filters (for a perfect channel, without noise). The combined effect of the two filters shows four clearly defined amplitude levels at the symbol decision points (\(t=4,12,20,28\ldots\)). The single application of the filter does not lead to clearly defined amplitude levels at the decision points as there is interference from adjacent symbols.
Fade-tracking in Multipath Environments

In order to be able to demodulate an amplitude modulated (or phase and amplitude modulated) signal the fading effect of the channel needs to be known by the receiver. In a moving multipath environment, this fading effect is a rapidly varying signal (as seen in figure 33, page 50), and so must be tracked dynamically. This chapter presents various different methods of tracking this fading effect. In order to compare these, and to investigate how they perform as $f_d T_s$ is varied, graphs of the bit-error rate against $f_d T_s$ are included for a mean channel noise ratio, $\gamma = \infty$. The bit-error rate for other values of $\gamma$ can be approximated by adding the bit-error rate due to fade tracking to the bit-error rate shown in figure 37 (page 56).

Automatic Gain Control (AGC)

Assuming that the modulated symbols are equiprobable, it is possible to get an estimate of the fading effect of the channel by averaging the received amplitude over $n$-symbols (as shown in figure 45). This does not provide an estimate of the phase change.
introduced by the channel, which is assumed to either be obtained from the carrier recovery circuit, or not to be required (for example, if differential phase encoding\textsuperscript{16} is used).

In the simulation model developed, the phase recovery of the channel is assumed to be perfect, which leads to a bit-error rate for bits 'a' and 'b' of zero when $\gamma = \infty$. Figure 46 shows how the bit-error rate for bits 'c' and 'd' varies with the rate of fading and also with the number of symbols, n, over which the AGC operates.

\textit{Figure 45: Automatic Gain Control}

The AGC is unable to follow rapid variations in the fading effect, and requires the symbols to be equiprobable over $n$-symbols. Further, figure 46 shows that the optimal

---

\textsuperscript{16} In differential phase encoding, the phase of the transmitted signal is the \textit{difference} in phase between the current and previous symbols. Assuming that the phase effect of the channel remains approximately constant between successive symbols, the receiver can cancel out the phase effect of the channel almost entirely.
The number of symbols over which the AGC is taken depends greatly on the rate of fading in the channel.

**Pilot Assisted Automatic Gain Control (PAAGC)**

A solution to these short-comings (at the expense of decreased bandwidth efficiency) is the Pilot Assisted Automatic Gain Controller (PAAGC) of figure 47. In this scheme, the transmitter sends a known phasor periodically. Once the receiver is synchronised, it has regular knowledge of the fading effect of the channel, and can then interpolate (usually linearly, though not always) adjacent phasors in order to obtain an estimate of the fading effect (both amplitude and phase) for a given input symbol.

The PAAGC scheme utilises the phasors before and after the received symbol by delaying the received phasors until the next pilot is received. This provides an estimate of the fading effect of the channel, and can be tuned (by varying the spacing of the pilot symbols) to give a particular bit-error rate based on the expected rate of variation of the fading pattern for the targeted application.

If the current pilot symbol is $p_i$, and the previous pilot symbol is $p_{i-1}$, then the interpolated channel effect for the $k^{th}$ symbol after pilot $p_{i-1}$ is given by:

$$\text{amplitude} = |p_{i-1}| + k \cdot \frac{|p_i| - |p_{i-1}|}{\text{spacing}}$$
\[ \text{phase} = \arg(p_{i-1}) + k \cdot \frac{\arg(p_i) - \arg(p_{i-1})}{\text{spacing}} \quad \text{if } -\pi < \arg(p_i) - \arg(p_{i-1}) < \pi \]

The phase effect is slightly more complicated as the assumption has to be made that the phase change is less than \( \pi \), and therefore that the direction of the phase change is in the direction of least phase change. If the fading is sufficiently severe, or if the pilots are spaced too far apart, then this assumption is not valid and the interpolation of the phase cannot be accomplished.

![Figure 48: Bit-error rate vs. \( f_d T \) for Pilot Assisted AGC (\( \gamma = \infty \))](image)

Figure 48 shows how the bit-error rates associated with all the bits that make up a QAM16-III symbol vary with the pilot spacing and the rate of fading of the channel (points accurate to about 1 part in \( 10^5 \)). In producing these results, the phase effect of the multipath channel has been estimated from the pilot symbols (rather than assuming perfect estimation, as for the AGC simulation).

**Spatial Diversity**

Spatial diversity\(^{17}\) schemes \([79, 80]\) have their basis in the fact that the fading pattern associated with the multipath environment varies rapidly in terms of spatial location. Since the transmission is most susceptible to error when the fading is severe, the use of two

\(^{17}\) Also possible are frequency diversity (the same message signal is transmitted using a different carrier frequency), and time diversity (the same message is transmitted in different time slots). Both of these cause a reduction in the bandwidth efficiency, and so are not considered further in this thesis.
uncorrelated fading patterns switched by received amplitude, reduces the frequency of occurrence of the severe fades. The system schematic is outlined in figure 49 below. Note that as there are effectively two independent receivers, each needs to have its own filter and fade tracking modules, but the demodulation stage can be common to all receivers.

![System Schematic](image)

**Figure 49: Pilot Assisted Fade Tracking with 2nd Order (Spatial) Diversity**

The selection of which receiver is used is based on hysteresis switching at the input to the demodulator, based on the received signal strength. Each receiver receives the same symbol, but the effect of the channel is different. The received signal with greatest amplitude is therefore the signal with least fading associated with it, and is thus least susceptible to errors due to additive noise. Figure 50 compares the bit-error rate for non-diversity and diversity (2nd order spatial diversity). The use of diversity produces a marked improvement in the bit-error rate, particularly when the channel quality is good.
Figure 50: Comparison of Diversity and Non-Diversity Schemes
\((f_d T_s = 0.001, \text{Pilot Spacing} = 10)\)

It is interesting to note that there is little change in BER as \(f_d T_s\) is altered provided the pilot spacing, \(ps\), is adjusted so that \(ps \cdot f_d T_s\) is constant. Starting from the equation for the fading pattern, equation 10:

\[
\frac{\Delta \text{fade}}{\Delta \text{symbol number}} = \frac{\Delta \text{fade}}{\Delta k} = \sum_{i=1}^{N} (R_i + jS_i) \cdot 2\pi j \cdot f_d T_s \cdot \cos \frac{2\pi i}{N} \cdot e^{2\pi j f_d T_s \cdot \cos \frac{2\pi i}{N}}
\]

\[
= 2\pi \cdot e^{jA} \cdot f_d T_s \cdot \sum_{i=1}^{N} (R_i + jS_i) \cdot \cos \frac{2\pi i}{N} \cdot e^{2\pi j f_d T_s \cdot \cos \frac{2\pi i}{N}}
\]

\[
\left| \frac{\Delta \text{fade}}{\Delta k} \right| \approx 2\pi \cdot f_d T_s \cdot \text{fade}
\]

Consequently, the fractional change in the fading pattern is given by:

\[
\left| \frac{\Delta \text{fade}}{\text{fade}} \right| \approx 2\pi \cdot f_d T_s \cdot \Delta k \text{ Equation 15}
\]

The accuracy of the fade tracking is primarily determined by the amount of change in the fading pattern between adjacent synchronising pilot phasors (\(\Delta k = ps\)). Equation 15 shows that this is approximately constant if \(ps \cdot f_d T_s\) is constant. Figure 51 shows the bit-error rates associated with transmission at \(\gamma = 20\text{dB}\) for diversity and non-diversity schemes.
The lines are not smooth because of the approximations made in deriving equation 15, and the statistical nature of the simulations.

Figure 51: Comparison of Non-diversity and Diversity schemes in terms of distance travelled between received pilots at 20dB (no differential phase encoding)

Figure 51 shows that the PAAGC scheme for tracking the fading environment is unable to track the fading at all when the mobile unit has travelled in the region of a wavelength between received pilots. From this, it can be seen that there is no correlation in the signals received at two antennae separated by a wavelength (about 35cm at 850MHz).

Error Correcting Codes

The purpose of an error correcting code is to use some excess capacity in a communication channel to provide information which can detect and correct transmission errors in either the original data or in the added information. It is not possible to be able to correct all errors that occur, but it is possible to correct more and more errors as the amount of error correcting information is increased. Table 7 shows an example (from [81]) of adding an error correcting code to the codewords 11, 00, 10, 01.
Table 7: Example error correcting code

In this example, all received codewords in a given column can be corrected to the originally transmitted code (the code at the top of the column), and in doing so all single errors are corrected and so are some dual errors.

There are two fundamental kinds of error correcting code: the block code (as in the above example) and the convolutional code. Block codes split the incoming data into blocks containing k-symbols, and adds n-k symbols of error correcting information to produce n-symbol codewords for transmission. Convolutional codes operate on the code sequence as a whole, producing a set of symbols for transmission that uniquely represent the input data. When an error occurs in either type of code, decoding consists of making a “best guess” at what the received data should have been, which for convolutional codes could involve looking at a very long stream of symbols.

In this thesis, only block codes are considered as their implementation is easier and their error correcting capabilities are more easily analysed mathematically. In particular, Bose-Chaudhuri-Hocquenghem codes (BCH codes) and Reed-Solomon codes (which are a subset of BCH codes) are considered. However, it is not necessary for this thesis to enter into a mathematical description of these codes (see [81], and [82, 83, 84] for BCH codes; [85] for RS codes) and how they are derived. To be able to control the bit-error rate of the received data it is sufficient to discuss the error properties of the data after error-correction.

**BCH codes**

BCH codes can correct t errors in n message bits (of which k are data bits). For any t (<n/2) and m, there is a BCH code of length \( n = 2^m - 1 \) which corrects all combinations of t or fewer errors, with no more than mt parity-check symbols. For a given error probability, \( p \), the
probability of successfully correcting all the errors is given by the probability that there are \( t \) or fewer errors in \( n \) bits:

\[
p(\text{errors correctable}) = \sum_{i=0}^{n} \binom{n}{i} p^i (1-p)^{n-i} \quad \text{Equation 16}
\]

from the Binomial distribution. This assumes that the probability of a given bit being incorrect is independent of all other bits, which is only true in an AWGN-channel. In order to formulate an equation for the error probability after error correction, the expected number of errors corrected is required. This is given (per n bit block) by:

\[
E[\text{number of errors corrected}] = \sum_{i=0}^{n} i \binom{n}{i} p^i (1-p)^{n-i} \quad \text{Equation 17}
\]

since \( E[x] = \sum_{x} x \cdot p(x) \)

The overall error probability is then:

\[
P(e) = p - \frac{1}{n} \sum_{i=0}^{n} \binom{n}{i} p^i (1-p)^{n-i} \cdot i \quad \text{Equation 18}
\]

For a Rayleigh fading channel, equations 16-18 are only an approximation as the errors occur in bursts (concentrated in the regions of deep fading).

The graphs that follow compare the results obtained from a BCH error correcting package\(^\text{18}\) (points: accurate to about 1 part in \( 10^7 \)) with those predicted from equation 18 (curves), for additive white gaussian noise. The experimental points are seen to be an accurate match to the theoretical curves, thus validating both equation 18 and the software implementation.

\(^{18}\) The author would like to thank Mr D.K. Biwott for the development of this package.
Reed-Solomon Codes

Reed-Solomon codes are a class of non-binary BCH codes, working on blocks of $n$ m-bit symbols as opposed to blocks of $n$ bits. The RS code is capable of correcting $t$ symbol errors in every $n$ symbols regardless of the number of bit errors within the symbols being corrected. The expression for the BER after RS error correction is given by:

$$P(e) = p \frac{-1}{n} \cdot \frac{\bar{x}}{m} \cdot \sum_{i=0}^{n} C_i \cdot \rho^i (1-\rho)^{n-i} \cdot i \quad \text{...........................................Equation 19}$$
where $\rho = \text{symbol error rate} = 1 - (1 - p)^m$

$$\bar{x} = \text{average number of errors per symbol containing errors}$$

$$= \frac{1}{\rho} \sum_{j=0}^{m} \binom{m}{j} \cdot p^j \cdot (1 - p)^{m-j} \cdot j$$

(the factor of $\frac{1}{\rho}$ makes $\bar{x}$ the average number of errors per symbol that contains errors rather than the average number of errors per symbol).

The graphs that follow compare the results obtained from an RS error correcting package [86] (points: accurate to about 1 part in $10^7$) with those predicted from equation 19 (curves), for additive white gaussian noise.

Figure 54: Error Correcting Capability of RS (15, k, i) code (4-bit symbols)
Figure 55: Error Correcting Capability of RS (63, k, t) code (6-bit symbols)

Figure 56: Error Correcting Capability of RS (255, k, t) code (8-bit symbols)

Scrambling / Interleaving

As stated before, equations 16-18 only apply when the errors are independent events. In order to retain an approximation to this for Rayleigh fading channels, k groups of n bits can be interleaved. The affect of this is to increase the duration of each of the encoded blocks by a factor k, and in doing so reduce the effective length of fade. Depending on the number of codes interleaved, and the duration of fade, this can lead to the errors due to fades becoming...
effectively independent, making equations 16-18 applicable. There is an inherent system delay of \( n \cdot k \) introduced.

Although it is possible to achieve the same result with dedicated bit-order scramblers, there is no advantage gained as the error correcting codes do not care about the distribution of errors within the \( n \) bits of the encoded data.

**Transmission Efficiency**

Having developed all the individual stages of the end-to-end transmission system, it is necessary to formulate an equation (equation 20) for the overall efficiency of the transmission incorporating:-

- **M-level modulation** \( \log_2 M \) bits/symbol
- **Filtering**, filter bandwidth \( = \frac{1}{1+\alpha} \cdot \) symbol rate
- **Error correcting codes [bit \( i \)]** \( n_i \) symbols transmitted for every \( k_i \) data symbols
- **Pilot assisted fade tracking** proportion of data symbols \( = 1 - \frac{1}{\text{pilot spacing}} \)

Thus,

\[
\eta = \left(1 - \frac{1}{\text{pilot spacing}}\right) \frac{1}{1+\alpha} \sum_{i=1}^{M} \frac{\kappa_i}{n_i} \quad \text{Equation 20}
\]

This will be used in Chapter Six to control the data rate input to the transmission system to achieve a fixed rate of symbol transmission.

**Summary**

This chapter has outlined the stages necessary for the simulation of transmission via a Rayleigh fading channel, based on the assumptions that:-

i) there is no interference at the receiver between adjacent symbols,
ii) carrier recovery is achieved perfectly

Performance of several multi-bit-per-symbol modulation schemes has been considered along with several techniques required to compensate for the fading effect of the Rayleigh fading channel, both in terms of estimation of the fading effect and in terms of mathematical error correcting codes applied to the received data. Mathematical analysis of both BCH and RS error correcting codes has been used to validate computer implementations of both.

The simulation results confirm that QAM16 modulation leads to different error rates for the different bits that make up a given symbol. It is this fundamental property that will be targeted throughout the remainder of this thesis, in order to decrease the amount of error correction that is required to enable reliable transmission of compressed video. The simulation has been validated against theoretical expressions and existing published results ([63, 87] for example).
Chapter Four: Error Resilience in MPEG-II
Chapter Four: Error Resilience in MPEG-II

No matter how the video data is transmitted, there will always be some data bits that become corrupted. In order to consider an overall transmission system, it is therefore important to investigate how these corrupt bits affect the decoding process, and at what level these errors are acceptable. This is particularly important when the transmitted data is compressed, as the errors are amplified by the decompression process.

This chapter considers MPEG-II [42] as a generic frequency transform based video compression standard, although the discussion applies equally to other video compression standards (MPEG-I[41], H.261[34], H.263[43], etc.). Since MPEG-II utilises several different techniques for achieving compression ratios in the order of fifty times (spatial and temporal compression, and entropy coding), corruption of the MPEG-II bitstream may well be severe, affecting the reconstructed video in different ways depending on how the data is damaged.

Having discussed the effect of errors, methods of efficiently reducing the impact on the decoded video are considered. There are essentially two different (and compatible) approaches [88]: re-coding the bitstream so that the significance of the errors is reduced, and error-concealment techniques which exploit any remaining redundancy in the compressed video to estimate the corrupted data. The first of these inherently leads to a discussion on the importance of consideration of transmission errors on the development of coding schemes.

MPEG-II Statistics

An understanding of how errors affect MPEG-II compressed video data can only be achieved if the distribution of the data bits (in terms of elements of the bitstream) is known, although this will vary with the amount of compression required of the encoder. Table 8 presents this distribution for various standard test sequences encoded at different output bit-rates. From this table, it can be seen that the bulk of the video bitstream is comprised of macroblock headers and DCT coefficients. These figures are averages over an entire video sequence, and are subject to significant local variations within the bitstream. I-frames, for
<table>
<thead>
<tr>
<th>Sequence</th>
<th>Resolution</th>
<th>Coding Rate (Mbits/s)</th>
<th>Compression Ratio</th>
<th>Outside Slice</th>
<th>Slice Header</th>
<th>Macroblk Header</th>
<th>DCT Coeffs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Salesman</td>
<td>352*288</td>
<td>0.5</td>
<td>122</td>
<td>&lt;1.0%</td>
<td>3.4%</td>
<td>35.5%</td>
<td>60.3%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1.0</td>
<td>61</td>
<td>&lt;1.0%</td>
<td>1.7%</td>
<td>18.0%</td>
<td>79.9%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2.0</td>
<td>30</td>
<td>&lt;1.0%</td>
<td>&lt;1.0%</td>
<td>9.6%</td>
<td>89.4%</td>
</tr>
<tr>
<td>Mobile &amp; Calendar</td>
<td>704*480</td>
<td>2.0</td>
<td>101</td>
<td>&lt;1.0%</td>
<td>1.3%</td>
<td>25.0%</td>
<td>63.5%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3.0</td>
<td>68</td>
<td>&lt;1.0%</td>
<td>&lt;1.0%</td>
<td>25.8%</td>
<td>73.1%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4.0</td>
<td>51</td>
<td>&lt;1.0%</td>
<td>&lt;1.0%</td>
<td>19.7%</td>
<td>79.5%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6.0</td>
<td>34</td>
<td>&lt;1.0%</td>
<td>&lt;1.0%</td>
<td>13.7%</td>
<td>85.8%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>9.0</td>
<td>23</td>
<td>&lt;1.0%</td>
<td>&lt;1.0%</td>
<td>9.3%</td>
<td>90.3%</td>
</tr>
<tr>
<td>Football</td>
<td>704*480</td>
<td>2.0</td>
<td>101</td>
<td>&lt;1.0%</td>
<td>1.4%</td>
<td>45.8%</td>
<td>52.6%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3.0</td>
<td>68</td>
<td>&lt;1.0%</td>
<td>&lt;1.0%</td>
<td>33.5%</td>
<td>65.5%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4.0</td>
<td>51</td>
<td>&lt;1.0%</td>
<td>&lt;1.0%</td>
<td>25.3%</td>
<td>74.0%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6.0</td>
<td>34</td>
<td>&lt;1.0%</td>
<td>&lt;1.0%</td>
<td>16.9%</td>
<td>82.6%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>9.0</td>
<td>23</td>
<td>&lt;1.0%</td>
<td>&lt;1.0%</td>
<td>11.4%</td>
<td>88.2%</td>
</tr>
<tr>
<td>Greenleaf</td>
<td>1920*1040</td>
<td>4.0&lt;sup&gt;19&lt;/sup&gt;</td>
<td>300</td>
<td>&lt;1.0%</td>
<td>&lt;1.0%</td>
<td>32.6%</td>
<td>66.9%</td>
</tr>
<tr>
<td>[89]</td>
<td></td>
<td>9.0&lt;sup&gt;19&lt;/sup&gt;</td>
<td>133</td>
<td>&lt;1.0%</td>
<td>&lt;1.0%</td>
<td>32.0%</td>
<td>67.6%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>15.0&lt;sup&gt;19&lt;/sup&gt;</td>
<td>80</td>
<td>&lt;1.0%</td>
<td>&lt;1.0%</td>
<td>30.4%</td>
<td>69.6%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>25.0</td>
<td>48</td>
<td>&lt;1.0%</td>
<td>&lt;1.0%</td>
<td>21.8%</td>
<td>78.0%</td>
</tr>
</tbody>
</table>

Table 8: MPEG-II Bitstream Contents

example, have a higher proportion of bits assigned to DCT coefficients. The number of bits assigned to other header information is approximately constant for a given image resolution.

It is also important to know how an error in a particular kind of data actually affects the decoded video. From the description in Chapter One (page 27) of the MPEG-II bitstream, table 9 can be constructed.

---

<sup>19</sup> Rate requested of the MPEG-II encoder, but not achieved. Lowest bit-rate achieved for Greenleaf is about 16Mbits/s (approximately 75x compression).
<table>
<thead>
<tr>
<th>Bitstream item</th>
<th>Effect of Error(s)</th>
<th>Synchronisation lost until</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequence Start Code</td>
<td>Decoder not started</td>
<td>End of sequence</td>
</tr>
<tr>
<td>Sequence Header</td>
<td>Damage to image properties (size, frame rate, etc.). This leads to damage until the end of the sequence.</td>
<td>End of sequence</td>
</tr>
<tr>
<td>Group of Pictures</td>
<td>Depends on application - pertains to editing data.</td>
<td>Picture data damaged until next I-frame</td>
</tr>
<tr>
<td>I-/P-Picture header</td>
<td>Damage to / loss of all information relating to a particular frame. For an I-/P-frame, this results in damage to all picture information until the next I-frame.</td>
<td>Picture data damaged until next I-/P-frame</td>
</tr>
<tr>
<td>B-Picture header</td>
<td>Damage to / loss of all information relating to a particular frame. For B-frames, this results in damage to picture information until the next I- or P-frame.</td>
<td></td>
</tr>
<tr>
<td>Slice header</td>
<td>Loss of slice data</td>
<td>Next slice</td>
</tr>
<tr>
<td>Macroblock header</td>
<td>Damage to motion vectors and coded block pattern.</td>
<td>Next slice</td>
</tr>
<tr>
<td>Block (DCT Coefficients)</td>
<td>Decode error for VLC codes, leading to damage to the error signal for this block.</td>
<td>Next slice</td>
</tr>
</tbody>
</table>

*Table 9: Affect of MPEG-II Bitstream Errors*

Bitstream errors can be classified into three different types:–

i) syntax elements are corrupted causing the decoder to expect a bitstream with different content. This is catastrophic for the decoding process, but fortunately there is relatively little syntactic information, and the actual parameters defining the overall form of the bitstream can be considered *a priori* knowledge in most situations.

ii) the error changes the value of a code, but synchronisation is not affected (code length unchanged). This form of error only affects the decoded video in a local area.

iii) the error causes a change in the length of a codeword (VLC) and also a change in the decoded value. Synchronisation is immediately lost leading to damage to the video over a large area.
Of these classifications, type iii) is by far the most likely to occur, although the probabilities actually depend on the properties of the VLC codebook.

**Decoder Error Resilience and Synchronisation**

The standard ISO MPEG-II decoder [90] has been used as the video decoder throughout this research. Figures 57 and 58 show the effect of errors in terms of the PSNR of the decoded video\textsuperscript{20}, and some examples of the damage caused to a video sequence are shown in figures\textsuperscript{21} 59.0 to 59.14 for a white noise error rate of 0.005%. These frames quite clearly show the sensitivity of the decoding process to errors in the video bitstream, and also the loss of bitstream synchronisation until the start of the next slice.

\[\text{Figure 57 : Effect of Errors in MPEG-II Coded Bitstreams} \]
\[\text{(Mobile & Calendar coded at 4Mbits/s)}\]

\textsuperscript{20} If the decoder is unable to decode a frame (due to loss of the picture header, for example), then the decoder knows nothing about that frame. The overall decoded sequence is shortened, and the PSNR comparisons lose frame synchronisation.

\textsuperscript{21} The choice of Mobile & Calendar for the test sequence for these particular results is based on the visibility of errors within the frames (due to the high contrast within the frames).
Figure 58 in particular shows a significant increase in image quality as an I-frame is decoded (frames 0, 12, 24, ...). This is to be expected as the other frames are predictions based on the I-frame and so are susceptible to errors that are already present in the decoded I-frames. Both figures show significant variations (in excess of 10dB) in the decoded image quality.
Figure 59: Effect of Errors in an MPEG-II Coded Bitstream
(Mobile & Calendar coded at 4Mbits/s, Error Rate = 0.005%)

Figure continued overleaf...
Persistence of Errors

The frames of figure 59 show clearly that decoder errors persist until the next I-frame is decoded, except where the motion in the sequence reveals new information. Not immediately obvious from these frames is that damage to B-frames only persists until the next I- or P-frame. Therefore, using the recommended I-frame spacing of 12 frames, and P-frame spacing of 3 frames, I-P-frame damage can persist for up to 0.5s whereas B-frame damage remains for less than 0.1s. As a result, errors in B-frames can be considered to be less significant than errors in I- and P-frames, although their presence is still visually annoying.

VLC Synchronisation

Earlier in this chapter it was pointed out that errors in the video bitstream would most likely lead to complete loss of synchronisation of the decoder with the bitstream (due to a change in length of a VLC codeword). Figures 60a and 60b [from 91,92] below show the effect of being able to re-synchronise the decoder to the bitstream at each slice (—), macroblock (—), block (---) and VLC code (—).

For both intra- and inter- (predicted) frames, there is little advantage to be gained from synchronising at each VLC codeword. In intra-frames it is sufficient to synchronise at the start of each block (and therefore provide some correct information for each block of image). Inter-frames primarily require synchronisation at the start of each macroblock, to ensure that the prediction information is decoded correctly for all of the image (the information contained
in the block level is just the error signal between the predicted and actual frames, and is less important provided the prediction is good).

Therefore the most important aspect for MPEG-II decoding is the successful location of the macroblock headers, and the ability to decode some information for every block of the image. Figure 61 shows the effect (in terms of PSNR) of losing some of the DCT coefficients from the end of the zigzag scan (in producing this figure, only the DCT has been considered: there is no quantization, motion estimation, etc.).

![Figure 61: PSNR vs. Number of Retained DCT Coefficients (Frame #1 of Mobile & Calendar)](image)

It should be noted that this graph shows the problem involved with the use of PSNR as an image quality measurement (see Appendix C). As such, this graph can only be used as a rough indication of image quality for a given number of DCT coefficients. Figure 62 shows a 64x64 pixel section from the first frame of Mobile & Calendar, for selected numbers of DCT coefficients being retained.
Figures 61 and 62 show that, although for accurate representation of detail within the images many DCT coefficients are required, reasonable approximations to the images can be obtained with only some of the DCT coefficients. Figure 63 for example shows the entire frame for only the first 5 DCT coefficients being retained.

Having noted that an improvement in the image quality can be achieved by synchronising the decoding of the bitstream more often, and that the image quality degrades steadily as the number of DCT coefficients discarded is increased, techniques for improving the transmission of MPEG-II coded data can be developed.
MPEG-II Transcoder

The concept of an “MPEG-II transcoder” is introduced in Appendix E [93] and, independently, in [94]. The purpose of the transcoder is to improve the transmission properties of the MPEG-II coded data by changing the way in which the bitstream is encoded (figure 64). The advantage of transcoding is the ability to target a particular application without having to completely redesign the encoding and decoding process. If a coding standard is adopted as the de facto standard, then it is unrealistic to introduce a new standard for a particular application that has similar properties. It is much more useful to be able to translate to/from the coding standard into the form required.

\[\text{MPEG-II} \rightarrow \text{Transcoder} \rightarrow \text{Channel} \rightarrow \text{Inverse Transcoder} \rightarrow \text{MPEG-II}\]

*Figure 64: Transcoding*

The transcoder has to have the following properties in order to be useful:

i) the quantity of data transmitted through the channel has to be in a fixed ratio to the amount of data input, as there is no feedback connection into the encoder.

ii) the output bitstream should be a good representation of the input bitstream (ideally, they should be identical).

iii) the amount of processing in the transcoder should be considerably less than the original encoding and decoding process (else it is more efficient to create specialist encoders/decoders for each application).

Having described the properties required of the transmitted data, and having defined the constraints introduced by the use of a transcoder, possible transcoding models can be defined.
DCT Interleaving\textsuperscript{22}

The MPEG-II coding standard is typical of all transform based video coding standards in that it scans the transform coefficients in approximate order of increasing frequency, and encodes the coefficients as a 2-dimensional (Run, Length) VLC. These are then stored in the output bitstream on a block by block (8x8 pixel) basis within a macroblock (figure 65). Consequently, if any block is damaged, all subsequent blocks (until the next synchronisation point in the bitstream) are completely lost.

![Figure 65: Existing macroblock layout](image)

To overcome this limitation, the VLC codes representing the DCT coefficients can be \textit{interleaved} as shown in figure 66. Within a given macroblock, the DCT coefficients are now encoded in order of approximately increasing frequency. It is not possible to guarantee that the coefficients are in ascending order of frequency, as the coefficients themselves have not been reordered, just the 2-D (run, length) encoded form. It should be noted that this rearrangement of the VLC codewords adds no overheads to the bitstream, nor introduces any potential decoding problems (such as start code emulation, or ambiguity in the decoding process) as the VLC codewords are not modified in any way.

![Figure 66: Interleaved macroblock layout](image)

However, this does not yet solve the problem of loss of synchronisation causing loss of video data until the end of the current slice (which usually means the end of the current video line). To overcome this, the decoder must be able to synchronise after each

\textsuperscript{22} Much of the literature discusses “interleaving” referring to the interleaving of error-correcting code protected blocks of data in order to approximate burst errors as Gaussian errors. Throughout this section, the word interleaving refers to the raw MPEG-II data not to the transmission packets.
macroblock. This can be achieved by introducing a “start of macroblock” start code (32 bits), but this would be prohibitively expensive in terms of the amount of data added to the bitstream (this could add as much as 40 kbits per frame to the header information for a sequence such as Mobile & Calendar which is equivalent to about 1 Mbits per second - a quarter of the typical coding rate!). An alternative, zero overhead, solution is to interleave all the macroblocks contained within a single slice as shown in figure 67. If the VLC decoder loses synchronisation due to errors, some DCT coefficients of every macroblock will already have been successfully decoded, and so some image information will exist for every macroblock in the slice. In addition, each of the macroblock headers will be decoded, and so motion estimation will not be affected by errors occurring in the DCT coefficients.

![Slice Header Diagram](image)

Figure 67: Macroblocks interleaved across a slice

Of course, such an arrangement is only an improvement if the errors occur part way through a slice and not in the header information at the very beginning of the slice (when they are as catastrophic as they are in the MPEG-II standard). Given the assumption that any bit error will be detected by the decoder, the probability of decoder failure for a given bit within a slice (with a white noise error probability, $p$) is given by:

$$p(\text{decoder failure at } j \text{ before bit } j) = \sum_{i=1}^{j} p \cdot (1 - p)^{i}$$

This is shown in figure 68 (noting that for video coded at 4Mbits/s the average slice is about 5000 bits long, though shorter for P- and B-type frames. In fact, the MPEG-II coding standard allows for slices shorter than the entire frame width, though the MPEG-II encoder used for this research does not implement this feature).

23 This is not a valid assumption, but does provide an upper bound. Formulation in terms of whether an error will lead to decoder detection is not practical as this depends on whether certain codes are possible at that point in the DCT coefficients.
The above discussion applies only for random (white noise) errors where the probability of a given bit being in error is independent of previous errors introduced in the bitstream. In many scenarios this is not the case, and errors can tend to be clustered together. If this is the case, the block of errors behaves as a single error causing decoder failure at the start of the error block (except when the burst of errors spans a slice boundary). The effective error probability is thus the probability of a bit being the start of an error burst, and so is significantly reduced. The use of scramblers in order to make error correcting codes capable of correcting bursts of errors (as discussed in Chapter Three) is thus not necessarily advantageous. If the error correcting code is not capable of correcting the entire burst of errors, then scrambling will have an adverse effect on the decoding process (as the errors are no longer clustered together). Note however, that the use of an error correcting code without a scrambler still remains advantageous.

![Graph showing probability of decoder failure before bit j](image)

*Figure 68: Probability of Decoder Failure at before bit j*

Figures 69 and 70 show the effect of interleaving across an entire slice in terms of PSNR of the decoded video. The curves show an improvement in decoded image PSNR over those of figure 58, and also exhibit reduced fluctuations and a rapid fall-off in overall bitstream decoding success above an error rate of about $10^{-4}$. As the bit error rate increases, the probability of damage to the more sensitive data (i.e. data outside macroblocks which contains information critical to the decoding process) becomes significant. Provided some method can be found to further protect syntax elements, the goal of increasing the error-
resilience in the MPEG-II coded data is achieved, although the image quality is severely dependent on the error rate.

Figure 69: Effect of Errors on Interleaved MPEG-II Bitstreams (Mobile & Calendar coded at 4Mbits/s)

Figure 70: Effect of Errors on Interleaved MPEG-II Bitstreams (Football coded at 4Mbits/s)
Figure 71 overleaf shows the effect of an error rate of 0.005% applied to an MPEG-II bitstream with interleaved DCT data. There are three types of errors present:-

i) localised noise (high frequency DCT coefficients damaged or discarded)

ii) large sections without any image data (caused by damage to syntax elements [slice or macroblock header]).

iii) sections with the wrong brightness level (caused by damage to differentially coded DC coefficients).

The last two of these are caused by damage to syntax elements within the bitstream, which is not addressed by DCT interleaving. The improvement in image quality for the areas that are damaged by errors within the DCT coefficients is substantial.
Figure 71: Effect of Errors in an Interleaved MPEG-II Coded Bitstream
(Mobile & Calendar coded at 4Mbits/s, Error Rate = 0.005%)

Figure continued overleaf…
Syntax Protection

In order to guarantee successful decoding of the MPEG-II bitstream, it is necessary to ensure that the existing synchronisation points are protected from errors. The use of "start codes" in the form 0000,0000,0000,0000,0000,0000,0000,0000,0000,0000,0000 (xxxxxxxxx denoting the actual start code) is advantageous in that they can be searched for once the decoder loses synchronisation. However, this will only occur when there are errors in the bitstream, and the start code system is susceptible to a corrupt bitstream in two ways:-

- damage to a start code causes a loss of video data until the next start code of the same (or a higher) level.

- start code emulation can occur. The structure of the video coding is such that it is not possible to accidentally create a start code. However, when errors are present in the bitstream this can no longer be guaranteed.

The requirement of being able to search for start codes provides a significant limitation on the techniques that can be considered for reducing the likelihood of start code errors. For example, suppose that a t-error error correcting code is added to the start code. The pattern to be searched for is now 0000,0000,0000,0000,0000,0000,0000,0000,0000,0000,0000 (xxxxxxxxx,ss...ss (ss...ss being the error correcting information) with up to t-errors. It is, however, not possible to guarantee that this pattern will not occur elsewhere within the bitstream, so this is not a valid solution.

The only viable alternative to start codes is the encoding of the length of the particular block of data, which inherently removes the need to be able to search for a particular pattern provided that it can be guaranteed that these codes will be received without error. Such a scheme is advocated in [95], where the slice start codes are replaced with codes indicating the length of four-slice sections, and the picture start codes are replaced with a 128-bit pseudo random bit pattern (though how this 128-bit pattern is detected in the presence of errors is not discussed).

When the bit-error rate is comparatively low (so that damage to a start code has a low probability of occurrence) then the use of start codes to provide re-synchronisation of the bitstream is effective. If there is a significant likelihood of damage to a start code occurring,
then the addition of a more error resilient method of detecting the start of a picture will lead to a more stable decoding of the video bitstream (with particular importance attached to the re-synchronisation at I- and P-frames).

**ERECD Transcoding**

An alternative method of increasing the frequency of synchronisation points within the video bitstream is the Error-Resilient-Entropy-Coding (ERECD) scheme of [94, 96], which is based on Error-Resilient-Position-Coding (ERPC) [97]. This algorithm splits the video data into fixed sized *slots* (of N bits each), whose size is transmitted (protected by an error correcting code) as part of the bitstream. As can be seen from figure 72a below, some of the slots are partially filled, whereas the remaining slots overflow. The essence of the EREC algorithm is how this is overcome, whilst guaranteeing that the bitstream can be uniquely decoded.

![Figure 72a: EREC coding](image)

The parts of slots that overflow are retained, and shifted right one slot. This data is then placed into a slot if there is room for some data in that slot (figure 72b). This process is repeated, until no data overflows a slot (figure 72c).

![Figure 72b: EREC coding (after one iteration)](image)
This process guarantees that the start of each slot can be located in the bitstream, regardless of errors. If a slot contains errors though, all data that has overflowed into other slots has to be considered invalid\(^{24}\).

The EREC scheme inherently contains an efficiency problem, as it requires the transmission of the (error protected) slot size, \(N\), which is not a property of the original MPEG-II bitstream. Further, the total number of bits transmitted is unlikely to be an exact multiple of \(N\), and so there will be some gaps (on average, there will be \((\#\text{slots} - 1)/2\) gaps in every EREC frame) in the transmitted data. It is not possible to shorten the final slot length, as the gaps do not occur at the end of the data block. For bitstream transcoding this inefficiency is a severe problem, as the output rate cannot differ from the input rate. The efficiency problem of EREC can be overcome by more complex transcoding (in [94], this is achieved by replacing four slice start codes with a single EREC framing code).

The number of slots transmitted for an EREC frame can be calculated from the macroblock headers, on the assumption that they are received error free. If not, there needs to be a mechanism for re-synchronising the decoder to the bitstream. This leads to another potential problem with the EREC scheme: start code emulation can occur within the transcoding process (as the VLC codewords can be split up at the end of a slot). The receiving transcoder cannot search for start codes if decoding breaks down, so the EREC framing code must be uniquely decodable which is impossible to guarantee within the EREC

\[^{24}\text{Although some of the data that has overflowed will still be valid, evaluation of which slots definitely contain valid data is difficult.}\]
transcoding scheme (though the probability of this occurring can be made very small by increasing the code length). Further, it must also be inserted after the last slice of a frame in order to locate the picture header accurately.

The resulting EREC algorithm is similar in many ways to the DCT coefficient interleaving already outlined. The EREC scheme provides more synchronisation points, but at the cost of introducing extra data that needs to be transmitted. The receiver is guaranteed to be synchronised for each EREC slot (typically a ‘block’). DCT coefficient interleaving only guarantees synchronisation at the start of a slice, but evenly distributes coefficients until errors are received. This is achieved at zero overhead.

Development of Compression Standards

The amount of effort required by an encoder/decoder to adopt one of the algorithms outlined above is minimal in comparison to the work performed by either the encoder or the decoder. Although unnecessary in some applications (e.g. CD-ROM), there are many occasions when the transmission is not noise free. Both of the approaches outlined in this chapter show a significant improvement in received image quality in noisy environments, indicating that the MPEG-II coding standard is (perhaps surprisingly) unnecessarily susceptible to errors.

Consequently, in the development of any coding standard, it is essential to consider whether the coding standard will be used in noisy situations. If this is the case, in developing the coding standard algorithms that minimise the effect of errors should be considered.

Error Concealment

Although not actually used in this research, a potentially important tool in transmitting acceptable quality video is that of error concealment. That is, when the video is received and known to contain errors, some of the remaining redundancy in the MPEG-II bitstream can be utilised to attempt to predict the damaged / missing video data. Numerous different approaches (summarised in [98]) have recently been discussed: some of these are listed in table 10 below.
Table 10: Summary of error concealment strategies

As can be seen from this table, all of these proposed schemes are far from perfect and their success depends mainly on the properties of the video itself. In fact, the conclusions reached in [98] are that:

i) to obtain good image quality, it is important to support error concealment already during the encoding process. [sic]

ii) use of shorter slices and concealment vectors for intra-coded macroblocks improve considerably the image quality in error-prone applications.

These conclusions are not surprising as the attempt in all these schemes (except for the use of intra concealment motion vectors) is to find additional redundancy that was ignored in the encoding process, and to utilise this in the reconstruction of the images. Since the encoder works on blocks of 8x8 pixels, it is unlikely that redundancy in the detail will be left within the encoded video once blocks are lost.

---

25 Whether or not ‘data retransmission’ can be classified as ‘error concealment’ is debatable, however it remains a valid technique for coping with transmission errors and so is included in this table.
In light of these remarks, the results presented in this thesis do not include any form of error concealment other than that which is inherent in the algorithms themselves.

**Summary**

This chapter has investigated how the MPEG-II standard responds to transmission errors. Having noted the extreme error susceptibility, the concept of DCT coefficient interleaving was introduced as a novel (zero overhead) solution which significantly improves the quality of the decoded video for a given error rate, although the quality of the received video still decreases rapidly with increasing bit-error rate. A more complex, alternative solution (which has been developed in parallel by other researchers) has been outlined for completeness of the discussion. Although very different in their approaches, the two solutions to the MPEG-II error resilience problem are based on the following key points:-

i) the probability of decoder failure at or before a given bit increases with the number of bits since the last synchronisation point. It is therefore important to synchronise the decoding of the bitstream frequently, so as to make sure that the decoder can decode some data for all of the image.

ii) image quality should decrease steadily as the error rate is increased.

The presence of methods of rearranging the video bitstream, in a zero overhead fashion, leads to a discussion on the development of coding standards. In general, standards are developed to provide maximum video compression, but with little or no regard for the possibility of transmission errors. With careful construction of the video bitstream, it is possible to significantly increase the quality of the decoded video in the presence of small amounts of noise. A brief discussion of some methods for recovering intelligible video data when the bitstream is damaged is presented. However, these methods are unable to predict detailed image data and so are not considered throughout this research.

An interesting point to note with the DCT coefficient interleaving scheme developed in this thesis is that the video is more susceptible to errors around the start of a slice than it is to errors towards the end of a slice. Chapter Three introduced transmission schemes that
inhernently provide different bit-error rates for different transmitted bits. The following chapter discusses how these two properties can be combined.
Chapter Five: Partitioning for Wireless Applications
Chapter Five: Partitioning for Wireless Applications

The previous chapter outlined two schemes for increasing the overall error resilience of an MPEG-II coded bitstream, although it was shown that for acceptable quality the overall bit-error rate had to remain low. Of the schemes outlined, the DCT coefficient interleaving scheme inherently places the most important part of the data closest to the synchronisation point. This chapter outlines how this property can be used advantageously to increase the efficiency of transmission, particularly for wireless applications. How is this efficiency increase brought about? Without prioritised partitioning, both partitions require error correcting codes, as outlined in Chapter Three, to provide a constant bit-error rate for the system. If the data can be prioritised, then the low importance data does not require as low a bit-error rate as the high importance data, and so requires less error correction. This removal of error correction from the second partition increases the efficiency of the transmission, at the cost of reduced video quality when errors occur. However, with careful construction of the prioritisation scheme, the quality of the reconstructed video can be made to decrease gracefully as the bit-error rate increases.

![Figure 73: Bitstream Partitioning](image)

Figure 73 outlines the basic principle of partitioning. The shaded area represents a single “object” from the video bitstream, with arrows showing the order in which the decoder removes bits from the partitions. In the simplest case, the partitioned object length \( l \) is just a single bit. In the more complicated systems outlined in this thesis, the “object” can be as large as an entire slice.
Targeting of Error-Rate Prioritised Channels

If the bit-error rates of the various channels associated with a single communication are different, then there are essentially two alternative strategies that can be used: add error correcting codes to the channels so that the post-error correcting bit-error rates are the same, or encode the data in such a way as to match the importance of the data to the channel error rate. The former of these is the approach taken by Hanzo et al in [55] which is outlined below. The previous chapter outlined a method of re-ordering the image data in terms of visual importance, which shall be used as the basis for developing the latter approach.

**Matched Error-Rates via ECC**

Careful choice of error correcting codes can lead to all channels having the same bit-error rate after error-correction. Once this is achieved, video data can be split into the required ratio very simply. There are, however, disadvantages associated with this approach:

i) such a scheme only works when the bit-error rate for the entire bitstream is very low as was shown in the previous chapter, when acceptable video was achieved for bit-error rates less than 0.005%. As such, it is difficult to target efficiently scenarios where the bit-error rate may fluctuate.

ii) in the mobile environment the error correcting code(s) need to be able to cope with (potentially long) bursts of errors. This either makes the error correcting codes inefficient, or requires scrambling of the data over many error correcting codes, resulting in additional delays.

**Error-Rate Based Priority Partitioning**

The technique developed in this thesis is to retain the different error resilience properties of the various channels, and to encode the data in such a way as to minimise the effect caused by the high bit-error rate channel(s). Providing that the re-coding of the data can be achieved efficiently, significant gains in efficiency can be achieved by removing error correction from all but partition 1 (assumed from now on to be the low error rate partition).
This scheme is particularly well suited to the mobile environment where long bursts of errors can occur, as there is no need to be able to correct all these errors.

The major difficulties associated with this system are:-

i) efficiently re-coding the data

ii) guaranteeing a minimum Quality of Service. However, later in this chapter, it will be shown that this is in fact a realisable target.

Partitioning the video bitstream will inherently introduce some added data to the video bitstream, either in the form of new codewords, or inefficiencies introduced by the requirement of the bitstream being decodable in the presence of errors. As a general rule, the quality of the video in the presence of errors on partition 2 will be higher if the amount of video data on partition 1 is maximised. In other words, where possible, the overheads associated with partitioning should be targeted to the error prone part of the transmitted data.

**Partitioning**

Initially, the partitioning of the video bitstream is developed in terms of the inherent prioritisation of the data introduced through DCT coefficient interleaving. The process of partitioning interleaved DCT coefficients is purely one of dividing the data and making sure that the re-combining process can reverse this process regardless of the errors present in all but partition 1. Throughout this discussion the partitioning is assumed to be into two partitions, although there is no reason why the data could not be partitioned into multiple partitions, and so the derivations that follow are based on an arbitrary number of partitions.

Before outlining particular algorithms that have been developed for partitioning the video data, it is necessary to outline the rules that the algorithms must adhere to. As errors on all but the first partition are expected to occur, it is also necessary to outline strategies to detect these errors and recreate a legal MPEG-II bitstream. These errors create a secondary problem: the re-combining process must be able to work out where to synchronise the partitions using information obtained from partition 1 only. This is made more complicated by two further requirements:-
• the partitioning ratio will not be 1:1, as there will be an error correcting code on partition 1 at least.

• the partitioning ratio needs to be achieved accurately over the entire sequence (so that the output bit-rate from the error corrector(s) is identical).

The following rules are pre-requisites of the partitioning process:-

1) The total amount of data after partitioning must always be at least as great as the original amount of data if the quality is to be preserved. The ideal partitioning location is derived below. It may not be possible to partition the data at this location, in which case partitioning should occur as soon after this location as possible. The total amount of data required to fill all the partitions may thus be more than the original amount of data, leading to part of the last partition not being filled.

2) Modifications to the bitstream cannot cause start code emulation in the re-combining process, as the start codes are used as synchronisation markers (and can indicate that decoding should be stopped, because synchronisation has been lost). This constraint has the following implications :-

• Unless the data that follows the modification is known \textit{a priori}, then the end of the data on partition 1 cannot end with ‘0’ (the first digit of the start code)\textsuperscript{26}.

• VLC codes cannot be arbitrarily split unless the data that follows them is a start code.

3) If new codewords are introduced, they must be unique. Their location in the bitstream must be such that the re-combining process would search for them. For example, they cannot follow the ESCAPE code (which the decoder expects to be followed by two fixed length codes).

4) Partitioning will always occur once, so if this is represented by a VLC codeword, it should have a similar length to EOB (which will occur up to 6 times per macroblock).

\textsuperscript{26} Actually it does not need to be as strict as this, but this \textit{guarantees} no start code emulation.
The re-combining process must always be able to find such a code, and so this code cannot occur where a fixed length code would be expected.

Table 11 below shows the information that is known in the partitioning process and in the re-combining process. From this information, formal expressions for the amount of data to send to each partition can be derived. The partitioning ratio, $p[i]^{27}$, is a priori knowledge defined by the error correcting codes in the system. The amount of data already sent to each partition, $bits[i]$, is required in order to achieve the exact partitioning ratio required over the entire bitstream (as the required ratio cannot be achieved accurately over a single “object”).

<table>
<thead>
<tr>
<th></th>
<th>Partitioning</th>
<th>Re-combining</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amount of data to partition</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>Length of partition 1</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Partitioning ratio, $p[i]$</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Bits already sent to each partition</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

*Table 11: Knowledge of bitstream properties in the partitioning and re-combining processes*

**Partitioning Equations**

Defining $L[i]$ to be the amount of data to be sent to the $ith$ partition for this “object”, then:

- total bits on partition 0 after this “object” = $bits'[0] = L[0] + bits[0]$

$$bits'[0] = \frac{p[0]}{\sum_{i=channels} p[i]} \left( \text{total bits to partition} + \sum_{j=channels} bits[j] \right)$$

$$bits'[i] = \frac{p[i]}{p[0]} \cdot bits'[0]$$

*Equation 21*

---

27 Throughout this discussion, $p[i]$ is an integer defining the distribution of bits to channels. For example, if partition 1 has a BCH(255,223,4) code and partition 2 has no error correction, then $p = \{223,255\}$. 
Therefore:

\[
L[0] = \left( \frac{p[0]}{\sum_{i=\text{channels}} p[i]} \left( \text{total bits to partition} + \sum_{j=\text{channels}} \text{bits}[j] \right) \right) - \text{bits}[0]
\]

\[
L[i] = \frac{p[i]}{p[0]} \cdot \text{bits}[0] - \text{bits}[i] = \frac{p[i]}{p[0]} \cdot (L[0] + \text{bits}[0]) - \text{bits}[i]
\]

These guarantee that, after the data is partitioned, the partitions will be in the correct ratio to the nearest bit. Due to rounding errors, they do not guarantee that \( \sum L[i] \geq \) total bits to partition, which is achieved by increasing \( L[0] \) and recalculating \( L[i] \) until this criterion is achieved. Obviously, increasing \( L[0] \) is likely to lead to \( \sum L[i] \) being more than the amount of data to be partitioned which results in a small inefficiency in the system.

Recombining Equations

In the recombining process, the partition lengths \( (L[i]) \) can be calculated from the length of the 1st partition, \( l \), and the partitioning ratio, \( p[i] \):

\[
L[0] = l
\]

\[
L[i] = \frac{p[i]}{p[0]} \cdot (L[0] + \text{bits}[0]) - \text{bits}[i]
\]

Implementation

In both the partitioning and re-combining processes, the use of \( \text{bits}[i] \) will eventually create a problem as \( \text{bits}[i] \) becomes large (and so the required accuracy is lost). Defining \( \alpha[i] = \text{bits}[i] \mod p[i] \), then:

\[
\text{bits}[i] = \alpha[i] + k \cdot p[i],
\]

so \( \sum_{i=\text{channels}} \text{bits}[i] = \sum \alpha[i] + k \cdot \sum p[i] \) where \( k \) is an integer.
\( k \) is the same for each channel as the bitstreams are synchronised after each “object” to ensure that the ratio of the total amount of data sent to each partition is correct (from Equation 21, \( \frac{\text{bits}[i]}{p[i]} = \frac{\text{bits}[0]}{p[0]} = k \)). Therefore:

\[
\text{bits}'[0] = \frac{p[0]}{\sum_{i=\text{channels}} p[i]} \cdot \left(\text{total bits to partition} + \sum \alpha[j] \right) + k \cdot p[0]
\]

\[
\alpha'[0] = \text{bits}'[0] \mod p[0]
\]

\[
\alpha'[0] = \left\lfloor \frac{p[0]}{\sum_{i=\text{channels}} p[i]} \cdot \left(\text{total bits to partition} + \sum \alpha[j] \right) \right\rfloor \mod p[0]
\]

\[
\text{bits}'[i] = \frac{p[i]}{p[0]} \cdot \text{bits}'[0] = \frac{p[i]}{p[0]} \cdot (\alpha[i] + k \cdot p[0])
\]

\[
\alpha'[i] = \text{bits}'[i] \mod p[i] = \left\lfloor \frac{p[i]}{p[0]} \cdot \alpha[i] \right\rfloor \mod p[i]
\]

Therefore, there is no difference in the achieved partitioning ratio if \( \text{bits}[i] \) is kept modulo \( p[i] \), thus removing the problem of \( \text{bits}[i] \) becoming large and losing accuracy.

**Error Detection**

Having defined the rules for partitioning the data, it is necessary to define how the re-combining process should respond when errors are present in partition 2. This, in itself, depends on the ability of the re-combining process to detect that errors are present in the bitstream.
The standard MPEG-II coding scheme can inherently detect three types of errors: -

- invalid VLC codeword (not part of the codebook)

- invalid Escape code (illegal fixed code for run length or level)

- DCT coefficient out of range (the DCT works on an 8x8 pixel block, producing 64 DCT coefficients. If decoding of the coefficients produces more than 64 coefficients, then a decoding error has occurred).

Swann and Kingsbury [91] define an additional approach to error detection. Based on a set of training video sequences, they have classified DCT coefficient values above a certain range (which varies with the coefficient index) as statistically unlikely, and hence potentially indicative of errors having occurred. Although this approach undoubtedly works in the majority of sequences, the possibility exists that this algorithm could flag correct data as being in error. As such, this technique is not utilised in this research.

In the case of errors being detected in partition 2, there are essentially two alternative strategies that can be utilised. Either the data decoded so far for partition 2 can be retained (on the assumption that bitstream decoding errors are detected soon after they occur), or the entire second partition between successive synchronisation points can be thrown out. Both of these methods (labelled Mode 0 and Mode 1 respectively) are investigated in what follows.

In terms of PSNR, mode 0 performs worse when the error rate is high, but has the advantage of recovering more quickly as the error rate is reduced.

It was noted in the derivation of the partitioning location, that there would be some inefficiencies caused by the requirement to produce an exact partitioning ratio across the entire sequence, and by the ideal partitioning location being impossible for the data being partitioned. In either case, some of the last partition is unused. This 'Dead-Time' can be used to store a known, pseudo-random, bit-pattern which can be tested for once DCT coefficient decoding is complete. If errors are present in this code, then only data from partition 1 is kept. Obviously, the effectiveness of this code in detecting errors will depend on the length of the Dead-Time, but once this Dead-Time has been introduced by the partitioning process, it makes sense to find a use for it. This should remove those errors
where the decoding of partition 2 is corrupt, but sufficient EOB codes\textsuperscript{28} are located before the end of the space that is allocated.

When errors are detected in the re-combining process, or when there is Dead-Time at the end of the second partition, then the reconstructed video is at a slower rate than the data that is entering the re-combining process. However, the amount of data lost is calculable from the amount of data in partition 1, and this much data can be inserted back into the bitstream provided this is syntactically legal. The MPEG-II coding standard permits implicitly the inclusion of bits (zeros) before a Start Code, provided the Start Code remains byte-aligned. At worst, therefore, the output bitstream is delayed by 7 bits (which will fluctuate throughout the sequence, but never exceeds this limit). This is a fundamental result as, regardless of errors, the bitstream output from the system can be held at constant rate.

**Partitioning Algorithms**

The requirement that re-combining can be guaranteed to occur regardless of errors is only possible if all the partitions are synchronised at frequent intervals. There are essentially three different methods of achieving this, which are detailed below. In all three cases, however, it is only really logical to synchronise on a single group of interleaved macroblocks. This is because the interleaving process prioritises the data within a single group of macroblocks.

Perhaps the easiest approach is to encode the number of bits on partition 1 before partitioning occurs, as shown in figure 74. Inherent with any partitioning system that includes synchronisation, there will sometimes need to be some extra bits inserted to synchronise all the partitions. In this case, the ‘Dead-Time’ (DT) will at most be a single bit, except when the amount of variable length coded data is insufficient to fill the two partitions [i.e. the amount of header information is more than the DCT information].

\textsuperscript{28} It is important to note that the EOB code is likely to be artificially produced by errors, as it is one of the shortest codes in the codebook.
The length field needs to be able to encode all possible lengths for the interleaved partition 1. This is particularly expensive in terms of additions to the bitstream, as lengths in excess of 5kbits (from 10kbits of data) are quite realistic for I-frames of CCIR sequences coded at 4Mbits/s interleaved across the entire slice. As such, encoding of this parameter as a fixed length code will add at least 13 bits to every interleaved block. Re-synchronisation of the partitions will occur if the length code is decoded correctly, regardless of errors in the remainder of partition 1 or in the whole of the second partition. An associated problem with this coding scheme is that the splitting of a VLC code at the end of partition 1 could lead to start code emulation, and hence errors being flagged in the re-combining process. Positioning the length field before the macroblock headers allows this to be avoided, but only with careful design of the length codewords.

Alternative methods of encoding the length parameter more efficiently are possible. If partitioning is only permitted on a VLC codeword boundary (at the expense of increased Dead-Time), then the length can be VLC-encoded as the number of VLC codewords before partitioning occurs. One of the main disadvantages with such a scheme is that the efficiency of the VLC code would depend greatly on the properties of the sequence being coded (and on the coding rate). It is therefore difficult to design a codebook that is always efficient for realistic video sequences.
As an alternative to this, a new VLC code (a ‘partitioning code’, PC) can be introduced which marks the end of the first partition (see Figure 75). Unfortunately, there are no short VLC codewords available, so possible solutions are to:-

i) use an available, but much longer code

ii) modify the VLC codebook (though the probabilities associated with the generation of the MPEG-II code books are not available, and so entirely recreating the codebook is not possible)

In order to minimise the overheads associated with introducing an additional code\(^{29}\), the approach adopted in this research is to modify the End-Of-Block (EOB) code by adding a trailing ‘0’ to those EOB codes that occur in partition 1 (see table 12). The PC code is then defined to be the original EOB code (2 or 4 bits depending on how the blocks are coded) with a trailing ‘1’. Re-synchronisation of the partitions will occur if the partition code is decoded correctly, which only occurs if the whole of partition 1 is decoded correctly.

<table>
<thead>
<tr>
<th>Macroblock Grouping</th>
<th>EOB codes in partition 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>19.2%</td>
</tr>
<tr>
<td>2</td>
<td>22.0%</td>
</tr>
<tr>
<td>4</td>
<td>23.8%</td>
</tr>
<tr>
<td>11</td>
<td>25.2%</td>
</tr>
<tr>
<td>22</td>
<td>25.8%</td>
</tr>
<tr>
<td>44 (Slice)</td>
<td>25.9%</td>
</tr>
</tbody>
</table>

Table 12: Percentage of EOB codes occurring in partition 1, for frames 1-51 of Mobile & Calendar coded at 4Mbits/s

Table 12 shows that the proportion of EOB codes does not vary much as the number of macroblocks interleaved is varied\(^{30}\). The overheads associated with the addition of the trailing bit to the EOB code are therefore approximately constant for this sequence. From this table, it can be seen that for a 44 macroblock slice, with (say) 6 blocks coded for each

\(^{29}\) The use of a modified EOB code only minimises the overheads when this code occurs frequently in the bitstream, i.e. when the number of macroblocks interleaved in a single group is relatively small.

\(^{30}\) The proportion of EOB codes on partition 1 varies from sequence to sequence, but is typically in the region of 25%.
macroblock, the modifications to the EOB codes total about 66 bits. It would, therefore, have been more efficient to encode the length of partition 1. This is only true for macroblock groupings in excess of about 11, as the overheads associated with modifying the EOB codes are constant. This is a manifestation of having to modify the existing codebook without recreating it from scratch. If a spare VLC codeword is available, then the use of a Partition Code will be more efficient than encoding the length. The inclusion of the partitioning code, and modifications to the EOB codes, leads to about 70 bits being inserted into every slice of 44 macroblocks. In practice, however, many more bits than this are added, as partitioning of B-frame headers often leads to inefficiencies (due to there being very few DCT coefficients). This is considered later in this chapter (page 143).

In the special case that an entire slice is interleaved, a modification to the partition code scheme outlined above can be made (figure 76). As the subsequent syntax element in the MPEG-II bitstream is guaranteed to be a start code (which can be uniquely searched for within the bitstream), then:

i) the partition code itself is not required (and hence modifications to the EOB codes are not required). Partitioning occurs when the next partition 1 code is a start code.

ii) partitioning does not need to occur at a VLC codeword boundary as start code emulation cannot occur\(^{31}\). Provided there is more DCT information than header information, the only overheads are caused by the synchronisation of the partitions at the start of each slice (in a 2 partition system, this is an average of 0.5 bits per slice).

\[\text{Partition 1} \rightarrow \text{SC Header \ldots VLC}_1(B_1), \ldots, \text{VLC}_6(B_6), \text{VLC}_1(B_1) \text{[part]} \rightarrow \text{SC} \rightarrow \]

\[\text{Partition 2} \rightarrow \text{VLC}_1(B_1) \text{[part]} \ldots \text{VLC}_6(B_6), \ldots, \text{EOB}_1(B_1) \rightarrow \text{DT} \rightarrow \]

\[\text{Figure 76 : Slice partitioning of interleaved DCT coefficients}\]

\(^{31}\) A slice is *always* followed by a start code, and so no pattern of bits at the end of the slice can cause premature emulation of a start code (in the absence of errors).
The use of the start code as the synchronisation marker means that synchronisation of the bitstreams is guaranteed provided that the start code itself is not damaged.

Slice interleaving partially overcomes the inefficiency associated with partitioning B-frames, as it permits the partition code to be positioned earlier after the macroblock header (in the first DCT coefficient, modifications to the MPEG-II codebook are active prohibiting the insertion of a modified EOB code).

**Partitioning Performance**

Figures 77 and 78 show the effect of varying the number of macroblocks that are interleaved. For small macroblock groups, the probability of detecting errors is reduced, so some damage is not detected by the recombining process (and so PSNR is lower for bit-error rates in the range $1 \rightarrow 0.01$). For larger macroblock groups, the lower the error rate has to be for the PSNR to increase. As the coding rate is varied, so the curves would be expected to move in line with the change in MB group size. Mode 1 reconstruction (where the secondary partition is thrown away when errors are located), with large numbers of macroblocks interleaved, leads to almost constant PSNR when the error rate is high. It is thus possible to provide a form of guarantee of Service Quality.

![Figure 77: Average PSNR vs. Partition 2 bit-error rate [1:1 partitioning](Mobile & Calendar coded at 4Mbits/s, Mode 0 Reconstruction, various Macroblock Groupings)](image)
Figure 78: Average PSNR vs. Partition 2 bit-error rate [1:1 partitioning]
(Mobile & Calendar coded at 4Mbits/s, Mode 1 Reconstruction, various Macroblock Groupings)

Figure 79 (overleaf) shows the same information as Figure 78, but with a pseudo-random bit-pattern inserted into any Dead-Time introduced in partition 2 by the partitioning process. At low bit-error rates, there is virtually no change in PSNR. At high bit-error rates, the PSNR is held almost constant at a level almost equivalent (within 0.5dB) to always throwing out the secondary partition regardless of the number of macroblocks interleaved in a single group. In other words, this scheme is very efficient at detecting errors within the bitstream, regardless of the number of macroblocks interleaved between synchronisation points.
Figure 79: PSNR vs. BER with validation pattern in P2 Dead-Time. (Mobile & Calendar coded at 4Mbits/s, 1:1 partitioning, Mode 1 Reconstruction, various Macroblock Groupings)

Figure 80: PSNR vs. BER Comparison of Slice Interleaving and Macroblock Grouping of the entire slice. (Mobile & Calendar coded at 4Mbits/s, 1:1 partitioning)

Figure 80 shows that slice interleaving performs almost identically to interleaving in groups of 44 macroblocks (i.e. the entire slice). This is to be expected, as the efficiency gains of slice interleaving are brought about primarily through not having to modify the codebook, rather
than through a fundamental change in the actual algorithm. Although the MPEG-II coding standard permits the use of slices that do not span the width of the entire frame, the implementation used for this research does not allow this. It is, therefore, not possible to compare the effectiveness of slice interleaving with interleaving groups of macroblocks less than the entire slice.

The conclusions that can be drawn from these graphs are that:-

i) in general it is best to throw away partition 2 when errors are detected

ii) the use of a pseudo-random bit pattern in the inherent Dead-Time introduced by partitioning leads to an approximate guarantee of service quality, by increasing the likelihood of errors being detected.

iii) interleaving smaller numbers of macroblocks provides a faster recovery (as the bit-error rate is reduced) from errors as less data is lost when errors occur.

The following sequences (figures 81 to 86, summarised in Table 13 below) show the effect, in terms of damage, of errors on the second partition. For all the sequences, the macroblock group size is the entire slice. Decoded video is shown for each reconstruction mode with a partition 2 bit-error rate of 0.1%, and for mode 1 reconstruction at 0.01%32 and 1%. These show that the quality does indeed degrade gracefully (though there are some obviously visible artefacts), and that the need for a very low bit-error rate in the previous chapter was the protection of syntactic information. Each sequence is accompanied by some frames showing the difference33 between the decoded video with and without errors. These show quite clearly (for mode 1 reconstruction) which slices have the second partition retained, and which have it thrown out.

---

32 This is equivalent to an overall bit-error rate of 0.005% averaged over both partitions, comparable to the frames produced in the previous chapter.

33 As the difference between two greyscale images is in the range -255 to +255, in producing these error frames, this has been mapped into the range 0 to +255. +128 (grey) represents no difference in the images.
The frames showing mode 0 reconstruction show the effects caused by keeping all coefficients decoded before an error was actually detected. It is not overly obvious from still frames, but the noise in this form tends to differ significantly between successive frames, leading to poor quality video.
Figure 81: Effect of Partition 2 Errors on Partitioned MPEG-II Coded Bitstreams (frames and difference from error free sequence) (Mobile & Calendar coded at 4Mbits/s, P2 Error Rate = 0.1%, 1:1 Partitioning, Mode 0 Reconstruction)

Figure continued overleaf...
Figure 82: Effect of Partition 2 Errors on Partitioned MPEG-II Coded Bitstreams (frames and difference from original sequence) (Mobile & Calendar coded at 4Mbits/s, P2 Error Rate = 0.01%, 1:1 Partitioning, Mode 1 Reconstruction)

Figure continued overleaf...
Figure 83: Effect of Partition 2 Errors on Partitioned MPEG-II Coded Bitstreams (frames and difference from original sequence)
(Mobile & Calendar coded at 4Mbits/s, P2 Error Rate = 0.1%, 1:1 Partitioning, Mode 1 Reconstruction)

Figure continued overleaf...
Figure 84: Effect of Partition 2 Errors on Partitioned MPEG-II Coded Bitstreams
(Mobile & Calendar coded at 4Mbits/s, P2 Error Rate = 1%,
1:1 Partitioning, Mode 1 Reconstruction)
Figure 85: Effect of Partition 2 Errors on Partitioned MPEG-II Coded Bistreams (frames and difference from original sequence) (Mobile & Calendar coded at 4Mbits/s, P2 Error Rate = 0.1%, 1:1 Partitioning, Mode 1 Reconstruction with Dead-Time Validation)

Figure continued overleaf…
Figure 86: Effect of Partition 2 Errors on Partitioned MPEG-II Coded Bitstreams (frames and difference from original sequence)
(Mobile & Calendar coded at 4Mbits/s, P2 Error Rate = 0.1%, 1:1 Partitioning, Slice Interleaving)

Figure continued overleaf...
Partitioning Efficiency

Figure 87 shows the efficiency of partitioning at different macroblock groupings, contrasted with the efficiency of applying a BCH error correcting code to the second partition. Also shown is the efficiency of slice interleaving. The graph stops at a compression ratio of about 64 (equivalent to coding at 2.1Mbits/s) as the MPEG-II encoder fails to encode this sequence at rates lower than this, even if requested to do so as a formal parameter.

Although this graph shows a significant improvement in efficiency as the number of macroblocks interleaved is increased, one of the main contributions to this is the partition codes that are no longer required in the bitstream. When this is considered, the partitioning efficiency of interleaving single macroblocks and the entire slice differs by no more than 3%. This is, of course, irrelevant in terms of the partition code system, as these codes will always need to be inserted. It does, however, show that the efficiency would not be expected to change greatly for slice interleaving if shorter slices could be used.
Comparison of figures 87 and 88 shows that there is very little variation in the efficiency of partitioning as the sequence (and, in this example, also the image dimensions) is changed. In both cases, the efficiency varies almost linearly with the achieved compression ratio of the encoder.

**Transcoding Efficiency and Bit-rate Control**

The partitioning process developed above inherently adds data to the MPEG-II bitstream as part of the transcoding process. The fact that the quantity of data added is variable poses a significant problem in the partitioning process, as the output bit-rate for transmission must be constant. There are essentially two possible solutions to this:

i) transmit at a rate that guarantees to be sufficient for the transcoded data

ii) modify the coded video so as to control the bit-rate.

The first of these is inherently inefficient, and so is not considered further. Modification to the bit-rate of an MPEG-II coded bitstream is considered in [102] and [103]. By far the simplest solution proposed is the loss of high frequency DCT coefficients in order to reduce
the amount of data transmitted. In the case of interleaved DCT coefficients, this algorithm is particularly easy to realise by throwing away coefficients at the end of the interleaved block. This has the added advantage over the existing schemes of inherently throwing out coefficients from more active blocks first.

For large changes in bit-rate this technique leads to poor quality reconstructed video, but for small changes the results are acceptable. Re-quantization of the DCT coefficients leads to a small improvement in video quality [104] (in [102], 0.4dB for rate conversion of "Flower Garden" from 4Mbits/s down to 3.2Mbits/s) but at the cost of having to re-encode the VLC coefficients. The other techniques require much more complicated transcoding, which results in near-complete video decoders / encoders being required to achieve the transcoding. This defeats the objective of this research, so for the small changes in bit-rate required for this research (<10%) only removal of high frequency DCT coefficients is considered.

To achieve an output rate that is $\kappa$ times the input bit-rate, the amount of data to remove, $\varepsilon$, is given by:-

\[
\varepsilon = (1 - \kappa) \cdot \text{bits to partition} + \text{overheads}
\]

Obviously, $\varepsilon$ has to be adjusted so that only whole VLC codes are removed from the bitstream, else decoding would fail. In the implementation developed for the results presented in this thesis, $\varepsilon$ is also restricted so that:-

\[
\varepsilon \leq \frac{1}{2} \cdot \text{bits to partition}
\]

The purpose of this is to spread out the effect of large changes in coding rate across an entire, or possibly several, encoded frame(s). The remaining overheads in the partitioned data are given by:-

\[
\text{overheads} = \text{overheads} - \left\{(1 - \kappa) \cdot \text{bits to partition} + \varepsilon \right\}
\]

Figure 89 shows how $\varepsilon$ and 'overheads' vary throughout a sequence, as the output bit-rate is controlled to match the input bit-rate. The significant overheads accumulate during B-frames.
(on the x-axis, the labels mark the start of frames in the order they occur within the MPEG-II bitstream: I,P,B,B,P,B,P,B,B,P,B,I,B,B,...).

![Graph of bits to remove and overheads vs. Frame Number](image)

**Figure 89**: Fluctuation in $e$ and overheads vs. Frame Number  
(Mobile and Calendar, coded at 4Mbits/s, $\kappa = 1.00$, Slice Interleaving)

The overheads can be seen to be brought back down to zero during every I- and P-frame\textsuperscript{34}, since these frames contain significant amounts of DCT coefficients which can be discarded to control the bit-rate. Although there are significant overheads, it should be noted that 40kbits only corresponds to a system delay (at 4Mbits/s) of 10ms, or considerably less than the three frames ($3 \times 40$ms) delay inherent in the frame re-ordering of the MPEG-II encoder.

Figure 90 shows the effect of controlling the transmission bit-rate in the transcoder (and hence, outside the DPCM loop) for frames 1-51 of Mobile & Calendar, coded at 4Mbits/s. This graph shows the PSNR as the transmission rate is varied from 3.2Mbits/s upwards, until no coefficients are being thrown away in the partitioning process. Shown also is the PSNR for the same sequence as the coding rate is varied (so image information is being lost inside the DPCM loop). The two partitioning curves behave as expected: the only

\textsuperscript{34} There is a slight lack of synchronisation between the curves and the axis labels which increases throughout the sequence (due to some macroblocks being 'not-coded').
significant difference between the PSNR of slice interleaving and a macroblock grouping of 44 is caused by the difference in efficiencies of these two methods. This is to be expected, as the bit-rate controlling algorithms are the same. Keeping the bit-rate constant at 4Mbits/s produces a 0.6dB degradation in the video for slice interleaving.

![Diagram showing the effect of controlling the transmission bit-rate outside the DPCM loop](image)

Figure 90: Effect of controlling the transmission bit-rate outside the DPCM loop (frames 1-51 of Mobile and Calendar, coded at 4Mbits/s).

It should be noted that this algorithm is not perfect. There is a PSNR difference of about 0.25dB between when this algorithm is not used, and when the output bit-rate is set to be the same as the output rate for slice interleaving without the bit-rate being controlled (4.12Mbits/s). This is due to the large fluctuations that occur in the overheads through the sequence causing some coefficients to be lost (and to keep the rate constant, some zeros being added in when the bit-rate is too low).

**Partitioning B-frames to improve efficiency**

Table 14 shows that the majority of the overheads associated with partitioning the video data are indeed associated with B-frames, and so it is not always sufficient to partition the DCT coefficients. The B-frames contain very little (if any) DCT coefficient information, but contain significant amounts of data in the macroblock headers (motion vectors, etc.), and so there is insufficient information to efficiently partition the data. In such cases, to keep the
partitioning process efficient, it becomes necessary to divide data that is more susceptible to errors (in the knowledge that the errors do not propagate to other frames).

<table>
<thead>
<tr>
<th>MB Grouping</th>
<th>Average Overheads (bits)</th>
<th>Average Overheads (bits)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>per MB Group</td>
<td>per MB</td>
</tr>
<tr>
<td>1</td>
<td>23.5</td>
<td>23.5</td>
</tr>
<tr>
<td>2</td>
<td>39.6</td>
<td>19.8</td>
</tr>
<tr>
<td>4</td>
<td>68.7</td>
<td>17.2</td>
</tr>
<tr>
<td>11</td>
<td>162</td>
<td>14.7</td>
</tr>
<tr>
<td>22</td>
<td>301</td>
<td>13.7</td>
</tr>
<tr>
<td>44</td>
<td>573</td>
<td>13.0</td>
</tr>
</tbody>
</table>

Table 14: Average Partitioning Overheads, for frames 1-51 of Mobile & Calendar coded at 4Mbits/s

<table>
<thead>
<tr>
<th>Bitstream element</th>
<th>Code length</th>
<th>Average bits/macroblock</th>
<th>Affects subsequent decoding?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Macroblock increment</td>
<td>1-11</td>
<td>1.00</td>
<td>1.02</td>
</tr>
<tr>
<td>Macroblock type</td>
<td>1-9</td>
<td>1.42</td>
<td>2.87</td>
</tr>
<tr>
<td>Spatial temporal weight code</td>
<td>2</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Frame/Field motion type</td>
<td>2</td>
<td>0.00</td>
<td>1.95</td>
</tr>
<tr>
<td>DCT type</td>
<td>1</td>
<td>1.00</td>
<td>0.92</td>
</tr>
<tr>
<td>Quantiser scale code</td>
<td>5</td>
<td>2.10</td>
<td>2.07</td>
</tr>
<tr>
<td>Motion vertical field select 1,2</td>
<td>1,(1)</td>
<td>0.00</td>
<td>0.95</td>
</tr>
<tr>
<td>Motion horiz. / vert. Code</td>
<td>1-11,1-11</td>
<td>0.00</td>
<td>6.26</td>
</tr>
<tr>
<td>Motion horiz. / vert. r</td>
<td>1-8,1-8</td>
<td>0.00</td>
<td>2.42</td>
</tr>
<tr>
<td>Dmv horiz. / vert.</td>
<td>1-2,1-2</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Marker bit</td>
<td>1</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Coded Block Pattern</td>
<td>3-9</td>
<td>0.00</td>
<td>6.16</td>
</tr>
<tr>
<td>Total</td>
<td>5.52</td>
<td>24.62</td>
<td>27.74</td>
</tr>
</tbody>
</table>

Table 15: Distribution of bits in B-frame macroblock headers (Mobile & Calendar, frames 1-51, coded at 4Mbits/s)

Table 15 shows the distribution of bits within macroblock headers. Of the possible elements, only the motion vectors do not affect syntax of the subsequent decoding significantly. In terms of the amount of data, the motion vectors contribute on average 16.53 bits out of the average B-frame macroblock header length of 27.74 bits, which enables much more efficient partitioning of the header by placing these codes on partition 2. Obviously,
damage to the motion vectors will cause significant damage to the reconstructed video, but damage to B-frames will only persist until the next I- or P-frame.

Partitioning of B-frame headers is achieved by re-ordering the codes, so that the motion vector codes come last\(^{35}\), and inserting a code to indicate, for the entire macroblock group, the number of VLC codes before partitioning of the B-frame header occurs. The efficiency of this mode is shown in figure 93, and can be seen to be a significant improvement. However, damage to partition 2 causes significant damage to the video although it only persists for one or two frames.

![Figure 91: Average PSNR vs. Partition 2 bit-error rate (1:1 partitioning, Mode 0 reconstruction)](image)

Figures 91 (mode 0 reconstruction) and 92 (mode 1 reconstruction) show the effect in terms of PSNR as the partition 2 bit-error rate is varied. At high error rates, there is a significant degradation (up to 4dB at a bit-error rate of 100%) compared to when the B-frame headers are not split (figures 77 and 78). This is to be expected as damage to the motion vectors of the B-frames will cause significant damage to these frames (as can be seen from figure 94).

\(^{35}\) The repositioning of the motion vector codes cannot produce start code emulation problems, as the motion vectors could be the last item in the macroblock header.
Figure 92: Average PSNR vs. Partition 2 bit-error rate [1:1 partitioning, Mode 1 reconstruction] (frames 1-51 of Mobile and Calendar coded at 4Mbits/s, B-frame headers partitioned)

Figure 93: Efficiency of Partitioning vs. Compression Ratio with B-frame headers partitioned (frames 1-51 of Mobile and Calendar, coded at 4Mbits/s)
Figure 94: Effect of Partition 2 Errors on Partitioned MPEG-II Coded Bitstreams (frames and difference from original sequence) 
(Mobile & Calendar coded at 4Mbits/s, P2 Error Rate = 0.1%, 1:1 Partitioning, B-frame headers partitioned)

Figure continued overleaf...
Figure 95 shows that the overheads are at least an order of magnitude lower than without partitioning of the B-frame headers (figure 89, page 142).

![Diagram of overheads vs. Frame Number](image)

**Figure 95**: Fluctuation in ε and overheads vs. Frame Number
(Mobile and Calendar, coded at 4Mbits/s, $\kappa = 1.00$, B-frame headers split, 44 macroblocks interleaved)

**Summary**

This chapter has outlined techniques that can be used to reduce the amount of error correction required for transmission of coded video. The use of data interleaving within the syntax of the coded video, and partitioning of the video data into different levels of visual importance, leads to a graceful reduction in decoded image quality as the transmission error rate increases. All methods of partitioning the video data inherently introduce additional data into the bitstream, and so an algorithm for controlling the bit-rate after partitioning has been outlined. Provided the change in bit-rate is small, damage to the video data is also small, and the system delay associated with controlling the bit-rate is small compared to the delays in the MPEG-II encoder.

The constraint of working within the codebooks of the MPEG-II coding standard leads to a reduction in efficiency of the partitioning process, except when the entire slice is interleaved and then partitioned. This has the disadvantage of spreading the effect of errors
over the entire width of the image with the encoder used in this research. Since the probability of errors occurring within a given section of the bitstream increases with the length of the section, shorter interleaved blocks will lead to better quality video.

Therefore, in order to achieve the required efficiency, there are two choices. Either modify the existing codebook, to make a partitioning code available; or reduce the length of slices in the original MPEG-II bitstream. The latter is provided for in the coding standard, but is not implemented in the version of the encoder used during this research.

With the algorithms outlined in this chapter, the existing video bitstream is subdivided into two (or more) layers with different error resilience requirements. This is then ideal for the transmission links outlined in Chapter Three where the properties of the modulation scheme are such that different bits have different error resilience. In the next chapter, these are combined (with appropriate error correction applied to the important syntax layer) to produce a video transmission link that is capable of withstanding the bursts of noise typical in the multipath fading environment of mobile radio.
Chapter Six: MPEGII Transmission System
Chapter Six : MPEGII Transmission System

Having outlined techniques to improve the error resilience of the video bitstream, and for separating the bitstream into different priority partitions, it is now possible to define an overall end-to-end system for transmission. The complete system contains numerous parameters that can be modified independently to vary the system performance, and it would be prohibitively time consuming to present results as all of these are varied. Consequently, where the results from the previous chapters have shown particular values to be suitable, these values are assumed throughout the results of this chapter.

Overall System Model

Figure 96 shows the overall transmission model. The original MPEG-II bitstream is DCT coefficient interleaved, and then partitioned to produce two sub-channels of information, with different error-resilience requirements. Partition 1 contains the more important data (syntax, motion vectors, and initial DCT coefficients) which is strongly error protected and then interleaved to distribute the transmission errors that will occur in deep fades in a more random fashion. Partition 2 contains the high frequency coefficients which are less important, and so are optionally protected from occasional random errors by a weak error correcting code. These two sub-channels are then modulated using QAM16 Type III.
before filtering and transmission through the fading channel. The receiver tracks the fading effect of the multipath channel in order to be able to demodulate the data, and then the output from the demodulator is de-interleaved and error-corrected before being recombined and rearranged into an MPEG-II conformant bitstream.

**Choice of System Parameters**

There are a large number of parameters to be chosen for the above model, however simulation results presented earlier can be used to reduce the number that need be investigated in the context of the overall system. For example, the error correcting codes and interleaving can be chosen from simulations of the transmission system (presented in Chapter Three), and a knowledge of the permitted error-rates on the two channels (see Chapters Four and Five). In all the simulation results that follow, the following parameters have been used:

\[ \alpha = 0.35 \]
\[ f_u T_s = 0.001 \]
\[ \text{CNR} > 15 \text{dB} \]

These represent typical values for existing mobile radio transmission systems, although the value of \( f_u T_s \) is towards the top of the expected range. The primary reason for choosing this value is that smaller values of \( f_u T_s \) require longer simulations to be run to achieve a reasonable statistical averaging.

**Fade-Tracking Algorithms**

Chapter Three outlined various techniques for tracking the fading effect of the channel in order to be able to demodulate QAM16 Type III. The combination of second order spatial diversity at the receiver, coupled with pilot-assisted automatic gain control produces the lowest bit-error rate (at the expense of having to transmit the pilot symbols) of the techniques outlined in this thesis. Although the effectiveness of the gain control improves as the pilot spacing is reduced, the overall effect on the bit-error rate is small at CNR up to about 30dB. Consequently a pilot spacing of 20 symbols is chosen, which represents a compromise between efficiency and residual bit-error rate.
Interleaving period

It was noted in Chapter Three that the errors were concentrated in bursts, and that the error correcting codes could be made more effective by interleaving them. The theoretical expressions that were derived assumed that the errors were randomly distributed, and this represents a bound on what can be achieved by the error correcting codes in the overall system model. The actual number of code blocks that need to be interleaved will depend on the duration of the fades, and consequently on the value of $f_d T$: the slower the fading, the longer the interleaving period should be to produce a good approximation to randomly distributed errors. Figure 97 shows the effect on bit-error rate of different interleaving periods for various BCH codes applied to partition 1, for $f_d T = 0.001$ at CNR=20dB.

![Figure 97: Effect on Bit-error rate of BCH code interleaving period](image)

Despite there being quite large fluctuations in the bit-error rates, due to statistical fluctuations in the pre-error-correction bit-error rate, there is no significant improvement above an interleaving period of 6kbits. This represents a delay of 6ms for a transmission rate of 1Msymbols/sec which is significantly less than the delays inherent in the rest of the system.

Error Correcting Codes

Figure 98 shows the effect of adding a BCH code to partition 1. In producing this graph, 96 code blocks (about 6kbits) have been interleaved in order to approach a random distribution of errors. Although there are large fluctuations (caused by 'only' running the simulation with about $10^7$ symbols), it is possible to see the general trends in this graph.
Chapter Five showed that reliable transmission of partition 1 required a bit-error rate of the order of 0.005%, which at 20dB requires at least a BCH(63,39,4) code.

Figure 98: Effect on Partition 1 Bit-error rate of various error correcting codes (with interleaving).

Figure 99 shows the effect of error correcting codes on partition 2, but without interleaving. As expected, the effect on the bit-error rate is small, even when a strong code is used. Despite not being able to correct the majority of the errors in the bitstream, it may be advantageous to correct a small number of occasional, random errors by using a weak error-correcting code.

Figure 99: Effect on Partition 2 Bit-error rate of various error correcting codes (without interleaving).
Partitioning Properties

Chapter Five introduced the partitioning algorithm, working on groups of interleaved macroblocks. The greatest efficiency was obtained when an entire slice was interleaved, though it was noted that similar efficiency should be obtained if the original MPEG-II encoder could produce shorter slices. In order to investigate what effect this would have in the end-to-end system, simulation results are also presented for macroblock groupings of 11 (quarter slice) and 44 (entire slice).

The partitioning process is not completely efficient, and so the bit-rate control algorithms outlined in Chapter Five are included to fix the post-partitioning bit-rate.

Efficiency

The combination of partitioning, error correction and pilot assisted fade tracking leads to a reduction in the actual amount of video data that can be transmitted. Equation 20 (Chapter Three, page 73) combines these with the increased bandwidth efficiency of QAM16, and the bandwidth of the filters, to produce an overall efficiency in terms of video bits / Hz. Table 16 shows how this efficiency varies for different partition 1 error correcting codes, and also the effect of a weak error correcting code on partition 2.

<table>
<thead>
<tr>
<th>Modulation</th>
<th>Levels</th>
<th>α</th>
<th>Pilot Spacing</th>
<th>P1 BCH n</th>
<th>k</th>
<th>t</th>
<th>n</th>
<th>P2 BCH k</th>
<th>t</th>
<th>Video bits/Hz</th>
</tr>
</thead>
<tbody>
<tr>
<td>QAM16</td>
<td>16</td>
<td>0.35</td>
<td>20</td>
<td>63</td>
<td>51</td>
<td>2</td>
<td></td>
<td>63</td>
<td>63</td>
<td>0</td>
</tr>
<tr>
<td>QAM16</td>
<td>16</td>
<td>0.35</td>
<td>20</td>
<td>63</td>
<td>45</td>
<td>3</td>
<td></td>
<td>63</td>
<td>63</td>
<td>0</td>
</tr>
<tr>
<td>QAM16</td>
<td>16</td>
<td>0.35</td>
<td>20</td>
<td>63</td>
<td>39</td>
<td>4</td>
<td></td>
<td>63</td>
<td>63</td>
<td>0</td>
</tr>
<tr>
<td>QAM16</td>
<td>16</td>
<td>0.35</td>
<td>20</td>
<td>63</td>
<td>36</td>
<td>5</td>
<td></td>
<td>63</td>
<td>63</td>
<td>0</td>
</tr>
<tr>
<td>QAM16</td>
<td>16</td>
<td>0.35</td>
<td>20</td>
<td>63</td>
<td>30</td>
<td>6</td>
<td></td>
<td>63</td>
<td>63</td>
<td>0</td>
</tr>
<tr>
<td>QAM16</td>
<td>16</td>
<td>0.35</td>
<td>20</td>
<td>63</td>
<td>39</td>
<td>4</td>
<td></td>
<td>63</td>
<td>57</td>
<td>1</td>
</tr>
<tr>
<td>QAM16</td>
<td>16</td>
<td>0.35</td>
<td>20</td>
<td>63</td>
<td>39</td>
<td>4</td>
<td></td>
<td>63</td>
<td>51</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 16: Bandwidth efficiency for various systems
System Results

Figures 100 and 101 show the quality of the received video in terms of the PSNR (with respect to the original unencoded frames) as the channel quality is varied\textsuperscript{36}. The output bit-rate requested of the MPEG-II encoder is about 4Mbits/s. After partitioning (controlled with $\kappa=1$, i.e. 100% efficiency) and error correction, this leads to a transmission rate of 1.33Msymbols/second.

![Figure 100: PSNR vs. Channel quality for frames 1-50 of Mobile & Calendar, BCH(63,39,4) on P1, BCH(63,57,1) on P2, slice interleaved, $\kappa=1$, transmitted at 1.33Msymbols/sec.](image)

Above 20dB, there is a gradual improvement in the quality of the decoded video as the number of errors on partition 2 is reduced. The error correcting codes on partition 1 have successfully corrected the majority of the partition 1 errors above this point. Below 20dB, the error correcting codes are unable to correct sufficient errors, and so damage to the syntax of the bitstream results. This has a potentially large effect on the quality of the decoded video, although this depends on exactly which bits are damaged. Consequently there are large fluctuations in the PSNR\textsuperscript{37}.

\textsuperscript{36} All MPEG-II bitstreams created in the production of the graphs in this chapter are included on a CD-ROM in the back of this thesis. Appendix D details the contents of this disk.

\textsuperscript{37} These results are produced from a single transmission of the video sequence, and the exact effect on the decoded video depends on exactly which syntax bits are damaged by noise. Re-running the simulation would produce decoded video with a different PSNR.
Figure 101: PSNR vs. Channel quality for frames 1-50 of Football, BCH(63,39,4) on P1, BCH(63,57,1) on P2, slice interleaved, κ=1, transmitted at 1.33M symbols/sec.
Figure 102: End-to-end system results
(Mobile & Calendar transmitted at 1.33Mbits/s, γ=20dB, P1 BCH(63,39,4), P2 BCH(63,57,1), slice interleaving with mode 1 reconstruction)
Figure 103: End-to-end system results
(Mobile & Calendar transmitted at 1.33Mbits/s, $\gamma=24$dB, P1 BCH(63,39,4), P2 BCH(63,57,1), slice interleaving with mode 1 reconstruction)
Figures 102 and 103 show example decoded frames for the sequence Mobile & Calendar, at 20dB and 24dB respectively. In both cases, the decoded video is intelligible though obvious artefacts remain. Figure 102 shows that some residual partition 1 damage has remained after error correction.

This inability to correct the errors on partition 1 is due to the choice of a BCH(63,39,4) error correcting code for that partition. Figure 98 (page 156) shows that this provides the required bit-error rate only when the channel quality is above about 20dB. Figure 104 shows the effect of using a stronger code, whilst retaining the same transmission rate. The quality of the decoded video in the absence of errors is decreased, as the extra error-correction information has required a decrease in the original coding rate (to maintain the same transmission rate). There is a slight improvement when the channel quality is worse than 20dB.

![PSNR vs. Channel quality for frames 1-50 of Mobile & Calendar, BCH(63,30,6) on P1, BCH(63,57,1) on P2, slice interleaved, \( \kappa = 1 \), transmitted at 1.33M symbols/sec.](image)

Figure 104: PSNR vs. Channel quality for frames 1-50 of Mobile & Calendar, BCH(63,30,6) on P1, BCH(63,57,1) on P2, slice interleaved, \( \kappa = 1 \), transmitted at 1.33M symbols/sec.

Figure 105 shows the effect of removing the weak error-correcting code from partition 2 (under the same conditions as figure 100, page 158). The quality of the decoded video recovers much more slowly as the channel quality improves, as even single bit errors are sufficient to damage all subsequent partition 2 data for a slice.
Figure 105: PSNR vs. Channel quality for frames 1-50 of Mobile & Calendar, BCH(63,39,4) on P1, no P2 error correction, slice interleaved, \( k=1 \), transmitted at 1.33Msymbols/sec.

Figures 106 and 107 show the effect of not using slice interleaving. First, figure 106 shows the result of interleaving 44 macroblocks (i.e. the entire slice) but using the partition code to mark the end of partition 1. This is purely a benchmark (as the efficiencies are more closely matched) for comparison with figure 107, which is for 11 macroblocks interleaved.

Figure 106: PSNR vs. Channel quality for frames 1-50 of Mobile & Calendar, BCH(63,39,4) on P1, BCH(63,57,1) on P2, 44 macroblocks interleaved, \( k=1.05 \), transmitted at 1.33Msymbols/sec.

Use of the inherently less efficient partition code has lead to a reduction in the PSNR obtained for the entire system as expected. Interleaving 44 macroblocks provides otherwise similar results to slice interleaving. Interleaving 11 macroblocks provides a more rapid
improvement in PSNR as the channel quality improves from 20 to 25dB, which is to be expected as an error (or a short burst of errors) only affects 1/4 of the width of the image. For both these figures, $\kappa$ has been set to 1.05 to combat (to some extent) the inefficiencies inherent in partitioning.

![Graph showing PSNR vs. CNR for different mobiles and codes.]

Figure 107: PSNR vs. Channel quality for frames 1-50 of Mobile & Calendar, BCH(63,39,4) on P1, BCH(63,57,1) on P2, 11 macroblocks interleaved, $\kappa=1.05$, transmitted at 1.33Msymbols/sec.

Summary

This chapter has presented results from combining all the algorithms outlined in this thesis into a single end-to-end system for transmitting MPEG-II compressed video. The use of DCT coefficient interleaving and then partitioning allows one of the sub-channels to be transmitted with little or no error correction, thus improving efficiency. Further, the fact that the errors tend to occur in bursts means that damage to the video caused by errors in this sub-channel is spatially localised.

To reduce the error rate on the first partition to a level that produces satisfactory decoded video, a strong error correcting code coupled with interleaving is employed. In conjunction with second order spatial diversity at the receiver, and pilot-assisted automatic gain control for fade tracking, the bit-error rate on this partition has been kept below the required 0.005% for CNR > ~20dB. To reduce the effect of the partition 1 errors occurring in
bursts, the interleaving period should be as long as possible, limited only by the delay inserted into the system, and the memory in the receiver.

The use of shorter blocks of interleaved data provides a more rapid improvement in decoded image quality as the channel quality improves. This is further evidence to back up the discussion in Chapter Five, which showed that the number of macroblocks interleaved should be as low as possible, within the limits of partitioning efficiency.
Chapter Seven: Conclusions
Chapter Seven: Conclusions

This thesis has investigated the particular problems associated with the transmission of highly compressed digital video bitstreams through noisy communication links. Rather than creating a new video coding algorithm that performs well in the presence of noise, this thesis has developed the concept of a transcoder operating on the output of an existing coding algorithm. This has several key advantages:-

i) the cost of developing entirely new video coding algorithms is much higher than that of developing a transcoder (which is essentially a partial decoder / encoder).

ii) the existing video coding standards are well established, and are beginning to appear as single chip-set VLSI implementations. As a result, consumer devices will soon start appearing which use these devices, and compatibility is an important aspect in any new system.

iii) the transcoder works on the compressed video, and so the bandwidth required for the communication link between the video source and the transmitter can be kept low.

Starting with a discussion of how digital signals are compressed and in particular how video frames can be compressed, Chapter One outlines how existing video coding algorithms work. Emphasis is placed on MPEG-II as this is a generic coding algorithm that targets many diverse applications. The other popular video coding standards (MPEG-I, H.261, H.263) are intrinsically similar to MPEG-II, and so techniques developed to work with MPEG-II should be extensible to these other algorithms.

Chapter Two outlines existing approaches to the problem of sending compressed video through noisy communication links. The majority of these require feedback to the transmitter which is disadvantageous because:-

i) the communication is point-to-point. Separate transmission is required to each receiver, and so broadcast applications are not possible.
ii) the video server at the transmitter has a lot of work to do for each communication link that is open.

The alternative approach is to use error correcting codes to correct all the errors incurred by transmission, but this is inherently inefficient when the channel quality is likely to fluctuate.

This thesis has been primarily concerned with the fast fading environment typical of wireless transmission to a mobile receiver. The lack of a direct line-of-sight between transmitter and receiver results in severe fades in the received signal level (as the receiver moves through the field of reflected signals), which in turn makes transmission particularly susceptible to noise. In addition, the reflected signals may have travelled different distances to reach the receiver, and so interference between successive symbols becomes a possibility. This interference is more significant if the signalling rate is very high, becoming significant in the region of about 2 million symbols/second. In order to increase the bit-rate, modulation schemes that encode several bits into each symbol are considered, but these are inherently more susceptible to errors. Chapter Three outlines all the stages (modulation, filters, fade tracking, and error correcting codes) required to provide an end-to-end simulation of transmission, and to achieve reliable error performance.

Having developed the transmission simulation, the concept of the video transcoder is introduced in Chapter Four in order to improve the error resilience of an MPEG-II coded video bitstream. The importance of decoder synchronisation has been investigated, and as a result the concept of DCT coefficient interleaving has been introduced. Without incurring any additional overheads, DCT coefficient interleaving significantly improves decoder performance when the error rate is relatively low (and so some coefficients for the entire image are decoded correctly).

Chapter Five utilises the error resilience properties of DCT coefficient interleaving in conjunction with the fact that the bitstream is more severely affected by errors towards the end of an interleaved block than towards the beginning, to divide the bitstream into an arbitrary number of partitions. The result of this is to produce a base partition containing syntax elements and low frequency coefficients, and remaining partitions comprising of the higher frequency DCT coefficients. The partitioned bitstream is suitable for transmission via the transmission channel developed in Chapter Three which inherently produced sub-channels
with different error resilience. Different approaches to handling errors that occur in the partitions are investigated.

The partitioning process is not 100% efficient, primarily because an additional codeword is required to mark the end of the first partition. Alternative approaches to marking the end of this partition are investigated, although the MPEG-II encoder used throughout this research is unable to modify the length of a slice which ought to lead to a more optimal solution (this feature is part of the MPEG-II coding specification). In any case, some small inefficiencies in the partitioning process will remain. To retain compatibility with a constant rate input bitstream, algorithms for controlling the transmitted bit-rate are considered, although these inherently reduce the quality of the reconstructed video in the absence of noise. However, the transcoder cannot provide feedback to the original encoder (as this would require modifications to the original encoder, and would create problems if two different transcoders operated on the bitstream) and so the output bit-rate must be constant.

Finally, Chapter Six presents results for an end-to-end system comprising all the techniques developed in this thesis. Error correcting codes are included to control the bit-error rate for the primary channel to an effectively error-free rate (above a threshold in channel quality). This guarantees a basic quality of service, which is enhanced by the second partition when the multipath fading is not severe. Consequently the goal of this thesis is achieved: to produce a compressed video bitstream transmission system that degrades gracefully as the channel quality worsens, whilst retaining compatibility with existing compression standards. There is a slight degradation in image quality in the absence of errors due to the inherent inefficiency in the partitioning process, and the requirement of the transmission rate being constant.

In addition, this thesis has shown that video coding schemes cannot solely concentrate on compression. Achieving high compression ratios is irrelevant if transmission requires significant error correction to be applied to the transmitted data in order to produce intelligible video. This thesis has shown that the existing MPEG-II coding standard can be improved on significantly, without affecting the compression efficiency.
Further Work

As mentioned above, the MPEG-II encoder used for this research is unable to divide a line of video into several separate slices. Doing so would be likely to retain the efficiency properties associated with interleaving all the DCT coefficients of a slice, but would increase the number of synchronisation points in the bitstream. As an alternative, the efficiency of partitioning can be improved by re-creating the codebook for the variable length coding of the DCT coefficients. The inclusion of a code to mark the end of partition without adversely affecting the statistical properties of the existing codebook should make a significant reduction in overheads (many of which are caused by the change to the EOB code when it occurs before partitioning has occurred).

The most significant source of video distortion remains damage to the higher order elements of the MPEG-II bitstream. Of particular significance are motion vectors and differentially coded parameters (especially the DC level) that are reset at the beginning of a slice. Alternative methods of coding this data exist in the literature, which have the advantage of removing small numbers of errors quite rapidly. Finally, the MPEG-II syntax includes the possibility of concealment motion vectors which can be used to reduce the visibility of errors that are detected in the bitstream. The inclusion of these techniques in the original MPEG-II bitstream, and appropriate utilisation in the partition recombining process, should lead to significant improvement in the quality of received video. Further, algorithms for increasing the resolution of images ("super-resolution" algorithms) [for example 10, pp. 331-345] could be employed to reduce the blurring effect caused when high frequency DCT coefficients are discarded due to errors. However, for P- and B-frames, the workload of the decoder would increase significantly if these were used (as these algorithms require the whole image, not the motion prediction error signal, to work).

Future video encoders are likely to be based on wavelet compression [105, 106] rather than the use of DCT based algorithms. It has not been within the scope of this thesis to investigate how a wavelet video encoder could benefit from coefficient interleaving and partitioning, and whether in fact use of these techniques would make sense.
Appendix A: MPEGII Syntax elements
Appendix A: MPEGII Syntax elements

This appendix outlines (as pseudo-code) some of the major syntax elements of the MPEG-II bitstream. It is not intended to be a definitive description of the MPEG-II bitstream (see [42] for example), rather to outline the structures that are referred to earlier in the thesis.

### pseudo-code

```plaintext
video_sequence()
{
    next_start_code()
    sequence_header()
    if (nextbits() == extension_start_code)
        sequence_extension()
        do
            extension_and_user_data(0)
            do
                if (nextbits() == group_start_code)
                    group_of_pictures_header()
                    extension_and_user_data(1)
                }
            picture_header()
            picture_coding_extension()
            extensions_and_user_data(2)
            picture_data()
        } while ((nextbits() == picture_start_code) ||
            (nextbits() == group_start_code))
    if (nextbits() != sequence_end_code)
        sequence_header()
        sequence_extension()
    } while (nextbits != sequence_end_code)
}
else
{
    /* MPEG-I decoder */
}
sequence_end_code

<table>
<thead>
<tr>
<th>No. of bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
</tr>
</tbody>
</table>

Table 17: MPEG-II bitstream structure
Sequence_header()
{
    sequence_header_code 32
    horizontal_size_value 12
    vertical_size_value 12
    aspect_ratio_information 4
    frame_rate_code 4
    bit_rate_value 18
    marker_bit '1'
    vbv_buffer_size_value 10
    constrained_parameters_flag 1

    load_intra_quantiser_matrix 1
    if ( load_intra_quantiser_matrix )
    {
        intra_quantiser_matrix[64] 8x64
    }

    load_non_intra_quantiser_matrix 1
    if ( load_non_intra_quantiser_matrix )
    {
        non_intra_quantiser_matrix[64] 8x64
    }

    next_start_code()
    if ( nextbits() != extension_start_code )
    {
        if ( nextbits() == user_data_start_code )
        {
            user_data()
        }
    }
}

Table 18: Sequence header fields

group_of_pictures_header()
{
    group_start_code 32
    time_code 25
    closed_gop 1
    broken_link 1

    next_start_code()
}

Table 19: Group of Pictures data fields
picture_header()
{
    picture_start_code 32
    temporal_reference 10
    picture_coding_type 3
    vbv_delay 16

    if ( picture_coding_type == 'P' || picture_coding_type == 'B' )
    {
        full_pel_forward_vector 1
        forward_fcode 3
    }

    if ( picture_coding_type == 'B' )
    {
        full_pel_backward_vector 1
        backward_fcode 3
    }

    while ( nextbits() == '1' )
    {
        extra_bit_picture '1' 1
        extra_picture_information 8
    }

    extra_bit_picture '0' 1

    next_start_code()
}

Table 20: Picture header fields

picture_data()
{
    do
    {
        slice()
    }
    while ( nextbits() == slice_start_code )

    next_start_code()
}

Table 21: Picture data fields
slice()
{
    slice_start_code()
    if ( vertical_size > 2800 )
        slice_vertical_position_extension

    if ( sequence_scalable_extension is present in the bitstream )
        if ( scalable_mode == 'data partitioning' )
            priority_breakpoint

    quantiser_scale_code
    if ( nextbits() == '1' )
    {
        marker_bit
        intra_slice
        reserved_bits

        while ( nextbits() == '1' )
        {
            extra_bit_slice
            extra_information_slice
        }
    }

    extra_bit_slice
    do
    {
        macroblock()
    }

    while ( nextbits() != '000 0000 0000 0000 0000 0000 0000' )

    next_start_code()
}

Table 22 : Slice fields

macroblock()
{
    while ( nextbits() == '0000 0001 0001' )
        macroblock_escape

    macroblock_address_increment
    macroblock_modes()

    if ( macroblock_quant )
        quantiser_scale_code

    if ( macroblock_motion_forward ||
        ( macroblock_intra && concealment_motion_vectors ) )
        motion_vectors(0)

    if ( macroblock_motion_backward )
        motion_vectors(1)

    if ( macroblock_intra && concealment_motion_vectors )
        marker_bit

    if ( macroblock_pattern )
        coded_block_pattern()

    for ( i = 0; i < block_count; i++ )
    {
        block(i)
    }
}

Table 23 : Macroblock fields

No. of bits

<table>
<thead>
<tr>
<th>Field</th>
<th>No. of bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>slice_start_code</td>
<td>32</td>
</tr>
<tr>
<td>slice_vertical_position_extension</td>
<td>3</td>
</tr>
<tr>
<td>sequence_scalable_extension is present in the bitstream</td>
<td>7</td>
</tr>
<tr>
<td>scalable_mode == 'data partitioning'</td>
<td>7</td>
</tr>
<tr>
<td>quantiser_scale_code</td>
<td>5</td>
</tr>
<tr>
<td>marker_bit</td>
<td>1</td>
</tr>
<tr>
<td>intra_slice</td>
<td>1</td>
</tr>
<tr>
<td>reserved_bits</td>
<td>7</td>
</tr>
<tr>
<td>extra_bit_slice</td>
<td>1</td>
</tr>
<tr>
<td>extra_information_slice</td>
<td>8</td>
</tr>
<tr>
<td>macroblock_escape</td>
<td>11</td>
</tr>
<tr>
<td>macroblock_address_increment</td>
<td>1-11</td>
</tr>
<tr>
<td>macroblock_modes()</td>
<td></td>
</tr>
<tr>
<td>macroblock_quant</td>
<td></td>
</tr>
<tr>
<td>quantiser_scale_code</td>
<td>5</td>
</tr>
<tr>
<td>macroblock_motion_forward</td>
<td></td>
</tr>
<tr>
<td>( macroblock_intra &amp;&amp; concealment_motion_vectors )</td>
<td>...</td>
</tr>
<tr>
<td>motion_vectors(0)</td>
<td></td>
</tr>
<tr>
<td>macroblock_motion_backward</td>
<td></td>
</tr>
<tr>
<td>motion_vectors(1)</td>
<td></td>
</tr>
<tr>
<td>macroblock_intra &amp;&amp; concealment_motion_vectors</td>
<td>...</td>
</tr>
<tr>
<td>marker_bit</td>
<td>1</td>
</tr>
<tr>
<td>macroblock_pattern</td>
<td></td>
</tr>
<tr>
<td>coded_block_pattern()</td>
<td></td>
</tr>
<tr>
<td>for ( i = 0; i &lt; block_count; i++ )</td>
<td>...</td>
</tr>
<tr>
<td>block(i)</td>
<td></td>
</tr>
</tbody>
</table>
```
 motion_vectors(s)
 { 
     if ( motion_vector_count == 1 )
     {
         if ( ( mv_frame == 'field' ) && ( dmv != 1 ) )
             motion_vertical_field_select
             motion_vector(s)
     }
     else
     {
         if ( dmv != 1 )
             motion_vertical_field(s)
             motion_vector(s)
     }
 }

 motion_vector(s)
 { 
     motion_horizontal_code(s)
     if ( ( horizontal_f != 1 ) && ( motion_horizontal_code != 0 ) )
         motion_horizontal_r(s)
     if ( dmv == 1 )
         dmv_horizontal(s)
     motion_vertical_code(s)
     if ( ( vertical_f != 1 ) && ( motion_vertical_code != 0 ) )
         motion_vertical_r(s)
     if ( dmv == 1 )
         dmv_vertical(s)
 }

 Table 24 : Motion-vector fields

<table>
<thead>
<tr>
<th>No. of bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-11</td>
</tr>
<tr>
<td>1-8</td>
</tr>
<tr>
<td>1-2</td>
</tr>
<tr>
<td>1-11</td>
</tr>
<tr>
<td>1-8</td>
</tr>
<tr>
<td>1-2</td>
</tr>
</tbody>
</table>

 block(i)
 { 
     if ( pattern_code[i] ) /* Is this block coded? */
     { 
         if ( macroblock_intra )
         { 
             if ( i < 4 ) /* Luminance block */
             { 
                 dct_dc_size_luminance
                 if ( dct_dc_size_luminance != 0 )
                     dct_dc_differential
             }
             else /* Chrominance block */
             { 
                 dct_dc_size_chrominance
                 if ( dct_dc_size_chrominance != 0 )
                     dct_dc_differential
             }
         }
         else 
         { 
             First DC coefficient (modified codebook) 
         } 
     }
     while ( nextbits() != End_Of_Block )
     { 
         Subsequent DCT coefficients
     }
     End_Of_Block
 }

 Table 25 : Block fields

<table>
<thead>
<tr>
<th>No. of bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-9</td>
</tr>
<tr>
<td>1-11</td>
</tr>
<tr>
<td>1-11</td>
</tr>
<tr>
<td>1-11</td>
</tr>
<tr>
<td>2-9</td>
</tr>
<tr>
<td>1-4</td>
</tr>
</tbody>
</table>

 176
Appendix B : Mathematical Derivations
Appendix B : Mathematical Derivations

This appendix presents some of the mathematical derivations that are used within the body of this thesis, but would have been a distraction within if presented in situ as they are either too complex or too long.

Integral for Fading Channel $P(e)$

Proof of :-
\[
I = \int_{0}^{\infty} e^{-\gamma/\gamma_0} \cdot \frac{1}{2} \cdot \text{erfc} \left( \sqrt{\gamma} \right) d\gamma = \frac{1}{2} \left( 1 - \frac{1}{\sqrt{1 + \frac{1}{\gamma_0}}} \right)
\]

Integrating by parts gives
\[
I = \left[ \frac{1}{2} \text{erfc} \left( \sqrt{\gamma} \right) - e^{-\gamma/\gamma_0} \right]_0^\infty - \int_0^\infty -e^{-\gamma/\gamma_0} \cdot \frac{1}{2} \cdot \frac{d}{d\gamma} \left( \text{erfc} \left( \sqrt{\gamma} \right) \right) d\gamma
\]

where:-
\[
\text{erfc}(x) = 1 - \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt
\]

so,
\[
\frac{d}{dx} \left( \text{erfc}(x) \right) = \frac{2}{\sqrt{\pi}} e^{-x^2}
\]

From this, I can be rewritten :-
\[
I = \frac{1}{2} + \int_0^\infty e^{-\gamma/\gamma_0} \cdot \frac{1}{2} \cdot \frac{1}{\sqrt{\gamma}} \cdot \frac{1}{\sqrt{\pi}} \cdot e^{-\gamma} d\gamma
\]

\[
= \frac{1}{2} - \frac{1}{2} \int_0^\infty e^{-\gamma(1+\gamma_0)} \cdot \frac{1}{\sqrt{\gamma}} \cdot \frac{1}{\sqrt{\pi}} d\gamma
\]

Substituting $t^2 = \gamma$ s.t. $2\sqrt{\gamma}.dt = d\gamma$
\[ I = \frac{1}{2} - \frac{1}{2} \int_{0}^{\infty} e^{-i(t + \gamma_0) \gamma_0} \frac{2}{\sqrt{\pi}} dt \]
\[ = \frac{1}{2} \left[ 1 - \frac{2}{\sqrt{\pi}} \int_{0}^{\infty} e^{-i(t + \gamma_0) \gamma_0} dt \right] \]

This is now a standard integral s.t. \( \frac{1}{\sigma \sqrt{2\pi}} \int_{-\infty}^{\infty} e^{-x^2/2\sigma^2} dx = \frac{1}{2} \)

\[ \therefore \frac{2}{\sqrt{\pi}} \int_{0}^{\infty} e^{-i(t + \gamma_0) \gamma_0} dt = \frac{2}{\sqrt{\pi}} \cdot \frac{1}{2} \sqrt{\frac{2\pi \cdot \frac{1}{2}}{1 + \frac{1}{\gamma_0}}} = \frac{1}{\sqrt{1 + \frac{1}{\gamma_0}}} \text{ using } \sigma^2 = \frac{1}{2} \frac{1}{1 + \frac{1}{\gamma_0}} \]

So,

\[ I = \frac{1}{2} \left( 1 - \frac{1}{\sqrt{1 + \frac{1}{\gamma_0}}} \right) \text{ as required.} \]

**Fourier Transform of Raised Cosine Filter**

The Raised-Cosine filter is defined by the equations:-

\[ H(f) = T \text{ for } 0 \leq |f| \leq \frac{1}{2T}(1-\alpha) \]

\[ H(f) = \frac{T}{2} \left( 1 + \cos\left(\pi \cdot \frac{f - \frac{1}{2T} \cdot (1-\alpha)}{\frac{2\pi}{2T}}\right) \right) \text{ for } \frac{1}{2T}(1-\alpha) \leq |f| \leq \frac{1}{2T}(1+\alpha) \]

\[ H(f) = 0 \text{ for } |f| \geq \frac{1}{2T}(1+\alpha) \]

For implementation within a software simulation, filter taps for a digital filter are required. These are obtained through taking the Fourier Transform of \( H(f) \).

\( H(f) \) is symmetrical in \( f \), and so the taps are given by:-

\[ h(t) = 2 \int_{0}^{\infty} H(f) \cdot \cos(2\pi ft) \cdot df \]
\[
= 2 \int_0^{\frac{1}{\alpha} T} T \cdot \cos(2\pi f) \cdot df + 2 \int_0^{\frac{1}{\alpha} T} \left(1 + \cos\left(\pi \cdot \frac{f - \frac{1}{\alpha} \cdot (1 - \alpha)}{2\alpha}\right)\right) \cdot \cos(2\pi f) \cdot df
\]

= \left[\frac{T}{\pi} \cdot \sin(2\pi f)\right]_0^{\frac{1}{\alpha} T} + \left[\frac{T}{2\pi} \cdot \sin(2\pi f)\right]_0^{\frac{1}{\alpha} T} +
\int_0^{\frac{1}{\alpha} T} T \cdot \cos\left(\pi T \cdot \frac{f - \frac{1}{\alpha} \cdot (1 - \alpha)}{\alpha}\right) \cdot \cos(2\pi f) \cdot df
\]

Define

\[
I = \int_0^{\frac{1}{\alpha} T} T \cdot \cos\left(\pi T \cdot \frac{f - \frac{1}{\alpha} \cdot (1 - \alpha)}{\alpha}\right) \cdot \cos(2\pi f) \cdot df
\]

\[
h(t) = \left[\frac{T}{\pi} \cdot \sin(2\pi f)\right]_0^{\frac{1}{\alpha} T} + \left[\frac{T}{2\pi} \cdot \sin(2\pi f)\right]_0^{\frac{1}{\alpha} T} + I
\]

\[
= \frac{T}{\pi} \left[\sin\left(\frac{\pi}{T} (1 - \alpha) + \cos\frac{\pi}{T} \cdot \sin\frac{\alpha \pi}{T}\right)\right] + I
\]

\[
= \frac{T}{\pi} \left[\left(\sin\frac{\pi}{T} \cos\frac{\alpha \pi}{T} - \cos\frac{\pi}{T} \sin\frac{\alpha \pi}{T}\right) + \cos\frac{\pi}{T} \cdot \sin\frac{\alpha \pi}{T}\right] + I
\]

\[
= \frac{T}{\pi} \cdot \sin\frac{\pi}{T} \cos\frac{\alpha \pi}{T} + I
\]

Define

\[
\theta = \frac{\pi T}{\alpha} \cdot (f - \frac{1}{\alpha}) \text{ s.t. } d\theta = \frac{\pi T}{\alpha} \cdot df
\]

\[
I = T \cdot \int_\frac{\pi}{2}^{\frac{\pi}{2}} \cos\left(\theta + \frac{\pi}{2}\right) \cdot \cos\left(2\pi \cdot \left[\frac{\alpha \theta}{\pi T} + \frac{1}{2}\right]\right) \cdot \frac{\alpha}{\pi T} \cdot d\theta
\]

180
\[
\begin{align*}
&= -\frac{\alpha}{\pi} \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \sin \theta \cdot \cos \left( \frac{\pi t}{T} + \frac{2\alpha t}{T} \right) \cdot d\theta \\
&= -\frac{\alpha}{\pi} \cos \frac{\pi t}{T} \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \sin \theta \cdot \cos \frac{2\alpha t}{T} \cdot d\theta + \frac{\alpha}{\pi} \sin \frac{\pi t}{T} \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \sin \theta \cdot \sin \frac{2\alpha t}{T} \cdot d\theta \\
\end{align*}
\]

Noting that \( \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \sin \theta \cdot \cos k\theta \cdot d\theta = 0 \), \( \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \sin \theta \cdot \sin k\theta \cdot d\theta = \frac{1}{1 - k^2} \left( 2k \cos \frac{k\pi}{2} \right) \) if \( k^2 \neq 1 \)

\[
I = \frac{\alpha}{\pi} \cdot \frac{\pi t}{T} \cdot \frac{1}{1 - \left[ \frac{2\alpha t}{T} \right]^2} \cdot \frac{4\alpha^2 t^2}{T^2} \cdot \frac{\pi}{T} \cdot \cos \frac{\pi t}{T} \\
= \frac{1}{1 - \left[ \frac{2\alpha t}{T} \right]^2} \cdot \frac{4\alpha^2 t^2}{T^2} \cdot \frac{\pi}{T} \cdot \cos \frac{\pi t}{T} \\
= \frac{T}{\pi} \cdot \sin \frac{\pi t}{T} \cdot \cos \frac{\alpha t}{T} + \frac{1}{1 - \left[ \frac{2\alpha t}{T} \right]^2} \cdot \frac{4\alpha^2 t^2}{T^2} \cdot \frac{\pi}{T} \cdot \cos \frac{\alpha t}{T} \\
= \frac{T}{\pi} \cdot \sin \frac{\pi t}{T} \cdot \cos \frac{\alpha t}{T} \left[ 1 + \frac{1}{1 - \left( \frac{2\alpha t}{T} \right)^2} \cdot \frac{4\alpha^2 t^2}{T^2} \right]
\]

Finally, we get the result that:

\[
h(t) = \frac{1}{1 - \left( \frac{2\alpha t}{T} \right)^2} \cdot \frac{T}{\pi} \cdot \sin \frac{\pi t}{T} \cdot \cos \frac{\alpha t}{T} \quad \text{provided } T \neq |2\alpha|
\]

181
If \( T = 2\alpha \) (\( k=1 \)),
\[
\frac{\pi}{2} \int \sin \theta \cdot \sin k\theta \cdot d\theta = \frac{\pi}{2} \int \sin^2 \theta \cdot d\theta = \frac{\pi}{2}
\]

So, \( I = \frac{\alpha}{\pi} \cdot \sin \frac{\pi t}{\pi} \cdot \frac{\pi}{2} = \frac{\alpha}{2} \cdot \sin \frac{\pi t}{T} \)

\[
h(t) = \frac{T}{\pi} \cdot \sin \frac{\pi t}{T} \cdot \cos \frac{\alpha \pi t}{T} + \frac{\alpha}{2} \cdot \sin \frac{\pi t}{T} = \sin \frac{\pi t}{T} \left( \frac{T}{\pi} \cdot \cos \frac{\pi}{2} + \frac{T}{4t} \right)
\]

The equations for \( h(t) \) are therefore given by:-

\[
h(t) = \frac{1}{1 - \left( \frac{2\alpha t}{T} \right)^2} \cdot \frac{T}{\pi} \cdot \sin \frac{\pi t}{T} \cdot \cos \frac{\alpha \pi t}{T} \quad \text{provided} \ T \neq 2\alpha |t|, \ t \neq 0
\]

\[
h(0) = 1
\]

\[
h(t) = \frac{T}{\pi} \cdot \sin \left( \frac{\pi t}{T} \right) \cdot \frac{\pi}{4} \quad \text{if} \ T = 2\alpha |t|
\]

**Fourier Transform of Square-Root Raised Cosine Filter**

The Square Root Raised Cosine filter is defined by the function:-

\[
H(f) = \sqrt{T} \left( 1 + \cos \left( \pi \cdot \frac{f - \frac{1}{2T} \cdot (1-\alpha)}{\frac{2\alpha}{2T}} \right) \right)
\]

for \( 0 \leq |f| \leq \frac{1}{2T} (1-\alpha) \)

\[
= 0 \quad \text{for} \ \frac{1}{2T} (1-\alpha) \leq |f| \leq \frac{1}{2T} (1+\alpha)
\]

for \( |f| \geq \frac{1}{2T} (1+\alpha) \)

Its Fourier Transform is given by (after removing the 'sin' term as \( H(f) \) is symmetrical in \( f \)):-
\[ h(t) = 2 \cdot \left\{ \int_0^{\frac{1}{2T}(1-\alpha)} \sqrt{T} \cdot \cos(2\pi ft) df + \int_0^{\frac{1}{2T}(1+\alpha)} \frac{T}{2} \left[ 1 + \cos \left( \pi \cdot \frac{f - \frac{1}{2T} \cdot (1-\alpha)}{2\alpha} \right) \right] \cdot \cos(2\pi ft) df \right\} \]

Noting that \( 1 + \cos 2\theta = 2\cos^2 \theta \), define \( 2\theta = \frac{\pi T}{\alpha} \cdot \left( f - \frac{1}{2T} \cdot (1-\alpha) \right) \).

This gives \( f = \frac{\alpha}{\pi T} \cdot 2\theta + \frac{1}{2T} \cdot (1-\alpha) \) and \( df = \frac{2\alpha}{\pi T} \cdot d\theta \).

So, \( h(t) = 2\sqrt{T} \cdot \left( \int_0^{\frac{1}{2T}(1-\alpha)} \cos(2\pi ft) df + \frac{2\alpha}{\pi T} \cdot \int_0^{\frac{\pi}{2}} \cos \theta \cdot \cos \left( 2\pi \cdot \left[ \frac{2\alpha \theta}{\pi T} + \frac{1-\alpha}{2T} \right] \right) \cdot d\theta \right) \)

Ignoring the first term for now (as this is trivial to integrate):-

\[ I = \int_0^{\frac{\pi}{2}} \cos \theta \cdot \cos \left( \frac{4\alpha \theta}{T} + \frac{\pi \theta}{T} (1-\alpha) \right) \cdot d\theta \]

\[ = \int_0^{\frac{\pi}{2}} \cos \theta \cdot \cos \frac{4\alpha \theta}{T} \cdot \cos \frac{\pi (1-\alpha)}{T} \cdot d\theta - \int_0^{\frac{\pi}{2}} \cos \theta \cdot \sin \frac{4\alpha \theta}{T} \cdot \sin \frac{\pi (1-\alpha)}{T} \cdot d\theta \]

Noting the results that

\[ \int_0^{\frac{\pi}{2}} \cos \theta \cdot \cos n\theta \cdot d\theta = \frac{1}{1-n^2} \cdot \cos \frac{n\pi}{2} \quad \text{and} \quad \int_0^{\frac{\pi}{2}} \cos \theta \cdot \sin n\theta \cdot d\theta = \frac{1}{1-n^2} \left( \sin \frac{n\pi}{2} - n \right) \]

provided that \( n^2 \neq 1 \)

\[ I = \cos \frac{\pi (1-\alpha)}{T} \cdot \frac{1}{1-\left( \frac{4\alpha}{T} \right)^2} \cdot \cos \frac{4\pi \alpha}{2T} - \sin \frac{\pi (1-\alpha)}{T} \cdot \frac{1}{1-\left( \frac{4\alpha}{T} \right)^2} \left( \sin \frac{4\pi \alpha}{2T} - \frac{4\alpha}{T} \right) \]

183
\[
\begin{align*}
&= \frac{1}{1 - \left(\frac{4\alpha}{T}\right)^2} \left( \cos \left[ \frac{\pi(1-\alpha)}{T} + \frac{4\pi\alpha t}{2T} \right] + \frac{4\alpha}{T} \cdot \sin \left[ \frac{\pi(1-\alpha)}{T} \right] \right) \\
&= \frac{1}{1 - \left(\frac{4\alpha}{T}\right)^2} \left( \cos \frac{\pi(1+\alpha)}{T} + \frac{4\alpha}{T} \cdot \sin \frac{\pi(1-\alpha)}{T} \right)
\end{align*}
\]

Thus,

\[
\begin{align*}
&= \frac{2\sqrt{T}}{2\pi t} \cdot \sin \left( \frac{\pi}{T} (1-\alpha) \right) + \frac{4\alpha}{\pi \sqrt{T}} \cdot \frac{1}{1 - \left(\frac{4\alpha}{T}\right)^2} \left( \cos \frac{\pi(1+\alpha)}{T} + \frac{4\alpha}{T} \cdot \sin \frac{\pi(1-\alpha)}{T} \right)
\end{align*}
\]

\[
\begin{align*}
&= \sin \frac{\pi(1-\alpha)}{T} \cdot \frac{4\alpha}{\pi \sqrt{T}} \left[ \frac{4\alpha}{T} \cdot \frac{1}{1 - \left(\frac{4\alpha}{T}\right)^2} + \frac{T}{4\alpha} \right] + \frac{4\alpha}{\pi \sqrt{T}} \cdot \frac{1}{1 - \left(\frac{4\alpha}{T}\right)^2} \cdot \cos \frac{\pi(1+\alpha)}{T}
\end{align*}
\]

Given \( \frac{x}{1-x^2} + \frac{1}{x} = \frac{1}{x} \cdot \frac{x^2+1-x^2}{1-x^2} = \frac{1}{x} \cdot \frac{1}{1-x^2} \), this becomes :-

\[
\begin{align*}
&= \sin \frac{\pi(1-\alpha)}{T} \cdot \frac{4\alpha}{\pi \sqrt{T}} \cdot \frac{T}{4\alpha} \cdot \frac{1}{1 - \left(\frac{4\alpha}{T}\right)^2} + \frac{4\alpha}{\pi \sqrt{T}} \cdot \frac{1}{1 - \left(\frac{4\alpha}{T}\right)^2} \cdot \cos \frac{\pi(1+\alpha)}{T}
\end{align*}
\]

which finally leads to the result that

\[
\begin{align*}
&= \frac{1}{1 - \left(\frac{4\alpha}{T}\right)^2} \cdot \left\{ \frac{\sqrt{T}}{\pi} \cdot \sin \frac{\pi(1-\alpha)}{T} + \frac{4\alpha}{\pi \sqrt{T}} \cdot \cos \frac{\pi(1+\alpha)}{T} \right\}
\end{align*}
\]

\[
\begin{align*}
h(t=0) &= \frac{1}{\sqrt{T}} \cdot \left\{ \frac{4\alpha}{\pi} + 1 - \alpha \right\}
\end{align*}
\]

In the case that \( 4\alpha = \pm T \), these equations are not valid. The equation for \( I \) becomes :-
\[ I = \int_{0}^{\frac{\pi}{2}} \cos \theta \cdot \cos \left( \pm \theta + \frac{\pi t}{T} (1-\alpha) \right) \cdot d\theta \]

\[ = \int_{0}^{\frac{\pi}{2}} \cos \theta \cdot \cos \theta \cdot \cos \left( \frac{\pi (1-\alpha)}{T} \right) \cdot d\theta - \int_{0}^{\frac{\pi}{2}} \cos \theta \cdot \sin \theta \cdot \sin \left( \frac{\pi (1-\alpha)}{T} \right) \cdot d\theta \]

where the \pm sign in the second integral is + for \( t \geq 0 \), - for \( t < 0 \).

Substituting \( \cos^2 \theta = \frac{1 + \cos 2\theta}{2} \), \( \cos \theta \cdot \sin \theta = \sin 2\theta \), and for \( 4\alpha t = \pm T \) this becomes :-

\[ I = \frac{1}{2} \cdot \cos \frac{\pi (1-\alpha)}{4\alpha} \cdot \left[ \left( 1 + \cos 2\theta \right) \cdot d\theta \right] - \frac{1}{2} \cdot \sin \frac{\pi (1-\alpha)}{4\alpha} \cdot \left[ \sin 2\theta \cdot d\theta \right] \]

\[ = \frac{1}{2} \cdot \cos \frac{\pi (1-\alpha)}{4\alpha} \cdot \left[ \vartheta + \frac{\sin 2\theta}{2} \right]_{0}^{\frac{\pi}{2}} + \frac{1}{2} \cdot \sin \frac{\pi (1-\alpha)}{4\alpha} \cdot \left[ \cos 2\theta \right]_{0}^{\frac{\pi}{2}} \]

which is equal to :-

\[ I = \frac{1}{2} \cdot \cos \frac{\pi (1-\alpha)}{4\alpha} \cdot \frac{\pi}{2} - \frac{1}{2} \cdot \sin \frac{\pi (1-\alpha)}{4\alpha} = \frac{\pi}{4} \cdot \cos \frac{\pi (1-\alpha)}{4\alpha} - \frac{1}{2} \cdot \sin \frac{\pi (1-\alpha)}{4\alpha} \]

This gives

\[ h(t = \pm \frac{T}{4\alpha}) = \frac{8\alpha}{2\pi \sqrt{T}} \cdot \sin \frac{\pi (1-\alpha)}{4\alpha} + \frac{4\alpha}{\pi \sqrt{T}} \cdot \left\{ \frac{\pi}{4} \cdot \cos \frac{\pi (1-\alpha)}{4\alpha} - \frac{1}{2} \cdot \sin \frac{\pi (1-\alpha)}{4\alpha} \right\} \]

\[ = \frac{4\alpha}{\pi \sqrt{T}} \cdot \left\{ 1 - \frac{1}{2} \cdot \sin \frac{\pi (1-\alpha)}{4\alpha} + \frac{\pi}{4} \cdot \cos \frac{\pi (1-\alpha)}{4\alpha} \right\} \]
The equations for \( h(t) \) are therefore given by:

\[
h(t) = \frac{1}{1 - \left( \frac{4\pi}{T} \right)^2} \left\{ \frac{\sqrt{T}}{\pi} \cdot \sin \frac{\pi(1-\alpha)}{T} + \frac{4\alpha}{\pi \sqrt{T}} \cdot \cos \frac{\pi(1+\alpha)}{T} \right\} \quad \text{provided } T \neq 4\pi\lvert t \rvert, t \neq 0
\]

\[
h(t = 0) = \frac{1}{\sqrt{T}} \cdot \left\{ (1-\alpha) + \frac{4\alpha}{\pi} \right\}
\]

\[
h(t = \pm \frac{T}{4\alpha}) = \frac{\alpha}{\sqrt{T}} \cdot \left\{ \frac{2}{\pi} \cdot \sin \frac{\pi(1-\alpha)}{4\alpha} + \cos \frac{\pi(1-\alpha)}{4\alpha} \right\}
\]
Appendix C : Image Quality Measurements
Appendix C: Image Quality Measurements

Throughout this thesis, measurements of image quality are presented in terms of the average PSNR (peak signal to noise ratio) of the luminance component of the video which is given by:

$$PSNR(y,x) = 10 \cdot \log_{10} \left( \frac{\sum 255^2}{\sum (y_i - x_i)^2} \right)$$

for 8-bit images.

This provides a useful indication of the similarity of the two images, but cannot be taken as more than a purely mathematical construct. The major problem with this formulation is that it does not take into account the properties of the human visual system which will be the ultimate judge of the quality of any video transmission system. The frames of figure 108 are two frames with near identical PSNR: the first is a frame taken from decoding an MPEG-II bitstream containing errors, the second is the original frame with random pixels set to black or white until the PSNR values are equal.

![Figure 108: Comparison of images with near identical PSNR (from frame 11 of Mobile & Calendar).](image)

As can be seen, the concentration of errors in one region leads to a severe loss in overall quality as perceived by the human eye. The random pixel errors produce severe damage also, but in a completely different way. Quite which is preferable is dependent on the application, and on the observer, and so it is therefore essential that the actual frames of video
are considered as well as producing overall trends from PSNR curves. For this reason, both PSNR curves and (some) frames are presented in this thesis.

Another problem with the PSNR figure can only be observed easily when the actual frames are played at the correct speed. If there are errors present in the video, which move from frame to frame, then the process of playing the actual video in real-time makes the errors easier to spot (as the human visual system responds well to changes in the scene it is looking at). Such errors, though visible from the frames shown in this thesis, are therefore best demonstrated through actual videos, and so a disk is included in the back of this thesis which includes sample MPEG-II bitstreams (see Appendix D for details).
Appendix D : Index of Media
Appendix D : Index of Media

Table 26 outlines the contents of the CD-ROM that is included in the back cover of this thesis. The purpose of this disk is to make available the MPEG-II bitstreams produced in Chapter Six. For convenience, MPEG-II players are included for the following operating systems: Windows 95/NT\textsuperscript{38}, Linux, SunOS 4 and SunOS 5. Source code is included for compiling on other UNIX / X-Windows systems if required.

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Parameters</th>
<th>CD Location</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>P1 BCH</td>
<td>P2 BCH</td>
</tr>
<tr>
<td>Mobile &amp; Calendar</td>
<td>63,39,4</td>
<td>63,57,1</td>
</tr>
<tr>
<td>Mobile &amp; Calendar</td>
<td>63,30,6</td>
<td>63,57,1</td>
</tr>
<tr>
<td>Mobile &amp; Calendar</td>
<td>63,39,4</td>
<td>63,57,1</td>
</tr>
<tr>
<td>Mobile &amp; Calendar</td>
<td>63,39,4</td>
<td>none</td>
</tr>
<tr>
<td>Mobile &amp; Calendar</td>
<td>63,39,4</td>
<td>63,57,1</td>
</tr>
<tr>
<td>Mobile &amp; Calendar</td>
<td>63,39,4</td>
<td>63,57,1</td>
</tr>
<tr>
<td>Football</td>
<td>63,39,4</td>
<td>63,57,1</td>
</tr>
<tr>
<td>Football</td>
<td>63,39,4</td>
<td>63,57,1</td>
</tr>
</tbody>
</table>

\textit{Table 26: Contents of the enclosed CD-ROM}

\textsuperscript{38} This implementation of the MPEG-II player is not particularly stable, but is based on the same encoder / decoder implementation that the rest of this thesis has used. Some syntactic errors are present in the bitstreams (due to the receiving transcoder not testing the partition one data sufficiently thoroughly), and so they do not necessarily play properly (see the relevant "Read.Me" files on the CD-ROM).
Appendix E: Publications
Appendix E: Publications

Pages


Efficient Partitioning of Coded Video Bitstreams for Multi-channel Wireless Transmission

H. Gharavi and Christopher I. Richards
Department of Electronic and Electrical Engineering,
Loughborough University,
Loughborough,
Leics., U.K.
E-mail: H.Gharavi@lboro.ac.uk Tel: +44 (0)1509 222891
E-mail: C.I.Richards@lboro.ac.uk Tel: +44 (0)1509 228120

Abstract

This paper introduces an efficient technique for dividing a pre-coded MPEG-2 bitstream into an arbitrary number of partitions, in decreasing order of visual importance. For a fixed partition size ratio, this has been achieved by VLC code interleaving within a macroblock or group of macroblocks. The algorithm is capable of recreating a syntactically correct bitstream, regardless of the error rate on secondary partitions, making it applicable to many efficient transmission schemes.

Introduction

There are many transmission schemes where it is advantageous to separate the data beforehand into two or more layers that have different error resilience requirements. For example, Quadrature Amplitude Modulation (QAM) has very different error probabilities associated with the different bits that make up a given symbol. The ability to use such modulation schemes without prohibitively large amounts of error correction is particularly important for high quality coded video transmission where the bandwidth requirements are very high. Stedman et al. [1] have used Quadrature Mirror Filters (QMF) to code video into two separate partitions suited to the different error rates of 16-QAM in a Rayleigh channel.

The discrete cosine transform (DCT) based MPEG-2 coding [2] is being advocated as the coding scheme to be used for commercial digital broadcasting, and much work is being done to produce real-time encoding and decoding chips. In this paper, we describe a method of partitioning an already coded MPEG-2 bitstream into an arbitrary number of partitions (Figure 1), in decreasing visual importance. This has the advantage of separating the coding process from the transmission process, but inevitably results in some extra data being transmitted. The overall output bit-rate could be controlled via algorithms such as those of Sun, Kwok and Zdepski [3], although we have not implemented them for this paper.
Separation into prioritised partitions

Analysis of several MPEG-2 coded video sequences shows that the bulk of an MPEG-2 bitstream is contained within the macroblock and block layers, although as the compression level increases this proportion of the total data decreases. It is also important to note that the layers above the macroblock contain information that affects how the bitstream is interpreted, and must be well protected from errors. It is therefore necessary to partition the bitstream at either the slice layer (as it contains the macroblocks), the macroblock layer or the block layer. Because the secondary partition(s) are expected to be received with errors, it is necessary to synchronise the partitions. This makes partitioning at the slice level an unattractive option (as synchronising at the start of the next slice means a single error on a secondary partition can affect an entire row of macroblocks). However, synchronising at a block level is unlikely to be very efficient, so we have initially investigated partitioning at the macroblock level.

Figure 2a shows a typical MPEG-2 macroblock. In the scheme we have developed, this would be re-coded as shown in Figure 2b. Since picture information is ultimately transmitted as DCT coefficients there is an in-built order of visual importance, which we utilise by interleaving the blocks by sending one variable length (run length, level) code for each non-finished block. The effect of this is to send the lowest frequency coefficients for each block first, then the higher frequency coefficients, etc. A partition code (PC) is used to mark the point in the bitstream where the switch to the second partition occurs. If more than two partitions exist, the switch to subsequent partitions occurs at the end of this PC - regardless of whether this occurs in the middle of a VLC.

![Fig. 2a. Macroblock from Original MPEG-2 Bitstream.](image)

![Fig. 2b. Interleaved Block Partitioning.](image)

Experimentation has shown that inserting a partition code is, in general, marginally more efficient than starting a macroblock with a code giving the exact offset of the partition point (which could then be part way through a VLC code), and also improves the SNR (up to 1dB) of the decoded video as the partition code adds less data to the primary partition.

The PC has to be quite short as it will occur in every macroblock containing picture data, however there are no suitable free codes available. The only viable solution is to modify an existing (frequently occurring) code, for example the End of Block (EOB) code which is present for every coded block. For our system, we have defined the PC code as an EOB code with a trailing '1' - all EOB codes that would have occurred in partition 1 have a trailing '0' added to them. This extension to the actual EOB code only affects blocks that are entirely coded within partition 1.

The PC is inserted as near to the ideal partitioning location as possible, but cannot occur before this point as this code marks the point at which the secondary partitions are synchronised at the start of the next macroblock. Partition 1 cannot therefore contain less information than the secondary partition. The PC code can only be inserted at a location where a decoder would be looking for a VLC, so the PC code usually has to be delayed by a few bits. This leads to a dead-time (DT) on the last partition, which is unavoidable. Typical dead-time length is 6.5% of the original amount of data (12.6 bits per macroblock) for the CCIR-601 sequence ‘Mobile’ coded at 4Mb/s, which together with the PC code constitutes the bulk of the overheads added by the splitting process.
During the dead-time, the last partition is filled with zeros, although it is not particularly important how it is filled as it is not decoded. The use of PC to re-synchronise the decoding of all the partitions at the start of each macroblock means that an error detected on the second partition only affects the decoding of this macroblock.

In order to recover a syntactically legal MPEG-2 bitstream, as soon as an error is detected all the coded blocks are closed with EOB codes. This may necessitate the removal of partly transmitted ESCAPE codes, and also the insertion of a single DCT code to make a block contain some data (else the macroblock would be classed as 'not coded'). Errors occurring within partition 1 affect the decoding process as far as the start of the next slice.

SNR vs P(e) for MOBILE 4Mb/s (40 frames)

Figure 3 shows the SNR performance of our system (interleaved PC) contrasting it with a similar system without the blocks being interleaved (non-interleaved PC). These results are for a 2 partition system with white noise added to the secondary partition (the results are for the CCIR-601 sequence 'Mobile' coded at 4Mb/s). Error correcting codes are not applied to this secondary partition. The graph shows a significant gain in SNR for the interleaved system (about 8dB), which has had no extra data added. It should be noted that the bitstream can be decoded properly even when the entire secondary partition is received in error, thus proving that the scheme will be suitable when there are burst of errors in the received data.

There is no fundamental requirement that the partitions should be of equal size and, in fact, the partitions must be of different sizes when error correcting codes are to be applied to any of the partitions. In this case, the partition point is calculated so as to make the error protected bitstreams have equal size. Figure 3 also shows that there is a slight decrease in SNR performance when more DCT data is placed on the second partition, due to the presence of a BCH (255,223,4) error correcting code applied to partition 1.

One significant problem remains to be addressed. As the coding rate is decreased the efficiency of the system decreases. The overall overheads of the system increase as shown in Figure 4 (single 'macroblock grouping' curve). This is due to the fact that the PC becomes a significant proportion of the data for a macroblock, and also that as there are fewer AC coefficients, balancing the partitions is more difficult. We are currently investigating a solution to this problem based on grouping macroblocks together when the coding rate is low. Figure 4 shows preliminary results of how this affects the overall efficiency. The reduction in overheads is due to there being fewer partition codes, and less dead-time. However, regrouping the macroblocks alters the way in which errors affect the decoded images, as a single error will affect the entire group of macroblocks.
Summary

The scheme we have developed is, therefore, suitable for use in high quality transmission when the bit rate is relatively high. We can successfully reconstruct an MPEG-2 bitstream regardless of how errors affect the secondary partition(s), with the advantage that a burst of errors is similar in effect to a single error within each macroblock that the burst covers. When the coding rate is low, however, the efficiency of the partitioning at the individual macroblock level decreases, and so the macroblocks need to be grouped together to control the amount of data added by the splitting process.

Acknowledgement

We would like to thank Mr D. Biwott for simulating the error correcting codes.

References


Partitioning of MPEG Coded Video Bitstreams for Wireless Transmission

Hamid Gharavi, Fellow, IEEE and Christopher I. Richards

Abstract—This paper introduces an efficient technique for dividing a precoded Moving Pictures Expert Group-2 (MPEG-2) bitstream into an arbitrary number of partitions in decreasing order of visual importance. For a fixed partition size ratio, this has been achieved by variable-length code (VLC) interleaving within a macroblock or group of macroblocks. The algorithm is capable of recreating a syntactically correct bitstream, regardless of the error rate on secondary partitions. This renders it applicable to many efficient transmission schemes, particularly over wireless channels.

I. INTRODUCTION

There are many transmission schemes where it is advantageous to separate the data beforehand into two or more layers that have differing error-resilience requirements. For example, 16-level quadrature amplitude modulation (QAM) has very different error probabilities associated with the different bits that make up a given symbol. This property was exploited earlier for the transmission of subband coded video signals [1]. In that scheme, the 4-b symbol of a 16-level QAM was interleaved to form two data subchannels—the first subchannel being represented by the most significant bit (MSB) of both in-phase and quadrature-phase codes, and the second subchannel formed by the remaining two bits (i.e., least significant bits). The ability to use such a modulation scheme for video signals would require splitting the coded bitstream into equal sized partitions. However, the concept of subchannel transmission is not limited to the use of multilevel QAM and, indeed, independent modulation techniques with different degrees of robustness against transmission errors can also be considered.

In this paper, we describe a method of partitioning an already coded Moving Pictures Expert Group-2 (MPEG-2) bitstream into equally sized partitions in order of decreasing visual importance (see Fig. 1). The method has the advantage of being independent from the coding process, but inevitably results in some extra data being transmitted for synchronization.

II. SEPARATION INTO PRIORITIZED PARTITIONS

Analysis of several MPEG-2 [2] coded video sequences shows that the bulk of the bitstreams are contained within the macroblock and block layers. However, as the compression level increases, the proportion of the total data is expected to decrease. It is also important to note that the layers above the macroblock contain information that affects how the bitstream is interpreted, and must be well protected from errors. It is, therefore, necessary to partition the bitstream at the slice layer (as it contains the macroblocks), the macroblock layer, or the block layer. Because the secondary partition(s) are expected to be received with errors, it is essential to synchronize the partitions. However, synchronizing at a block level is unlikely to be very efficient, so we have investigated partitioning at the macroblock level and group of macroblocks.

Fig. 2(a) shows a typical MPEG-2 macroblock. In our scheme this would be recoded as shown in Fig. 2(b). Since picture information is ultimately transmitted as discrete cosine transform (DCT) coefficients, there is an in-built order of visual importance that we utilize by interleaving the blocks by sending one variable-length code (VLC) for each nonfinished block. The effect of this is to send the lowest frequency coefficients for each block first, then the higher frequency coefficients, etc. This arrangement, which is based on a macroblock, can be extended over a group of macroblocks to enhance the efficiency of the VLC interleaving and reduce the overhead for synchronization. The following presents the partitioning procedure.

A partition code (PC), as shown in Fig. 2, is used to mark the point in the bitstream where the switch to the second partition occurs. The PC has to be quite short, as it can occur as often as every macroblock (or group of macroblocks) containing picture data. However, there are no suitable free codes available. The only viable solution is to modify an existing (frequently occurring) code; for example, the end of block (EOB) code, which is present for every coded block. For our system, we have defined the PC code as an EOB code with a trailing one—all EOB codes that would have occurred in partition one have a trailing zero added to them. This extension to the actual EOB code only affects blocks that are entirely...
Fig. 2. Macroblock partitioning of MPEG-2 coded bitstream. (a) Macroblock from original MPEG-2 bitstream. (b) Proposed interleaved synchronized partitioning (two-level). Bi: the ith block within a macro-block. PC: partition code. DT: dead time.

PSNR vs P(e) for 'Mobile and Calendar' 4Mb/s (40 frames)

Fig. 3. PSNR of decoded video versus partition two error probability.

coded within partition one.

The PC is inserted as near to the ideal partitioning location as possible as this code marks the position at which the secondary partitions are synchronized at the start of the next group of macroblocks. The exact location of a PC is governed by the following rules: i) a PC cannot occur in the middle of a VLC and, therefore, has to be inserted at a location where a decoder would be seeking a VLC, and ii) the PC cannot occur in non-DCT information including the first VLC. These restrictions lead to a dead time (DT) on the second partition, which is unavoidable. Typical dead-time lengths together with the PC code constitute the bulk of the overheads added by the splitting process. During the dead time, the last partition is filled with zeros, although it is not particularly important how it is filled as it is not decoded.

For instance, in the case of equal size partitions where no forward error corrections (FEC's) are considered, the PC should be placed at the center point of the macroblock bitstream including the PC (i.e., using upper integer division for odd numbers). If this point occurs anywhere within a VLC, e.g., at its jth bit, the following arrangement will take place: if j is equal or smaller than the length of the partition code, the PC is placed before the VLC, otherwise it will be after. In the event where the halfway point happens to be exactly at the end of a VLC (and if the length of this VLC is equal or less than the length of the partition codeword), the PC will be put before the VLC. This arrangement guarantees that the length of the secondary partition will always be equal to or smaller than the length of the primary partition. This will also ensure that the DT length is as short as possible.

The use of a PC to resynchronize the decoding of all the partitions at the start of each group of macroblocks means that an error detected on the second partition only affects the decoding of these macroblocks. In order to recover a
fading channels. This would be based on the assumption that partition two includes only the missing EOB's.\footnote{If need be, the remainder of the partition can be filled with stuffing bits and inserted at the next available slice-header to avoid any substantial drift in the bitstream.} This will cause the removal of some of the transmitted DCT codes. Nevertheless, the loss of the secondary partition gracefully reduces the quality of the reconstructed macroblocks which could also affect their associated macroblocks in the other P or B frames until the next I frame is encoded. However, errors occurring within partition 1 affect the decoding process as far as the start of the next slice.

Fig. 3 shows the PSNR performance of our system (interleaved PC), contrasting it with a similar system without the blocks being interleaved (noninterleaved PC). These results are for a two-partition system based on a number of macroblock groupings: single macroblock, eight macroblocks, and slice, with white noise added to the secondary partition (the results are for the CCIR-601 sequence Mobile and Calendar coded at 4 Mb/s). Error correcting codes are not applied to this secondary partition. The graph shows a significant gain in peak signal-to-noise ratio (PSNR) for the interleaved system (about 8 dB), which has had no extra data added. It should be noted that the bitstream can be decoded properly even when the entire secondary partition is received in error. This would prove that the scheme is particularly suitable when there are bursts of errors in the received data, as is the case of multipath fading channels.

There is no fundamental requirement that the partitions should be of equal size. Indeed, the partitions must be of different sizes when error-correcting codes are to be applied to any of the partitions. In this case, the partition point is calculated so as to make the error-protected bitstreams equally sized. For a single macroblock group, as shown in Fig. 3, there is a slight decrease in PSNR performance when more DCT data is placed on the second partition to compensate for a BCH (255,223,4) error-correcting code added to partition one. One significant problem remains to be addressed. When the coding rate is decreased, the efficiency of the system also decreases, as shown in Fig. 4 (single macroblock grouping curve). This is due to the fact that the PC and DT form a significant proportion of the data for a macroblock, and also that, as there are fewer alternating current coefficients, balancing the partitions is more difficult.

However, the efficiency of the partitioning, both in terms of VLC block interleaving as well as overheads, can be significantly enhanced by incorporating more macroblocks in the partitioning process. Fig. 4 also shows the results of multiple macroblock groupings and their effect on the overall partitioning performance. The reduction in overheads is due to there being fewer partition codes and less DT. Furthermore, since block activities can change from one macroblock to the next, VLC block interleaving can become more effective in partitioning the coded bitstream. At the same time, regrouping the macroblocks alters the way in which errors affect the reconstructed image, as a single error will impact the entire group of macroblocks. Consequently, this will prohibit the involvement of a significantly large number of macroblocks in the partitioning process (see Fig. 3).

III. CONCLUSION

The scheme we have developed is suitable for use in high-quality transmission when the bit rate is relatively high (e.g., 2 Mb/s and above). We can successfully reconstruct an MPEG-2 bitstream regardless of the nature of errors on the secondary partition(s). In other words, a burst of errors has a similar effect as a single error and, thus, makes it suitable for the transmission over mobile channels. When the coding rate is low however, the efficiency of partitioning decreases.
REFERENCES


Bibliography
Bibliography


44 Video Coding for narrow telecommunications channels at < 64 kbit/s, ITU-T Recommendation H.263, 1996.


64 Ng, W.Y., *Subband coding for visual communications*, M.Phil. thesis, Loughborough University, April 1997.


78 North American Digital Mobile Cellular System (NADCMS) specification IS-54.


86 The Reed-Solomon error correcting package was obtained from http://people.qualcomm.com/karn/feccode/ This is a rewrite of code by Robert Morelos-Zaragoza (robert@spectra.eng.hawaii.edu) and Hari Thirumoorthy (harit@spectra.eng.hawaii.edu), which was in turn based on an earlier program by Simon Rockliff (simon@augean.ua.oz.au).


89 The Greenleaf HDTV sequence was supplied by Sony Corporation.

90 The ISO implementation of the MPEG-II coding standard was obtained from ftp.netcom.com/pub/cfogg.


95 Swann, R.E.M., and Kingsbury, N.G., Bandwidth efficient transmission of MPEG-II video over noisy mobile links, Personal copy received from author.


