Scalable video coding by stream morphing

**Author:**
Macnicol, James Roy

**Publication Date:**
2002

**DOI:**
https://doi.org/10.26190/unsworks/6894

**License:**
https://creativecommons.org/licenses/by-nc-nd/3.0/au/
Link to license to see what you are allowed to do with this resource.

Downloaded from http://hdl.handle.net/1959.4/60775 in https://unswworks.unsw.edu.au on 2023-10-23
Scalable video coding techniques are essential for transmission over communication channels where the amount of capacity available cannot be specified or known in advance and tends to fluctuate over time. Examples of such environments are streaming video over the Internet or mobile wireless networks. Scalable video systems allow for some video quality to be traded-off in exchange for improved service continuity when capacity is constrained. This thesis introduces "stream morphing", a new approach to the scalable video problem. Traditional systems, such as those described in the MPEG standards, create enhancement layers by measuring and coding the quantization error from the previous layer. In contrast, stream morphing works on a hierarchy of single-layer bitstreams of increasing quality. The enhancement layers of the scalable system encode the operations required to transform a single-layer bitstream that has already been received into another bitstream which is of higher quality. This process allows for the set of single-layer bitstreams to be simulcast in an efficient manner. The ability to move easily between a series of single-layer bitstreams and a scalable representation of the same video sequence is a significant new functionality that is not present in other scalable coding systems. The bandwidth efficiency of the new technique is also higher than the methods for SNR scalability defined in MPEG-2 and MPEG-4. This is confirmed by subjective test results over a range of different types of video sequence. In addition to implementation details and results for SNR scalability under VBR conditions, this thesis also develops a CBR rate control algorithm for the SNR scalable case and extends the stream morphing concept to cover spatial and temporal scalability.
Scalable Video Coding by Stream Morphing

Submitted by
James Roy Macnicol
for the degree of
Doctor of Philosophy

School of Electrical Engineering
Australian Defence Force Academy
University College
The University of New South Wales

OCTOBER 2002
(REVISED MAY 2003)
I hereby declare that this submission is my own work and to the best of my knowledge it contains no material previously published or written by another person, nor material which to a substantial extent has been accepted for the award of any other degree or diploma at UNSW or any other educational institution, except where due acknowledgement is made in the thesis. Any contribution made to the research by colleagues, with whom I have worked at UNSW or elsewhere, during my candidature, is fully acknowledged.

I also declare that the intellectual content of this thesis is the product of my own work, except to the extent that assistance from others in the project's design and conception or in style, presentation and linguistic expression is acknowledged.

Signed ..........................................

on this ........................ day of ............. in the year 2003
Acknowledgements

I would like to thank the following people for their part in ensuring this story has a happy ending:

- My supervisors Michael Frater and John Arnold for not only arranging for me to do this in the first place but also ensuring that the entire process was as pleasant and painless as one could possibly expect it to be. It should also be said that the quality of the thesis was improved greatly by their extremely careful reading and comments on the draft versions of the document.

- Telstra for providing the funding that paid my scholarship.

- My parents who (amongst other things) provided plenty of encouragement and trips away from Canberra where I was fed properly and got lots of work done.

- Finally I should also thank all those people from ANU that provided such worthwhile distractions, especially during the middle stages where things were going slowly. The idea behind stream morphing occurred to me during one of those long walks over there so they must take some (rather indirect) credit for it. Special thanks to Alex Smola, Jonathan McCabe, Jason Chen and Zbigniew Stachurski.
About the Accompanying CD

The CD that is included at the back of this volume contains the following items:

- Source code for the experimental implementation of stream morphing that was used to obtain the results shown in the thesis. The software has been mostly tested on Unix/Linux platforms although it should work under Microsoft Windows with relatively few changes.

- The test scripts and parameters used to generate all the results shown in the thesis.

- YUV files for the video sequences used in the subjective tests in Chapter 5 plus viewing software for Microsoft Windows and X11.

- \LaTeX{} source for the thesis plus all diagrams.

More information can be found in the file readme.txt in the root directory of the CD.
Abstract

Scalable video coding techniques are essential for transmission over communication channels where the amount of capacity available cannot be specified or known in advance and tends to fluctuate over time. Examples of such environments are streaming video over the Internet or mobile wireless networks. Scalable video systems allow for some video quality to be traded-off in exchange for improved service continuity when capacity is constrained. This thesis introduces “stream morphing”, a new approach to the scalable video problem. Traditional systems, such as those described in the MPEG standards, create enhancement layers by measuring and coding the quantization error from the previous layer. In contrast, stream morphing works on a hierarchy of single-layer bitstreams of increasing quality. The enhancement layers of the scalable system encode the operations required to transform a single-layer bitstream that has already been received into another bitstream which is of higher quality. This process allows for the set of single-layer bitstreams to be simulcast in an efficient manner. The ability to move easily between a series of single-layer bitstreams and a scalable representation of the same video sequence is a significant new functionality that is not present in other scalable coding systems. The bandwidth efficiency of the new technique is also higher than the methods for SNR scalability defined in MPEG-2 and MPEG-4. This is confirmed by subjective test results over a range of different types of video sequence. In addition to implementation details and results for SNR scalability under VBR conditions, this thesis also develops a CBR rate control algorithm for the SNR scalable case and extends the stream morphing concept to cover spatial and temporal scalability.
Contents

1 Introduction 19

2 Overview of Video Compression and MPEG-4 23
   2.1 History ....................................................... 23
   2.2 Definitions ............................................... 24
   2.3 Video Compression ........................................ 26
      2.3.1 Input Colourspace .................................. 27
      2.3.2 Spatial Correlation .................................. 28
      2.3.3 Quantization .......................................... 32
      2.3.4 Temporal Correlation ................................. 37
      2.3.5 Entropy Coding ...................................... 44
   2.4 Overview of MPEG-4 Syntax .............................. 45
      2.4.1 The Macroblock Header ............................. 46
      2.4.2 Error Recovery ...................................... 48
   2.5 Overall Structure ........................................ 49
   2.6 Rate Control ............................................... 51
   2.7 Conclusion ................................................ 53

3 Scalable Video Techniques 54
   3.1 Introduction ............................................... 54
   3.2 MPEG-2 SNR Scalability ................................. 56
      3.2.1 Drift .................................................. 62
   3.3 Temporal and Spatial Scalability ...................... 65
3.3.1 MPEG-4 Generalized Scalability ........................................ 65
3.3.2 Changes from MPEG-2 ..................................................... 71
3.3.3 SNR Scalability in MPEG-4 .............................................. 72
3.4 MPEG-4 Fine Granularity Scalability ................................. 73
  3.4.1 MPEG-4 FGS Applications ............................................. 79
3.5 Experimental Procedure .................................................. 80
  3.5.1 Objective Tests ......................................................... 81
  3.5.2 Subjective Tests ........................................................ 83
3.6 Results ................................................................................... 87
3.7 Discussion ............................................................................. 96
  3.7.1 Critique of FGS ............................................................. 96
  3.7.2 Critique of Multi-Loop SNR Scalability ............................... 99
  3.7.3 Comments on Performance of Scalable Video Coders ........ 100
3.8 Other Approaches to Scalable Video .................................. 104
  3.8.1 Conditional Replenishment ............................................ 104
  3.8.2 Bitstream Decomposition .............................................. 105
  3.8.3 Multiple Description Coding ......................................... 108
  3.8.4 3D Subband Coding .................................................... 110
3.9 Conclusion ............................................................................ 112

4 Stream Morphing ................................................................. 113
  4.1 Introduction ....................................................................... 113
  4.2 Overview of Stream Morphing .......................................... 114
    4.2.1 Arithmetic Coding ...................................................... 119
  4.3 Morphing of Bitstream Components .................................. 124
    4.3.1 Quantization Step Size ............................................. 125
    4.3.2 Macroblock Mode ..................................................... 127
    4.3.3 Skipped Macroblocks ................................................ 128
    4.3.4 Intra AC Coefficient Prediction .................................. 129
    4.3.5 Signalling of Coded Blocks ....................................... 130
CONTENTS

4.3.6 Morphing of DCT Coefficients .................................. 130
4.3.7 Layer Separation .................................................. 138
4.4 Applications ......................................................... 141
4.5 Computational Complexity ........................................... 144
  4.5.1 Methodology ...................................................... 144
  4.5.2 Coding and Decoding of Symbols ................................. 146
  4.5.3 Per Macroblock Operations ...................................... 146
  4.5.4 Intra Coefficients ................................................ 146
  4.5.5 Inter Coefficients ............................................... 147
  4.5.6 Memory Requirements ........................................... 147
  4.5.7 Optimizations .................................................... 149
4.6 Experimental Procedure ............................................. 150
4.7 Results ........................................................................ 151
4.8 Discussion ................................................................... 170
4.9 Conclusion .................................................................. 172

5 Rate Control for Stream Morphing and MPEG-2-Compliant SNR Scalability 173
  5.1 Introduction ............................................................. 173
  5.2 Rate Control for Single-layer MPEG-4 Encoders .................... 174
    5.2.1 Previous Approaches .............................................. 174
    5.2.2 Encoder Architecture ............................................ 180
    5.2.3 Bit Production Model ............................................. 180
    5.2.4 Perceptual Quantization ......................................... 190
    5.2.5 Thresholding ....................................................... 192
    5.2.6 Frame-Level Rate Control ....................................... 193
    5.2.7 Monitoring of the Coding Process ............................... 194
    5.2.8 Intra-Coded Frames ............................................. 196
    5.2.9 Algorithm .......................................................... 196
  5.3 Rate Control for MPEG-2-Compliant SNR Scalability ............... 198
List of Figures

2.1 Video system overview .............................................. 25
2.2 Chrominance subsampling modes ............................. 28
2.3 DCT basis functions .................................................. 30
2.4 DC and AC coefficient prediction ............................... 31
2.5 Effect of quantization in spatial and DCT domains ............... 33
2.6 Zig-zag scanning of quantized coefficients .................. 36
2.7 Motion-compensated prediction .................................. 37
2.8 Motion vector prediction ........................................... 39
2.9 MPEG frame types ................................................... 41
2.10 MPEG-4 syntax flowchart ......................................... 46
2.11 MPEG-4 single-layer encoder ................................. 50
2.12 MPEG-4 single-layer decoder ................................. 50

3.1 MPEG-2 SNR Scalable decoder ................................. 57
3.2 Single-loop MPEG-2 SNR Scalable encoder (with drift) ........... 58
3.3 Two-loop MPEG-2 SNR scalable encoder ...................... 60
3.4 Two-loop MPEG-2 SNR scalable encoder (pyramid) .......... 61
3.5 Single-loop SNR Scalable encoder (requires non-MPEG-2-compliant decoder) .................................................. 64
3.6 Decoder corresponding to drift-free encoder of Figure 3.5 ........... 65
3.7 MPEG-4 decoder for generalized scalability .................. 66
3.8 References for prediction in MPEG-4 enhancement layer P-VOPs
(corresponds to Table 3.1) ......................................... 67
3.9 References for prediction in MPEG-4 enhancement layer B-VOPs
(corresponds to Table 3.2) ......................................... 68
3.10 Prediction for MPEG-4 spatial scalability .......................... 69
3.11 Prediction for MPEG-4 temporal scalability ...................... 70
3.12 Formation of prediction for MPEG-2 spatial scalability ........ 71
3.13 MPEG-4 Fine Granularity Scalability encoder .................... 73
3.14 MPEG-4 Fine Granularity Scalability decoder ................... 75
3.15 Frequency weighting for MPEG-4 FGS ........................... 76
3.16 Selective enhancement for MPEG-4 FGS .......................... 76
3.17 Progressive Fine Granularity Scalability encoder ............... 78
3.18 Partial scoresheet for subjective tests ............................ 84
3.19 Constant quantizer results for “Mother & Daughter” (CIF 10fps) 88
3.20 Constant quantizer results for “Foreman” (CIF 10fps) ......... 89
3.21 Loss due to drift (Mother & Daughter) ......................... 93
3.22 Loss due to drift (Foreman) .................................... 94
3.23 Subjective comparison of MPEG-2 SNR Scalability and FGS (95%
confidence intervals) ............................................. 95
3.24 Typical single-layer rate-distortion profile ...................... 103
3.25 Two-layer conditional replenishment encoder .................... 106
3.26 Two-layer conditional replenishment decoder .................... 107
3.27 3D Subband decomposition ..................................... 110
4.1 Stream morphing encoder ........................................ 115
4.2 Morphing between macroblocks in successive layers ............ 116
4.3 Stream morphing decoder ....................................... 118
4.4 Arithmetic coding example ...................................... 120
4.5 Arithmetic coder renormalization ................................. 122
4.6 Efficient quantizer step size variation ............................ 125
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.7</td>
<td>Clipping of quantizer step size at low rates in MPEG-4</td>
<td>126</td>
</tr>
<tr>
<td>4.8</td>
<td>Surrounding causal macroblocks</td>
<td>129</td>
</tr>
<tr>
<td>4.9</td>
<td>Coding of intra coefficients</td>
<td>132</td>
</tr>
<tr>
<td>4.10</td>
<td>Effect on region transition location on symbol statistics</td>
<td>134</td>
</tr>
<tr>
<td>4.11</td>
<td>Stream morphing as encoder post-processing</td>
<td>141</td>
</tr>
<tr>
<td>4.12</td>
<td>Upstream recovery for single-layer switching</td>
<td>142</td>
</tr>
<tr>
<td>4.13</td>
<td>Constant quantizer results for “Akiyo” (CIF 10fps)</td>
<td>153</td>
</tr>
<tr>
<td>4.14</td>
<td>Constant quantizer results for “Carphone” (CIF 10fps)</td>
<td>154</td>
</tr>
<tr>
<td>4.15</td>
<td>Constant quantizer results for “Coastguard” (CIF 10fps)</td>
<td>155</td>
</tr>
<tr>
<td>4.16</td>
<td>Constant quantizer results for “Foreman” (CIF 10fps)</td>
<td>156</td>
</tr>
<tr>
<td>4.17</td>
<td>Constant quantizer results for “Mother &amp; Daughter” (CIF 10fps)</td>
<td>157</td>
</tr>
<tr>
<td>4.18</td>
<td>Constant quantizer results for “Foreman” (CIF 30fps)</td>
<td>158</td>
</tr>
<tr>
<td>4.19</td>
<td>Constant quantizer results for “Mother &amp; Daughter” (CIF 30fps)</td>
<td>159</td>
</tr>
<tr>
<td>4.20</td>
<td>Constant quantizer results for “Foreman” (QCIF 30fps)</td>
<td>160</td>
</tr>
<tr>
<td>4.21</td>
<td>Constant quantizer results for “Mother &amp; Daughter” (QCIF 30fps)</td>
<td>161</td>
</tr>
<tr>
<td>4.22</td>
<td>Constant quantizer results for “Foreman” (720x576 25fps)</td>
<td>162</td>
</tr>
<tr>
<td>4.23</td>
<td>Constant quantizer results for “Mother &amp; Daughter” (720x480 30fps)</td>
<td>163</td>
</tr>
<tr>
<td>4.24</td>
<td>Loss due to drift (Foreman)</td>
<td>164</td>
</tr>
<tr>
<td>4.25</td>
<td>Loss due to drift (Mother &amp; Daughter)</td>
<td>164</td>
</tr>
<tr>
<td>5.1</td>
<td>TM5 rate control: example of non-uniform bit allocation at low rates</td>
<td>176</td>
</tr>
<tr>
<td>5.2</td>
<td>Modified single-layer encoder with rate control</td>
<td>179</td>
</tr>
<tr>
<td>5.3</td>
<td>Non-zero motion vector bits</td>
<td>181</td>
</tr>
<tr>
<td>5.4</td>
<td>Bits for macroblock overhead</td>
<td>181</td>
</tr>
<tr>
<td>5.5</td>
<td>Bits for DCT coefficients</td>
<td>182</td>
</tr>
<tr>
<td>5.6</td>
<td>Number of non-zero coefficients</td>
<td>184</td>
</tr>
<tr>
<td>5.7</td>
<td>Bits per non-zero intra coefficient</td>
<td>184</td>
</tr>
<tr>
<td>5.8</td>
<td>Bits per non-zero inter coefficient</td>
<td>185</td>
</tr>
<tr>
<td>5.9</td>
<td>Number of coded macroblocks</td>
<td>186</td>
</tr>
<tr>
<td>5.10</td>
<td>Overhead per intra macroblock</td>
<td>186</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

5.11 Overhead per inter macroblock ........................................ 187
5.12 Stream morphing: number of bits required to code new coefficients
("Foreman", 1st enhancement layer of Figure 4.16) ...................... 202
5.13 Stream morphing: number of bits required to code existing coefficients ("Foreman", 1st enhancement layer of Figure 4.16) ....... 205
5.14 Stream morphing: coded block pattern overhead ("Foreman", 1st enhancement layer of Figure 4.16) .................................. 207
5.15 Stream morphing: miscellaneous syntax (pass through) ("Foreman", 1st enhancement layer of Figure 4.16) ................................ 208
5.16 Rate control results for "Foreman" (CIF 10fps) ....................... 213
5.17 Bits used (cumulative) for discrete scalable encoders ("Foreman", CIF 10fps) ................................................................. 214
5.18 Rate control results for "Mother & Daughter" (CIF 10fps) ........ 215
5.19 Rate control results for "Foreman" (CIF 30fps) ....................... 216
5.20 Rate control results for "Mother & Daughter" (CIF 30fps) ........ 217
5.21 Rate control results for "Foreman" (QCIF 30fps) .................... 218
5.22 Rate control results for "Mother & Daughter" (QCIF 30fps) ...... 219
5.23 Subjective comparisons with stream morphing (95% confidence intervals) ................................................................. 220

6.1 Prediction for MPEG-4 spatial scalability (reproduction of Figure 3.10) .............................................................. 227
6.2 Prediction for stream morphing spatial scalability .................... 228
6.3 Prediction for MPEG-4 temporal scalability (reproduction of Figure 3.11) ................................................................. 234
6.4 Interleaving temporal scalable layers to form a single layer ....... 234
6.5 Prediction references for temporal scalability (IPP GOP structure, "causal" configuration) ................................................... 236
6.6 Spatial scalability results for "Foreman" ................................ 242
6.7 Spatial scalability results for "Mother & Daughter" ................... 243
6.8 Temporal scalability results for "Foreman" ............................. 244
6.9 Temporal scalability results for "Mother & Daughter" 245
7.1 Suggested method for more efficient scanning 253
A.1 Stream morphing syntax 267
B.1 Example frames from sequences (CIF 10fps) 289
List of Tables

2.1 DC scaler calculation for H.263 quantization .......................... 34
2.2 Scaled quantizer values for MPEG quantization (MPEG-4) .......... 35
2.3 MPEG-4 VLC table for AC luminance coefficients (part) ............ 44

3.1 References for prediction in MPEG-4 enhancement layer P-VOPs . 66
3.2 References for prediction in MPEG-4 enhancement layer B-VOPs . 67
3.3 Subjective test conditions .................................................. 86
3.4 Bitstream statistics for Foreman (CIF 10fps) .......................... 90
3.5 Bitstream statistics for Mother & Daughter (CIF 10fps) ............. 91
3.6 Spread of subjective test results ......................................... 94
3.7 Confidence that MPEG-2-compliant SNR scalability (reference) is subjectively better for the average user ......................... 94
3.8 FGS bit-plane spacing ....................................................... 98

4.1 Symbol probabilities for arithmetic coding example .................. 121
4.2 Probability model selection for intra coefficients (two quantization regions) ............................................................. 134
4.3 Probability model selection for intra coefficients (three quantiza-
tion regions) ....................................................................... 135
4.4 Probability model selection for coefficient-oriented symbols ....... 139
4.5 Probability model selection for block-oriented symbols ............. 140
4.6 Operation summary for stream morphing ............................... 148
4.7 Operation summary with fixed-point implementation of division
   with restricted divisors ........................................... 150
4.8 Operation counts for Foreman (CIF 10fps) .......................... 165
4.9 Operation counts for Mother & Daughter (CIF 10fps) .............. 166
4.10 Overall operation summary for Foreman (4 enhancement layers,
   CIF 10fps) ............................................................... 166
4.11 Overall operation summary for Mother & Daughter (4 enhancement
   layers, CIF 10fps) ..................................................... 167
4.12 Bitstream statistics for Foreman (CIF 10fps) ........................ 168
4.13 Bitstream statistics for Mother & Daughter (CIF 10fps) .......... 169

5.1 MPEG-4 fixed overhead for inter macroblocks with no non-zero
   DCT coefficients ................................................... 188
5.2 Spread of subjective test results .................................. 222
5.3 Probability of average user finding stream morphing (reference
   condition) worse than other tested technique ..................... 222

A.1 Cross-reference between syntax and content in Chapters 4, 6 ... 266
A.2 Model type mb_coded_prev_skipped .................................. 280
A.3 Model type block_coded_prev_skipped ............................... 280
A.4 Model type num_new_coeffs_intra ................................ 280
A.5 Model type num_new_coeffs_inter ................................ 280
A.6 Model type intra_ac_pred ........................................... 280
A.7 Model type change_mode ............................................. 281
A.8 Model type inter_existing_coeff ................................. 281
A.9 Model type inter_restored ......................................... 281
A.10 Model type refine_contract_escape ............................... 281
A.11 Model type inter_newcoeff_type ................................. 282
A.12 Model type dquant ................................................ 282
A.13 Model type newcoeff_run ........................................ 282
A.14 Model type intra_coeff_mag2 ................................. 282
A.15 Model type intra_coeff_mag3 .................................. 283
A.16 Model type intra_coeff_mag4 .................................. 283
A.17 Model type spatial16_mvdiff ................................. 283
A.18 Model type mv_diff_abs ....................................... 283
A.19 Model type temporal_mvdiff_sign .......................... 284
A.20 Model type temporal_existing_diff ......................... 284
A.21 Model type temporal_existing_mag ........................ 284
A.22 Model type temporal_nextframe8x8 ........................ 284
A.23 Model type temporal_nextframe16x16 ....................... 285
A.24 Model type sign ............................................... 285
A.25 Model type raw_bits ......................................... 285
A.26 Complete model list ........................................... 286

C.1 Source sequences for subjective tests ......................... 295
C.2 Subjective test presentations ................................ 296
C.3 Standard deviation of subjective difference scores for each observer 297
C.4 Confidence that MPEG-2-compliant SNR scalability (reference) is subjectively better for average user (alternate method) .... 298
C.5 Probability of average user finding stream morphing (reference condition) worse than other tested technique. (alternate method) 298
Chapter 1

Introduction

The use of the techniques of digital video compression in a particular application scenario demands careful consideration of the properties of the environment in which the system is to operate. There is no single approach that is optimum under all possible conditions. The widely-used MPEG-1 [1] and MPEG-2 [2] multimedia compression standards were primarily designed for deployment in controlled, predictable environments. Some of the most popular transport media for MPEG-1 and MPEG-2 content are optical disks, terrestrial broadcast, cable and satellite. A common feature of these environments is that there is dedicated bandwidth set aside for exclusive use by the individual content stream (or streams) which is largely static over time and unaffected by external influences. Furthermore, none of these applications are normally subject to “hard” real-time constraints and can therefore introduce small processing delays without adversely affecting the experience of the end user. This allows for some degree of flexibility when used with transport media that provide adequate bandwidth but cannot guarantee error-free operation. Optical disk is a good example of such a medium: identical copies of a given block of data are stored at various places on a disk so that if one area is affected by a scratch on the surface of the disk the data can be accessed again a short time later on another part of the disk that is likely to be readable without error.

A number of emerging applications, the most prominent being video over the Internet and mobile wireless networks, exist in more dynamic environments.
Introduction

Data error rates and the amount of available channel capacity are constantly changing due to factors such as Internet traffic congestion or fading in a wireless channel. The desire to reduce the end-to-end latency of a system may prevent the use of solutions such as the transmitter switching to another lower (or higher) bit rate representation of the content when conditions change. In such cases a different approach is required. *Scalable* (or *layered*) video compression schemes partition some notional amount of channel bandwidth that is only available in the real system for some small fraction of the time when conditions are favourable. This layering is done in such a way that, when combined with appropriate support from the transport medium, a reduction in the available bandwidth results in a reduced number of layers arriving at the receiver that can still be decoded and a mildly degraded service being presented to the user. This is in contrast to traditional non-scalable (also known as *single-layer*) systems that do not function correctly, or exhibit unacceptable loss of quality for significant periods of time under these types of conditions. Bandwidth adaptability in scalable systems should be largely automatic and not require constant feedback to the encoder and/or transmitter about the state of the communications channel as this will increase end-to-end latency. Scalable video coding has been a topic of research for some years and a number of techniques have been incorporated in the MPEG-2 and MPEG-4 [3] standards.

The primary original contribution in this thesis is the development of the technique known as *stream morphing*, which is a new method for creating a scalable compressed representation of a video sequence. Stream morphing will be shown to have a number of advantages over existing scalable coding techniques such as those in the MPEG standards:

- Stream morphing is *universal* in that the same basic concept can be used in systems containing different frame sizes and rates in each layer.

- A scalable representation of a video sequence can be generated from a set of standard single-layer descriptions at low computational cost. Similarly,
the original set of single-layer descriptions can be recovered without full transcoding. This is an important functionality that cannot be achieved using previous techniques.

- Its coding efficiency is no worse, and is in some cases significantly better than the techniques in the MPEG standards.

- It does not suffer from any additional loss of subjective quality due to repeated quantization of the input signal as is the case with some previous techniques.

A new rate control algorithm is also described that can be applied to stream morphing and MPEG-2 SNR scalability as well as conventional single-layer MPEG-4.

The thesis is structured as follows:

Chapter 2 provides an introduction to the key ideas behind video compression as well as an overview of a number of the key internal details of the MPEG-4 Video standard. While the concept of stream morphing itself is not tied to MPEG-4, or indeed to any particular video coding standard, the details of any specific implementation do depend heavily on the workings of the system chosen. The experimental stream morphing system that has been constructed for this thesis is based on MPEG-4 and the later chapters will make extensive reference to the description of single-layer MPEG-4 in this chapter.

Chapter 3 describes some of the previous approaches to scalable video. These systems can be classified into three main types: SNR scalable systems use the same frame size and rate in consecutive layers, spatial scalability refers to the use of different spatial resolutions in different layers while temporal scalability fixes the frame size and changes the frame rate between layers. Combinations of these three basic types are also possible in most cases. This chapter focuses on MPEG-2 SNR scalability and the technique called Fine Granularity Scalability which is defined in the MPEG-4 standard and is also primarily de-
signed for SNR scalability. These are the two main existing techniques which currently target the same type of applications as stream morphing and will be used as the basis for the later performance comparisons. As well as pointing out the respective strengths and weaknesses of these two techniques some other solutions to the scalable video problem will be discussed later in the chapter.

Stream morphing itself is described in detail in Chapter 4. Along with a description of the morphing process itself for the case of SNR scalability and performance comparisons with other techniques under conditions of constant quantization, this chapter describes some of the application scenarios to which stream morphing can be applied and provides an analysis of its computational complexity.

Chapter 5 develops a rate control algorithm which can be applied not only to stream morphing but also to standard single-layer MPEG-4 and MPEG-2 SNR scalability with minimal modifications. Using this algorithm the subjective performance of stream morphing is compared to MPEG-2 SNR scalability and MPEG-4 Fine Granularity Scalability.

Chapter 6 applies the basic concept of stream morphing, which has been discussed in previous chapters only in the context of SNR scalability, and extends it to cover temporal and spatial scalability. These developments provide stream morphing with the same level of flexibility as the existing techniques in MPEG-2 and MPEG-4 which offer all three basic types of scalability.

Finally, Chapter 7 summarizes the conclusions of this thesis and outlines some directions for future development of stream morphing.
Chapter 2
Overview of Video Compression and MPEG-4

2.1 History

MPEG-4 is the third in the series of multimedia standards [4] produced by the Moving Picture Experts Group (MPEG) [5]:

MPEG-1 Completed in 1991 [1] and optimized for use with digital storage devices such as Compact Disc, Digital Audio Tape (DAT) and optical disk drives. Typical bit rates are 1-1.5Mb/s.

MPEG-2 (1994 [2]) is targeted at applications that require higher bit rates (typically around 4Mb/s) than MPEG-1 and is used in applications such as digital television and DVD Video. Significant new features in MPEG-2 not present in MPEG-1 include support for interlaced video, measures to provide resilience to errors introduced into the bitstream and scalability.

MPEG-4 covers much of the same ground as the earlier standards while adding composition of presentations from a number discrete objects which will enable new forms of interactive applications. Efforts were also made to deal with delivery of services over packet networks and the Internet which are normally at much lower bit rates than MPEG-1. Here MPEG's work converges with that of the standards group of the ITU which had devel-
2.2 Definitions

oped a number of low bit rate video coding standards for teleconferencing [6]. There is much similarity between MPEG-4 video and the ITU H.263 standard [7]. Indeed, there is an optional compatibility mode in MPEG-4 which allows decoders to work with H.263 bitstreams in addition to MPEG-4 bitstreams. Version 1 of the MPEG-4 standard was ratified in December 1998 [3], Version 2 was completed in December 1999 for ratification in 2001.

In this thesis we are concerned only with the video portions of these standards; in addition MPEG standardizes audio compression and systems issues (packetization of data, multiplexing and synchronization of the various components of a complete presentation etc.). Some of the new features present in MPEG-4 video, such as object-based coding and still texture coding are not relevant to the main content of this thesis and will not be discussed. The focus will be on rectangular, non-interlaced video for single-layer and scalable video systems.

It is important to note that the MPEG standards themselves only specify the structure of the coded representation of a given sequence and the behaviour of the decoder. There are operations in the encoder, notably motion estimation (Section 2.3.4.2) and rate control (Section 2.6), which could be implemented in many different ways but which do not require changes to the encoded representation or for the decoder to be modified in any way. Such operations do not need to be discussed in the standards and are described as non-normative.

2.2 Definitions

Figure 2.1 gives a high-level description of the types of systems we discuss in this thesis. There is a video encoder which takes a constant stream of frames each of the same size and spaced at regular intervals in time. From this input the encoder produces one or more streams of bits, bitstreams for short, which
2.2 Definitions

are themselves usually split into short packets that then map onto the transport or storage medium being used. Normally, all of these bitstreams are then sent out over some channel, although some systems only use a subset of these at any one time. Video may be multiplexed along with other data, such as audio, to be separated again at a decoder. Transport over certain types of channel may introduce various kinds of errors into the bitstreams, ranging from simple bit errors to the loss of packets or even entire streams. The decoder takes the data from the streams it receives and reconstructs a video sequence from this for presentation to the user.

The case where the encoder produces a single bitstream from a sequence of video is called single-layer video. Not all delivery systems can make a guarantee that all data generated by the encoder will be delivered in a timely fashion to the decoder. Even for systems that can provide such a guarantee, many users will not be prepared to pay the extra cost associated with a "perfect" system. Perhaps the best example of this is the Internet, where congestion can cause significant data loss for real-time applications where packet re-transmission is not possible. In such scenarios, it is often advantageous to create multiple streams so that if one or more stream is lost or corrupted but at least one
is received intact then the system will continue to work, albeit in a degraded manner. The more streams received, the better the perceived "quality" of the video will be. Such systems are referred to as scalable in that quality scales with the amount of data received. Chapter 3 discusses approaches to this in more detail while the remainder of this chapter focuses on the single-layer case.

Note: Since MPEG-4 allows for a scene to be composed from multiple Video Objects (VOs) that are combined at the decoder, the term "frame" is generally not used when referring to an individual piece of video but rather the term Video Object Plane (VOP) is used. Since this thesis is only concerned with single video objects, we will use the terms "VOP" and "frame" interchangeably since for our purposes there is always a one-to-one correspondence.

2.3 Video Compression

While it is possible to code video such that the output from the decoder is exactly the same as the original input to the encoder, bandwidth constraints normally rule this out and we must be prepared to (selectively) degrade the quality of the service in order to reduce the volume of data generated. As such, some techniques that are applied to video are known as lossy because they cause some information to be irreversibly destroyed whereas others are lossless transformations that can be completely reversed at the decoder.

The following sections describe the key methods behind Discrete Cosine Transform (DCT) video coders with block-based motion-compensated prediction which is used in all the current MPEG video standards. The key features exploited to produce compression are:

- Human visual system insensitivity to chrominance relative to luminance,
- Spatial correlation: "natural" images are inherently low-pass,
- Some quantization of image data, especially at high spatial frequencies, is
tolerable,

- Temporal correlation: successive frames of video are often very similar,

- Entropy coding takes symbol statistics into account when generating the final bitstream.

### 2.3.1 Input Colourspace

While capture and display of analog video is largely done in the RGB or similar colourspace for digital video the standard practice is to transform to the YUV (or similar) colourspace. The Y component is known as the *luminance* and corresponds to the brightness-only signal that is seen on a black-and-white television. The U and V components are known as *chrominance* and describe colour information only. The transformation from RGB to YUV is linear [8]:

\[
Y = 0.257 \times R + 0.504 \times G + 0.098 \times B + 16 \tag{2.1}
\]

\[
U = -0.148 \times R - 0.291 \times G + 0.439 \times B + 128 \tag{2.2}
\]

\[
V = 0.439 \times R - 0.368 \times G - 0.071 \times B + 128 \tag{2.3}
\]

where [R,G,B] are in the range [0,1] and [Y,U,V] are 8-bit integer-valued samples, although note that this conversion does not use the complete range [0,256) but allows some buffer space at the end of each range for subsequent processing.

Studies of the human visual system have shown that it is less sensitive to degradation of chrominance information than luminance [9] so the standard practice in digital video is to subsample the chrominance information that is input into a video encoder. Figure 2.2 details two common subsampling schemes and the positions for the chrominance samples relative the luminance samples. 4:2:0 uses only one quarter the number of chrominance samples as the luminance samples. For applications that require higher accuracy, such as video editing, the 4:2:2 scheme has double the number of chrominance samples as 4:2:0. Note that not all video standards use identical positioning for the chrominance samples,
2.3 Video Compression

Figure 2.2: Chrominance subsampling modes

e.g. MPEG-2 4:2:0 aligns the chrominance samples horizontally with one of the columns of luminance pixels unlike the MPEG-4 scheme shown in Figure 2.2(a).

Subsampling of chrominance is a \textit{lossy} transformation.

2.3.2 Spatial Correlation

The MPEG standards are geared towards the coding of \textit{natural} images that exhibit smooth variation over most of the image area with relatively few sharp discontinuities. This is a reasonable assumption for much of the content we might want to encode. However, for material such as line art or cartoons, with many sharp edges, this will require higher bit rates to encode to a similar quality as other methods that are optimized towards these other types of images. A similar situation exists with still image standards (e.g. for use on the World Wide Web) where JPEG compression (Joint Photographic Experts Group, a still image compression standard which like the MPEG video standards is based on the DCT) is used for natural images but the Graphics Interchange Format (GIF) or Portable Network Graphics (PNG) formats are more appropriate for “artificial” content [10].

A “smooth” image contains less high frequency energy than low frequency
2.3 Video Compression

energy. To exploit this, the MPEG standards apply the Discrete Cosine Transform (DCT) to blocks of input image data. The DCT itself is a reversible, lossless transformation and as such does not perform compression, however, as will be seen in the next section, it allows for significantly heavier quantization to be performed for a given level of subjective distortion which, in turn, reduces the number of bits required to code the image. The two-dimensional $N$-point forward DCT that transforms from the spatial domain pixels $f(x, y)$ to frequency domain coefficients $F(u, v)$ can be expressed as:

$$ F(u, v) = \frac{2}{N} C(u)C(v) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x, y) \cos \left( \frac{u\pi(x + 1/2)}{N} \right) \cos \left( \frac{v\pi(y + 1/2)}{N} \right) $$

(2.4)

where

$$ C(\alpha) = \begin{cases} 
\frac{1}{\sqrt{2}} & \text{for } \alpha = 0 \\
1 & \text{otherwise}
\end{cases} $$

The inverse DCT (IDCT) which recovers the original pixels is given by:

$$ f(x, y) = \frac{2}{N} \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} C(u)C(v)F(u, v) \cos \left( \frac{u\pi(x + 1/2)}{N} \right) \cos \left( \frac{v\pi(y + 1/2)}{N} \right) $$

(2.5)

The two-dimensional DCT and IDCT are separable transforms and can be alternatively expressed as a one-dimensional DCT performed on each row of pixels/coefficients followed by the same operation performed on the columns of that result.

The standard video coding algorithms choose the block size $N = 8$, henceforth known as the 8x8 DCT and IDCT, as a trade-off between compression efficiency and computational complexity. It is important to appreciate that performing the DCT or IDCT on video sequences in real time requires a significant amount of computational power which until recently was only available in expensive custom hardware and which may be still an issue for small footprint applications such as decoders embedded in battery-powered mobile devices. Indeed there are other transforms such as the Karhunen-Loève Transform (KLT) [11] which are better than the DCT at compacting coefficients into
the low-frequency "band" but are not used because of still higher computational requirements. Another technique that has been used to reduce spatial correlation is the wavelet transform [12] which is now commonly used for still image compression. Its use in video applications has not been widespread to date, however, due to poor interactions with other techniques used in video to reduce temporal redundancy (to be discussed in Section 2.3.4 below) however this is a topic of a great deal of ongoing research [13].

Figure 2.3 shows the 64 basis functions which are the cosine terms that are common to the forward (equation 2.4) and inverse 8x8 DCTs (equation 2.5). The term in the top left corner where \( u \) and \( v \) are zero is constant and is known as the DC coefficient. The other 63 coefficients are called the AC coefficients. As we go down through the rows in the figure the value of \( u \) increases while \( v \) increases the further we go the right. Natural images, when expressed as a weighted sum over these basis functions, tend to have only small contributions from those high frequency basis functions towards the bottom right of the figure, which can be seen to undergo significant changes over small distances. This corresponds to small values for the transform coefficients \( F(u, v) \) where \( u \) and/or \( v \) are large.
The lowest frequency components in an image generally change slowly over the frame and as such their values tend to be similar across adjacent 8x8 blocks. Increasing the DCT block size would reduce this inter-block redundancy, however this would come at the cost of significant extra computation. MPEG-4 seeks to exploit this redundancy while keeping the 8x8 block size by differentially coding the DC coefficient and, optionally, the first row or column of DCT coefficients in each block. Rather than coding such coefficients independently the difference between the coefficient value and the corresponding coefficient in an adjacent block is calculated. Since the coefficient values are likely to be similar, this differential generally has a small range and thus require fewer bits to code. Figure 2.4 shows DC coefficient prediction (dark grey squares) for block X from either block A or block C and the optional AC prediction (light grey squares) is either the first column of X being predicted from the first column of A or the first row of X is predicted from the first row of C. If \( F_A \) is the DC coefficient value in A, \( F_C \) is the DC value in C and \( F_B \) is the DC value in the diagonally adjacent block B the prediction is made from A if \( |F_A - F_B| \geq |F_B - F_C| \) otherwise C is used. Note that neither DC nor AC coefficient prediction is used in
2.3 Video Compression

conjunction with any temporal prediction (Section 2.3.4) given that both prediction types have largely the same effect and as such there is no advantage in performing both. Similar techniques are used in other compression standards such as MPEG-2 and JPEG.

2.3.3 Quantization

If we must resort to lossy coding of images because we cannot achieve our bandwidth targets through lossless techniques alone, one obvious strategy is to decrease the precision of the original pixels (spatial domain) or transform domain coefficients that we are using to construct our bitstream. Doing this in the spatial domain is not very effective since a reduction of one or two bits per pixel in precision of pixel values tends to produce very noticeable artifacts in the image where distinct regions all have the same value instead of being continuously varying as in the original. If we perform the same operation on a DCT-domain representation of the image the result is quite different: many high-frequency coefficients which were close to zero can be forced to zero with only moderate amounts of quantization.

It is also known that the human visual system is relatively insensitive to the introduction of high-frequency noise and so the loss of some coefficients is often not subjectively perceptible [9]. Heavy quantization of an image and the loss of high-frequency information results in blurring and in extreme cases "blocking" artifacts are visible where two adjacent blocks have a large discontinuity at their common edge. At sharp edges inside blocks loss of high frequency components due to quantization introduces "ringing" artifacts. This is the same as the Gibb's phenomenon which is well known in the context of the Fourier Transform [14].

Figure 2.5 shows the effect of spatial and DCT-domain quantization on the luminance component of the first frame of the MPEG test sequence "Foreman". Panel (a) shows the original frame. Panel (b) shows the effect of rounding off each pixel value to the nearest multiple of 16 which equates to the loss of the
Figure 2.5: Effect of quantization in spatial and DCT domains

four least-significant bits of information (the original pixels have 8-bit precision). Panel (c) shows the effect of the same loss of precision on the DCT coefficients of the image which can be seen to make very little difference to the image subjectively, unlike quantization in the spatial domain. Panel (d) also shows DCT-domain quantization which rounds off to the nearest multiple of 64 which eliminates almost all the AC coefficients and shows significant blocking artifacts.

MPEG-4 allows for the use of two different types of quantization: the first is the uniform quantizer defined in H.263, henceforth known as "H.263 Quanti-
2.3 Video Compression

The degree of quantization is specified by the Quantization Parameter (QP), also known as the quantizer step size. For this method the quantized coefficient value $F'(u, v)$ is calculated by

$$F'(u, v) = \begin{cases} \frac{((2 \times F(u, v)/s + 1)/2) \times s)}{s} & \text{for DC coefficients} \\ \text{sgn}(F(u, v)) \times (|F(u, v)|/(2QP)) & \text{for AC coefficients} \end{cases}$$

where $s$ is the "DC scaler" that is calculated from QP as shown in Table 2.1. The division operator $/$ in equation 2.6 is integer division where the result is rounded off to the integer value closest to zero. This division is the "lossy" part of the quantization operation.

Note that only the inverse quantization operation, which we do not show here, is normative in the MPEG standards. Implementers need not use the exact form of equation 2.6 but may choose some other, possibly less accurate, form.

The second quantization algorithm is the non-uniform quantizer from MPEG-2 which we shall refer to from now on as "MPEG Quantization". Non-uniform quantizers of this type are designed to quantize the high frequency coefficients more heavily than the low frequency coefficients since, as stated previously, the human visual system is less sensitive to heavy quantization of high frequency information. For MPEG quantization in MPEG-4, DC coefficients are processed in the same way as the H.263 quantization. AC coefficients are dequantized using the following expression:

$$F''(u, v) = ((2 \times F'(u, v) + k) \times W_n(u, v) \times \text{quantizer_scale})/32$$

where $W_n(u, v)$ is an 8x8 weighting matrix to emphasize high frequency components which have been more heavily quantized than low frequencies, $k$ is zero.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Luminance</td>
<td>8</td>
<td>2QP</td>
<td>QP + 8</td>
<td>2QP - 16</td>
</tr>
<tr>
<td>Chrominance</td>
<td>8</td>
<td>$\frac{QP+13}{2}$</td>
<td>QP - 6</td>
<td></td>
</tr>
</tbody>
</table>

Table 2.1: DC scaler calculation for H.263 quantization.
2.3 Video Compression

<table>
<thead>
<tr>
<th>QP</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>quantizer_scale</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>11</td>
<td>12</td>
</tr>
<tr>
<td>QP</td>
<td>17</td>
<td>18</td>
<td>19</td>
<td>20</td>
<td>21</td>
<td>22</td>
<td>23</td>
<td>24</td>
<td>25</td>
<td>26</td>
<td>27</td>
<td>28</td>
<td>29</td>
<td>30</td>
<td>31</td>
</tr>
<tr>
<td>quantizer_scale</td>
<td>13</td>
<td>14</td>
<td>14</td>
<td>15</td>
<td>16</td>
<td>17</td>
<td>17</td>
<td>18</td>
<td>19</td>
<td>20</td>
<td>20</td>
<td>21</td>
<td>22</td>
<td>23</td>
<td>23</td>
</tr>
</tbody>
</table>

Table 2.2: Scaled quantizer values for MPEG quantization (MPEG-4)

for intra-coded blocks and $k = \text{sgn}(F'(u, v))$ for inter blocks and *quantizer_scale* is a function of QP and is listed in Table 2.2.

MPEG-4 defines a default weighting matrix $W_{\text{intra}}(u, v)$ for intra-coded blocks of the type we have been discussing so far.

$$W_{\text{intra}}(u, v) = \begin{bmatrix}
8 & 17 & 18 & 19 & 21 & 23 & 25 & 27 \\
17 & 18 & 19 & 21 & 23 & 25 & 27 & 28 \\
20 & 21 & 22 & 23 & 24 & 26 & 28 & 30 \\
21 & 22 & 23 & 24 & 26 & 28 & 30 & 32 \\
22 & 23 & 24 & 26 & 28 & 30 & 32 & 35 \\
23 & 24 & 26 & 28 & 30 & 32 & 35 & 38 \\
25 & 26 & 28 & 30 & 32 & 35 & 38 & 41 \\
27 & 28 & 30 & 32 & 35 & 38 & 41 & 45
\end{bmatrix} \quad (2.8)$$

Support for downloading custom matrices in the video sequence header is also provided.

2.3.3.1 Zig-zag Scan

For the generation of the bitstream we must rearrange the two-dimensional array of DCT coefficients into a linear sequence of individual coefficients. Panel (a) of Figure 2.6 shows the original pixels from an 8x8 block taken from the "Foreman" sequence and panel (b) shows the result of applying the 8x8 DCT to this block. These values have been rounded to the nearest integer for clarity. The result of the DCT is a set of real-valued coefficients however the fractional parts of these coefficients are not relevant if the next operation to be performed is quantization with an integer step size. The result of applying H.263 quantization (using QP=20) to these coefficients is shown in panel (c). MPEG-4 defines the scan order shown in panel (d) for scanning these coefficients which gives
Run-length coding is then performed on the AC coefficients of this sequence, the DC coefficient with value 41 is processed separately. For each non-zero coefficient, we count the length of the run of zeros that precede that coefficient and note whether this is the last non-zero coefficient in the scan. We then form a list of (run,level,last) tuples which are to be coded, e.g. for the above sequence we have:

\[(0,-13,0), (1,1,0), (0,1,0), (0,-5,0), (1,-1,0), (1,2,0), (0,2,0), (0,-1,0), (0,-1,0), (4,-1,0), (0,1,0), (1,-1,1)\]

To summarize: we have gone from a set of 64 pixel-domain coefficients which are in general distributed fairly uniformly over an 8-bit input space to (in this specific case) a DC coefficient and a list of 13 tuples whose level values are mostly ±1. This can be reconstructed at the decoder into a block which is subjectively very similar to the original block but which can been represented using a far smaller number of bits (see Section 2.3.5 on entropy coding).
2.3 Video Compression

2.3.4 Temporal Correlation

The techniques discussed so far have all worked at the level of individual frames. Indeed, the differences between MPEG and the JPEG still-image standard are minor so far as these intra-frame compression techniques are concerned. For the video case, perhaps the most important element of redundancy that can be exploited is temporal or inter-frame similarity: consecutive frames of video tend to be very similar to one another.

The MPEG video standards use block-based motion-compensated prediction (MCP) for removing temporal redundancy. Rather than coding the full texture of the image, it is divided up into blocks, each block being associated with a motion vector which specifies where in the previous frame an equal size block should be taken to use as a prediction for the block in question. This prediction block is then subtracted from the block in the current frame and the difference is coded in a manner similar to the previously described intra case. For example, if the current block is size 16x16 in a region from (17,33) to (32,48) inclusive and the motion vector for this block is (-4,-5) then we use the pixels in the region (13,28)
2.3 Video Compression

to (28,43) inclusive in the previous frame as the prediction (Figure 2.7).

All the MPEG standards divide each frame into a series of 16x16 blocks, known as macroblocks, for the purposes of motion-compensated prediction. In MPEG-4 each macroblock can have a single motion vector or, optionally, the macroblock can be split into four 8x8 blocks, each with its own motion vector. This allows for better prediction in areas where there is more than one motion present although coding four vectors instead of one requires significantly more bits and so should be used wisely.

As well as full-pel motion-compensated prediction as just described, MPEG-4 allows for fractional-pel prediction where motion vectors do not have integer values but can be specified to \( \frac{1}{2} \)-pixel (MPEG-4 Version 1) or possibly \( \frac{1}{4} \)-pixel (MPEG-4 Version 2) precision. Fractional-pixel motion-compensated prediction requires interpolation between pixel values in the frame store. For example, if the prediction for a particular pixel is to be taken from the reference frame at the point half way between two pixels \( A \) and \( B \) (half-pel prediction) then bilinear interpolation is used and the pixel in the prediction frame is computed as \((A+B+1)/2\) where the integer division operation here always rounds towards zero.

2.3.4.1 Motion Vector Prediction

At standard frame sizes, most moving objects tend to be much larger than an individual macroblock which implies that the motion vectors in adjacent macroblocks are usually correlated. To exploit this, MPEG-4 does not code motion vectors directly but rather forms differential motion vectors by subtracting a prediction vector that is calculated using one or more vectors from neighbouring blocks. To calculate the prediction vector, three candidate vectors are identified then the median value of the horizontal and vertical components of these vectors are taken to form the prediction. Which blocks are used to provide these candidate vectors depends on whether the macroblock has one vector (16x16
2.3 Video Compression

Figure 2.8: Motion vector prediction

mode) or which of the four (8x8 mode) vectors is current being considered. Figure 2.8 shows the location of the candidate blocks as 1,2,3 while X is the current block.

2.3.4.2 Motion Estimation

How the motion vectors themselves are calculated is non-normative in the MPEG standards and is not defined therein. Given that the motion vectors are transmitted in the bitstream and that the method for generating predictions is part of the standard, it does not matter how the vectors themselves are computed. The vector generation process is known as motion estimation and usually involves a search of the previous frame in the area around the block in question for the block that is the “best match” for the one in the current frame. A common
metric for this is the sum of absolute differences between the two blocks:

\[
SAD = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} |f(t_x + i, t_y + j) - f_p(t_x + m_x + i, t_y + m_y + j)|
\]  

(2.9)

where

- \(f(x, y)\) is the texture in the current frame,
- \(f_p(x, y)\) is the texture in the previous frame,
- \((t_x, t_y)\) is the location of the top-left corner of the block in the current frame,
- \((m_x, m_y)\) is the motion vector for the block, and
- \(N\) is the block size (16x16 or 8x8 in MPEG-4).

Even for small windows an exhaustive search for the best match is extremely computationally intensive. Many methods have been developed for performing non-exhaustive searches that significantly reduce the complexity of this operation. However, note that the search space is not necessarily convex and as such any global minima may not be found by these methods. Another point to consider is that motion estimation ideally finds the motion vector that results in the least number of bits once the prediction difference has been coded which is something the sum of absolute differences does not predict well in many cases. Consider the case of consecutive frames where the illumination of a static object is changing: the sum of absolute difference for the zero motion vector will be significant due to the luminance offset between frames although once the DCT of the difference is taken it would only result in a difference in the DC coefficient. The motion estimation problem is still an area of current active research [15-18] where many different techniques have been devised to deal with these issues.

2.3.4.3 Frame and Macroblock Coding Modes

All MPEG video standards classify each frame into one of three types as described below and shown in Figure 2.9.
In order to perform inter-frame prediction, it is necessary to have a previous frame to predict from therefore the first frame in a sequence can only be coded in intra mode (without prediction). A frame that is composed only of intra coded blocks is known as an I frame (I-VOP in MPEG-4). I frames are also useful where a scene change occurs and prediction from a past frame is likely to result in the difference frame requiring many more bits to code than simply using intra mode.

An MPEG video P picture (P-VOP in MPEG-4) allows for the use of temporal prediction from an earlier frame as previously described. For areas of an image where no useful prediction exists, for example at the edge of a frame where new material is coming into view from off-screen, intra coding can still be used. This mode decision can be made on a macroblock-by-macroblock basis. In order to be available for use later as a reference for prediction, each frame must be kept in a frame store. For the encoder and the decoder to produce identical predictions the reconstructed frame must be stored in the encoder frame store, not the original input frame. Differences between the contents of the encoder and decoder frame stores at a given point in time (frame store mismatch) can produce very serious degradation in decoded image quality, an issue that we
will return to in the later discussion of scalable video.

The third and final picture coding mode is a B picture (B-VOP in MPEG-4). The significance of a B picture is that it is never used in the prediction of any other frame; in contrast to I frames and P frames which can be used to predict other P frames. This means that the texture in a B picture can be quantized more heavily without requiring more bits to be spent in other frames that "depend" upon the frame. Another major difference is that prediction can be performed from one or both of two reference frames; one "past" frame as in the P frame case and a "future" frame. This selection is again done at the macroblock level; there are four macroblock modes for MPEG-4 B-VOPs, forward which uses the past reference frame and is similar to the 16x16 mode in P-VOPs, backwards which uses the "future" reference frame, bi-directional mode which averages predictions generated from both and finally "direct" mode which is similar to the bi-directional mode except that the motion vectors are computed from the corresponding vectors in the two reference frames rather than being coded directly, plus a small (± 1 maximum) difference term. Since it is clearly not possible to predict from the "future" frame without having first received it, the use of B frames require frame re-ordering. In the case of Figure 2.9, after the initial I frame is coded the fourth frame (the first P) is coded next followed by the two B frames that are between the I and the P and which use these as references for prediction. This also introduces processing delay at both the encoder and decoder which may be unacceptable for low frame rate real-time applications.

An I picture and the set of P and B pictures that follow it is known as a Group of Pictures (GOP) (Figure 2.9). Splitting a sequence into GOPs is useful for applications that do not always process a sequence in a purely linear fashion but allow for jumps to arbitrary points. Without the presence of I frames, the contents of the frame store would not be known unless the entire sequence was decoded from the beginning. With regular I frames decoding can start at the last I frame. In error-prone environments (Section 2.4.2) I frames are useful as
they “flush out” errors which may have accumulated in the frame stores.

2.3.4.4 Quantization in Inter Macroblocks

Quantization is performed in a slightly different way for macroblocks with temporal prediction than was previously described for the intra case in Section 2.3.3.

The H.263 quantization for inter blocks is modified from the intra case of equation 2.6 to provide a larger dead zone around zero:

\[ F'(u, v) = \text{sgn}(F(u, v)) \times \left( \frac{|F(u, v)| - QP/2}{2 \times QP} \right) \]  

(2.10)

Note that in this case \( F'(u, v) = 0 \) for \( F(u, v) \in (-QP, QP) \) unlike equation 2.6 which is zero for \( F(u, v) \in (-QP/2, QP/2) \). This is done to increase the number of small coefficients that are quantized to zero and thus do not need to be coded.

For MPEG quantization, MPEG-4 defines a different default weighting matrix for inter-coded blocks that is to be used instead of the intra case given in equation 2.8:

\[
W_{\text{inter}}(u, v) = \begin{bmatrix}
16 & 17 & 18 & 19 & 20 & 21 & 22 & 23 \\
17 & 18 & 19 & 20 & 21 & 22 & 23 & 24 \\
18 & 19 & 20 & 21 & 22 & 23 & 24 & 25 \\
19 & 20 & 21 & 22 & 23 & 24 & 26 & 27 \\
20 & 21 & 22 & 23 & 25 & 26 & 27 & 28 \\
21 & 22 & 23 & 24 & 26 & 27 & 28 & 30 \\
22 & 23 & 24 & 26 & 27 & 28 & 30 & 31 \\
23 & 24 & 25 & 27 & 28 & 30 & 31 & 33
\end{bmatrix}
\]  

(2.11)

Again, support for downloading a different weighting matrix is provided by the standard.

After temporal prediction the distribution of coefficients in each block is not as heavily biased towards the lower frequency components so this matrix does not attenuate high frequencies as much as the intra matrix of equation 2.8.
2.3 Video Compression

2.3.5 Entropy Coding

The (run, level, end) tuples generated by the zig-zag scan (Section 2.3.3.1) would each require 19 bits to code if we use a simple binary encoding for each value: the run length requires six bits for a maximum value of 63, the precision of the output of the DCT is 12 bits and the end flag is a single bit. The statistical distribution of these values is, however, far from uniform so we should expect to be able to do better than simple fixed-length codes. The low-pass assumption for the images we are trying to code means that long runs of zeros (implying the existence of high frequency coefficients) are rare. Similarly, studies of the distribution of AC luminance DCT coefficients, which make up the bulk of the coefficients in a coded sequence, have been shown to follow a Laplacian distribution [19] and as such large values are uncommon.

The MPEG standards define lists of variable-length codes (VLCs) to represent (run, level, end) tuples in the bitstream. VLC coding is an example of entropy coding which tries to match the number of bits required to code a symbol to the statistics of the source. Arithmetic coding, which will be discussed in Chapter 4, is another commonly-used entropy coding technique. Table 2.3 shows a small part of the list of VLCs defined in the MPEG-4 standard for inter luminance

<table>
<thead>
<tr>
<th>Run</th>
<th>Level</th>
<th>End</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>10s</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>0</td>
<td>1111s</td>
</tr>
<tr>
<td>0</td>
<td>3</td>
<td>0</td>
<td>010101s</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>...</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0111s</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>1</td>
<td>000011001s</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>001111s</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>...</td>
</tr>
</tbody>
</table>

Table 2.3: MPEG-4 VLC table for AC luminance coefficients (part)
coefficients. The symbol $s$ in the codeword is the sign of the coefficient (1 for negative, 0 for positive) so the run-level-end tuple (0,2,0) is represented by the bits 11110 and (0,-2,0) by 11111. Given statistics on the frequency of occurrence of each symbol, these codewords can be generated by applying the Huffman procedure [20].

MPEG-4 does not specify individual VLCs for all $2^{19}$ possible (run,level,end) tuples; several hundred VLCs are defined for the most common tuples and then three types of escape codes are defined to generate the bit sequences used for the remainder. The Type 1 escape code puts a regular VLC into the bitstream whose level is modified by an offset that is calculated from the run length. Type 2 escape codes work in the same way as Type 1 except in this case the offset is calculated from the level and is applied to the run length. The Type 1 and Type 2 escape codes can only be used for a fraction of the set of all possible (run,length,end) tuples. In all other cases the Type 3 escape code which codes (run,level,end) directly must be used. All escape codes start with the escape sequence 0000011 followed by a one- or two-bit VLC to indicate the escape code type followed by either a regular VLC (Type 1 and Type 2) or 19 bits (Type 3) to directly code (run,level,end).

VLC coding is also used to code many other quantities in an MPEG bitstream apart from DCT coefficients. For example, VLC tables are defined for the coding of motion vectors (Section 2.3.4.1) and coded block patterns which will be described in the next section.

2.4 Overview of MPEG-4 Syntax

Considerable portions of the MPEG standards are devoted to the exact definition of the bitstreams themselves, commonly known as the bitstream syntax. In the case of MPEG-4 we can split up a bitstream into sections that fall into one of four types (Figure 2.10):
The sequence header appears once at the start of the bitstream. This is used to specify fixed parameters that do not vary over time, such as the size in pixels of all the VOPs in the sequence, the type of quantization (H.263 or MPEG) used and any user-defined quantization matrices for the MPEG quantization.

Each VOP has its own header which contains a timestamp for this VOP, the coding type (I, P or B-VOP) and the quantizer step size to use in the first macroblock.

A header for each macroblock which will be described more fully in the next section.

The coefficients themselves, in the form of VLC-encoded (run,level,end) tuples, are then added as discussed in Section 2.3.5.

The sequence and VOP headers describe many other parameters and options that we will not discuss here. Macroblocks are processed in row order, from left to right then top to bottom.

2.4.1 The Macroblock Header

Apart from the coefficient values themselves, there are a number of other properties of a macroblock which must be coded in the bitstream:
2.4 Overview of MPEG-4 Syntax

1. The prediction mode to use in this macroblock. For P-VOPs this is either intra mode, inter mode with one motion vector or inter mode with four motion vectors (Section 2.3.4). For B-VOPs, this is one of the four modes listed in Section 2.3.4.3: forward, backward, bi-directional or direct. For I-VOPs no mode indication is required as no prediction is ever performed.

2. An indication of which of the 8x8 DCT blocks in the macroblock have any non-zero coefficients and need to be scanned. This is known as the coded block pattern (CBP). For the 4:2:0 input format there are six 8x8 blocks per macroblock; four luminance and one for each chrominance component. Using the 4:2:2 format increases this to eight with two block for each chrominance component. For intra-coded macroblocks the DC coefficient is always coded and is not included in the determination of the CBP.

3. The quantizer step size is coded differentially; the predicted value is the value in the previous macroblock or in the case of the first macroblock the quantizer step size value coded in the VOP header. MPEG-4 allows the quantizer step size to be varied by up to ±2 relative to the previous macroblock.

4. VLCs for any differential motion vectors (not required in I-VOPs or intra-coded macroblocks in P-VOPs).

In MPEG-4 the coded block pattern for the chrominance components and the macroblock mode indication are combined into a single VLC. The coded block pattern for the remaining luminance components are coded in a second VLC.

In P- and B-VOPs, the first bit of the macroblock header is an extra bit called the macroblock skipped indicator which when equal to '1' signals that the macroblock has a zero full (not necessarily zero differential) motion vector, the same quantizer step size as the previous macroblock and no non-zero DCT coefficients (i.e. the coded block pattern is all zero). When this occurs, no other
information about this macroblock needs to be transmitted. In a typical sequence many macroblocks can be “skipped” in this way and so bits are saved since such macroblocks require only a single bit to code.

2.4.2 Error Recovery

Many transport media, notably wireless networks, are liable to introduce errors into a bitstream before reaching the decoder. Many transports compute checksums that detect such errors. However, if such checks are not made then errors will normally be apparent during the VLC decoding process when an invalid bit pattern is seen or a sequence of run-level codes extends beyond the end of the block. Note that detection of an error may occur some time after the original error itself and the VLCs decoded after the original error will be incorrect. Once such an error has been detected, a synchronization point must be found so the state of the decoder is known and normal operation can be resumed.

At the start of each VOP header is a VOP start code which is a synchronization point. Additionally, if the stream has been packetized, the start of each packet contains a start code. VOP start codes are always byte aligned so if the previous frame did not end on a byte boundary stuffing bits are inserted to ensure the correct alignment. In MPEG-4, the start code is a sequence of 23 zero bits followed by a '1'. Care must be taken that this bit sequence does not occur anywhere else in the bitstream. Where it is possible that this might happen, the syntax uses a marker bit with value '1' to break up any sequence of zeros that could be too long. If the start code bit pattern occurs anywhere else other than at the location of a real start code is known as start code emulation and should be considered as a flaw in either the standard or the encoder that generated the bitstream.

It is normally the case that all data between where an error was detected and the following synchronization point must be discarded. In the case of isolated bit errors or short error bursts almost all of this data is uncorrupted but can-
2.5 Overall Structure

not be used. MPEG-4 supports an optional feature called Reversible VLC (RVLC) which allows this data to be processed backwards from the synchronization point after the error has been detected [21]. This improved error recovery behaviour comes at the cost of longer VLCs than using the regular tables.

Another use for VOP header start codes is to provide random access to a bitstream: a decoder can ignore the contents of each frame and scan the bitstream for start codes, read some or all of the information in the VOP header and recommence full decoding when the desired location has been reached.

Note that on many packet networks the presence of bit errors is detected by the transport layer and the packet will be discarded. In this case techniques like RVLC are not useful. Methods do exist, however, to attempt to conceal the loss of video packets [22, 23].

2.5 Overall Structure

The previous sections have provided all the tools we need to describe the fundamental operation of a single-layer MPEG-4 encoder and decoder. Figure 2.11 shows the high-level data flow in a single-layer encoder and Figure 2.12 shows the structure of a decoder to match.

To briefly summarize the encoder: for B-VOPs and P-VOP macroblocks that are not coded in intra mode, motion vectors and a corresponding prediction macroblock are generated, this prediction is then subtracted from the input signal. The DCT is then applied to this residue signal, quantized and if the block is intra coded any intra coefficient prediction is subtracted before the bitstream syntax generation stage. To form the reconstructed frame that goes into the frame store (for use as reference for prediction in later frames), the quantized signal (without any intra coefficient prediction being applied) is unquantized, the inverse DCT is applied and added to the original prediction.

The operation of the decoder (Figure 2.12) is essentially identical to the last
2.5 Overall Structure

Figure 2.11: MPEG-4 single-layer encoder

Figure 2.12: MPEG-4 single-layer decoder
part of the encoder loop: once the bitstream is decoded it is unquantized, converted back to the spatial domain and added to any prediction that is required for the current macroblock mode.

2.6 Rate Control

If we keep the quantizer step size at a constant value, it should be apparent that the compression ratio that we achieve is certainly not constant over the ensemble of all possible frames. A frame that is a single uniform colour will only be composed of DC coefficients (which after DC coefficient prediction will almost all be zero) whereas one that is some “noise” signal will contain many coefficients and require significantly more bits. Many transport and storage media are designed to work at constant bit rates so it is necessary to manipulate the quantization of sequences that are to be used with such media. This is a process known as rate control. Note that rate control is non-normative to video coding standards: so long as the syntax supports signalling quantizer changes to the decoder all the work of the rate control algorithm is confined to the encoder.

Using a constant quantizer step size can also be unsatisfactory due to the fact that the resulting video does not appear to be of uniform quality to human observers. We have already discussed blocking artifacts that appear in flat areas of individual frames: these areas often appear to be significantly worse than other areas quantized to the same degree. Similarly, quantization artifacts are often masked in regions that are rapidly changing over time, whereas static areas must be quantized less heavily to avoid leaving obvious imperfections. Perceptual factors affecting rate control will be discussed more fully in Chapter 5.

Rate control algorithms can be categorized into two main types: constant bit rate (CBR), which as described earlier, is for transport over channels with hard
bandwidth constraints, for example videoconferencing over leased telephone lines. Variable bit rate (VBR) rate control, on the other hand, allows the bit rate to fluctuate in the short term but attempts to meet an average bit rate target over a long period of time while maintaining approximately constant quality. Rate control for DVD Video is an example of the VBR case [24]. One point should be made here about CBR rate control: it is usually not possible to achieve the bit target exactly for every frame. However, since the decoder is assumed to be receiving input at a constant rate it is necessary for this data to be placed into a buffer which acts to smooth out any fluctuations in the actual number of bits required to code a picture. Since decoder memory is expensive to provide in some applications and frames that fill the buffer create end-to-end delay when transmitted by a constant rate channel, it is essential that CBR rate control algorithms keep the required buffer space to a minimum and ensure that the buffer never overflows or underflows.

Achieving good performance in a rate control algorithm often involves a trade-off against higher computation and latency in the encoder. To achieve useful estimates of how many bits will be generated by a frame of video given a specific set of macroblock QP settings, the algorithm needs more information than is readily available at the time we want to make choices for values of QP. The frame to be coded tells us very little about how many bits will be required to compress it, since we do not know what effect prediction and the DCT will have. For low rate applications, motion vectors make up a large proportion of the bitstream and the total number of bits required for this simply cannot be estimated without going through the motion estimation process. As will be seen in Chapter 5, reasonable estimates can be made if we know how many bits the motion vectors will require and we can make measurements of the unquantized DCT coefficients. However, this requires that almost all the computation for the frame be completed before any quantizer step size choices can be made and bits can be output and so introduces a single frame delay. For real-time applications,
such as videoconferencing, this is unacceptable. For other applications such as DVD authoring which are done off-line, latency is not an issue and some extra computation is tolerable in which case a two-pass approach can be made: the sequence is encoded once and various statistics are gathered and then used to refine the rate control decisions in a second pass through the sequence.

2.7 Conclusion

This chapter has introduced the basic concepts and techniques used in single-layer MPEG-4 video compression. The next chapter will survey existing techniques for multi-layer scalable video compression, many of which are also supported in the MPEG-4 standard. The later chapters on stream morphing also rely heavily on the details of single-layer MPEG-4 and will refer back to details that has been covered here.
Chapter 3

Scalable Video Techniques

3.1 Introduction

Section 2.2 introduced the concept of scalable video: that a single high-quality video service can be created and partitioned such that if we receive only one section of the data we can still provide a low-quality service and that the quality of the service scales with the receipt of additional data.

Scalable coding is useful in a number of contexts. The first is service interworking where the video is to be received by many different users, only some of whom have the resources (bandwidth and/or computational power) to decode the higher quality service. Scalable coding allows all classes of users to be served without the need for bandwidth-intensive simulcasting, where a number of independent services with differing qualities are transmitted in full over separate channels. In the simulcasting case the receiver decodes the service that best satisfies its constraints and ignores all the others. Scalable coding provides these simultaneous services while making use of all available bandwidth. Provision of standard and high definition digital television is a good example of a potential application for service interworking. Another application of scalable coding is in error-prone environments, including wireless or packet networks such as the Internet. In this case, a low-quality service can be augmented by extra data under favourable conditions of low packet loss or bit error rate.
The “quality” of a video sequence is a combination of three factors:

- The spatial resolution of each frame in the sequence.
- The frame rate.
- The amount of distortion present in the frames themselves.

As such, scalable video comes in three main forms:

**Spatial Scalability** The base layer service alone has low spatial resolution which is improved if the enhancement layer is received.

**Temporal Scalability** The frame size is constant however the frame rate is higher when the enhancement layer is decoded.

**SNR Scalability** The frame size and frame rate are the same for both layers but the quality appears better when multiple layers are received because each frame has more detail, e.g. because more DCT coefficients have been received and/or they are coded more precisely.

Hybrid systems that employ two or more of these techniques in a single enhancement layer are possible. There is also no theoretical limit to the number of layers that may be present in a given system, where different techniques or combinations of techniques may be used in different layers.

The application of scalable video in a particular scenario is very dependent on the properties of the transport medium being used. Most schemes create layers that can only be used in a particular order. This requires that either the transport medium has some in-built mechanism for prioritization of different data streams or extra protection from errors needs to be provided to specific parts of the data. The current Internet is an example of a medium where prioritized scalable video cannot be effectively used since all packets are treated the same and error protection can only take the form of packet re-transmission which is very bandwidth intensive (and often counter-productive in environments where significant congestion already exists). In this case a symmetric
scheme, in which the layers can be treated equally and received in any order without the need for prioritization, is more appropriate. Multiple description coding, to be discussed in Section 3.8.3 is an example of such a technique. For wireless networks different amounts of forward error correction can be applied, which give the effect of different priority levels being available.

This chapter describes some existing approaches to the problem of how to partition a video sequence into a scalable form and is structured as follows: Section 3.2 covers SNR scalability in MPEG-2 which is the starting point for the discussion of the new scalable video scheme that will be introduced in subsequent chapters. Section 3.3 discusses temporal and spatial scalability in both the MPEG-2 and MPEG-4 standards. Section 3.4 covers MPEG-4's Fine Granularity Scalability scheme which has been recently added to the standard for use with streaming network video. Sections 3.5 and 3.6 show some experimental results comparing the behaviour of the previously described SNR scalability schemes. Section 3.7 discusses these results. Finally in Section 3.8 some other approaches to scalable coding that are not in the MPEG standards are outlined. Conclusions are drawn in Section 3.9.

3.2 MPEG-2 SNR Scalability

Figure 3.1 shows a block diagram of the SNR scalable decoder that is defined in the MPEG-2 [2] standard. There is a single prediction loop and frame store as in the single-layer case, but the residue that is added to the prediction is refined by adding the reconstructed DCT coefficients from the enhancement layer bitstream to the coefficients from the base layer. MPEG-2 allows for only a single enhancement layer although it should be clear that this process can be extended to allow for the addition of unquantized DCT coefficients from two or more enhancement layers to form a system with three or more layers.

MPEG-2 does not specify the structure of the encoder corresponding to the
3.2 MPEG-2 SNR Scalability

decoder shown in Figure 3.1. There are three main encoder configurations that are compatible with this decoder:

1. A simple approach to encoding the enhancement layer is to calculate the error introduced by the base layer quantization and to quantize this signal again with a smaller quantizer step size then send this as the enhancement layer. Figure 3.2 shows an encoder of this form with a single enhancement layer. This can be extended to two or more enhancement layers by measuring the difference across the enhancement layer quantizer/inverse quantizer pair and passing this through a yet finer quantizer and so on. This approach is known as single-loop because it has only the one prediction loop (in the base layer).

2. Figure 3.3 shows another encoder that is compatible with the decoder of Figure 3.1. The enhancement layer now has its own prediction loop, this adds complexity to the encoder, but this will be shown to be worthwhile as there is normally useful prediction to be gained from the residue in previous frames, especially when the quality of the base layer is low. If the
Figure 3.2: Single-loop MPEG-2 SNR Scalable encoder (with drift)
3.2 MPEG-2 SNR Scalability

quality of the base layer is very high the residual signal and the prediction is uncorrelated "noise", which when subtracted may result in a signal with greater energy that the original residue. In this case, a single-loop scheme is preferable. This scheme is known as a two-loop (or multi-loop) encoder, although again this can be extended to two or more enhancement layers by subtracting the sum of the reconstruction signals in all lower layers rather than just the reconstruction from the base layer.

3. Figure 3.4 shows a two-loop pyramid encoder and is a variation on the encoder of Figure 3.3. The major difference between these two structures is that while the enhancement layer frame store of Figure 3.3 stores a differential signal (the quantized difference between the original signal and the base layer reconstruction) the same frame store in Figure 3.4 stores a signal that is identical to the frame store in the single-loop decoder (for Figure 3.3 the decoder frame store is the sum of all the encoder frame stores). From an implementation point of view this is preferable since the same frame store and prediction logic can be used in all layers rather than having to design a separate pair of these to deal with 9-bit (signed) differential signals in the enhancement layers. The pyramid encoder can be extended to two or more enhancement layers by using the sum of the unquantized DCT coefficients from all lower layers to predict the DCT residue.

Since there is only a single motion-compensation loop in the MPEG-2 SNR scalable decoder (Figure 3.1), it is important to note for the multi-loop encoders that all layers must use the same set of motion vectors when calculating their respective motion-compensated predictions. This also means that motion estimation need only be performed in one layer (the figures show this being done in the base layer but this can be done in any layer) and no motion vectors need to be sent in the enhancement layer bitstream(s).
Figure 3.3: Two-loop MPEG-2 SNR scalable encoder
3.2 MPEG-2 SNR Scalability

Figure 3.4: Two-loop MPEG-2 SNR scalable encoder (pyramid)
3.2.1 Drift

As well as being less efficient than the multi-loop schemes when the base layer quality is low, the single-loop encoder of Figure 3.2 has another problem. If we decode only the base layer because the enhancement layer is unavailable, the contents of the encoder and decoder frame stores will tend to drift apart since the enhancement layer reconstructed texture is still being added into the encoder frame store whereas this is not occurring at the decoder. This will cause the quality of the decoded pictures to degrade significantly in a short period of time [25]. The effects of drift are removed by the use of intra-coded macroblocks or entire frames however this decreases coding efficiency substantially.

To avoid drift it is essential that the contents of any frame store be computed from the current layer and any lower layers but should not be dependent on data from higher layers that might not be received.

The use of the multi-loop encoders shown in Figure 3.3 and 3.4, in conjunction with the same MPEG-2 decoder (Figure 3.1) does not suffer from significant drift when some of the layers are not received at by decoder. To see why this is the case, we begin by considering the contents of the decoder frame store $F_{n,l}$ in frame $n$ of an $l$ layer system where all $l$ layers have been received since the last $I$ frame:

$$F_{n,l} = P(F_{n-l,1}, V_n) + \sum_{i=1}^{l} T_{n,i}$$

where $P(\ldots)$ is the motion compensated prediction operation, $V_n$ is the set of motion vectors in frame $n$ and $T_{n,i}$ is the received texture in layer $i$ of frame $n$. Consider that at frame $n$ only $l' < l$ layers are received, so we have

$$F_{n,l'}^* = P(F_{n-l',1}, V_n) + \sum_{i=1}^{l'} T_{n,i}$$

We can express $F_{n-1,l}$ as the sum of $F_{n-1,l'}^*$ which is the frame store we would have obtained if we had been decoding only $l'$ layers all along (which corre-
3.2 MPEG-2 SNR Scalability

spends to the frame store in layer \( l' \) in the encoder) plus a mismatch term \( C \):

\[
F_{n,l'}^* = P(F_{n-1,l'} + C, V_n) + \sum_{i=1}^{l'} T_{n,i} \tag{3.3}
\]

Full-pel motion compensated prediction is a linear operation so

\[
P(F_{n-1,l'} + C, V_n) = P(F_{n-1,l'}, V_n) + P(C, V_n) \tag{3.4}
\]

which means that

\[
F_{n,l'}^* = P(F_{n-1,l'}, V_n) + P(C, V_n) + \sum_{i=1}^{l'} T_{n,i} \tag{3.5}
\]

\[
= F_{n,l'} + P(C, V_n) \tag{3.6}
\]

i.e. decoder frame store mismatch is shifted around by prediction but does not increase in energy over time like the single-loop case. In many cases, the \( P(C, V_n) \) term will be small and spread over the whole frame and as such may not be noticeable to the viewer. Indeed at very high quality levels this term may be effectively uncorrelated noise.

The important point here is that \( F_{n,l'} \) has a corresponding loop at the encoder which controls the difference between this signal and the input. There is no counterpart to \( F_{n,l'} \) in the single-loop case since there is only one feedback loop which is predicated on all layers being received. Having \( l \) encoder loops controls drift in all layers. The linearity of prediction keeps any mismatch effectively separate.

Fractional-pel motion-compensated prediction is not a linear operation like the full pel case. For example, the bilinear interpolation (Section 2.3.4) between two adjacent pixels \( A \) and \( B \) is \((A + B + 1)/2\) but if the sum of \( A \) and \( B \) is odd (and therefore \( A + B + 1 \) is even) then adding an extra 1 to either \( A \) or \( B \) will not have any effect on the result due to truncation.

We predict that the effect of this nonlinearity will be very small since it affects only macroblocks with fractional motion vectors and even when it does occur it only results in a difference of \( \pm 1 \) per pixel per frame. Over time, the use of
intra-coded macroblocks (especially in high-motion sequences) will reduce the effect of the mismatch term. Experimental results will be given on the behaviour of multi-loop SNR scalable encoders with frame store mismatch in Section 3.6.

Finally, we note that the single-loop encoder shown in Figure 3.5 can be considered to be drift-free since it does not feed the enhancement layer texture into the base layer frame store. It is, however, not compatible with the MPEG-2 SNR scalable decoder, which would need to be modified to prevent the enhancement layer texture from entering the single decoder frame store (Figure 3.6).
3.3 Temporal and Spatial Scalability

Both the MPEG-2 and MPEG-4 standards support temporal and spatial scalability and do so in largely the same way. We will describe the MPEG-4 case first and then mention differences that exist between MPEG-4 and MPEG-2.

3.3.1 MPEG-4 Generalized Scalability

The MPEG-4 standard defines the single architecture shown in Figure 3.7 for implementing all types of scalable decoding. The base layer decoder here is an unmodified single-layer MPEG-4 decoder as discussed in Chapter 2. The enhancement layer decoder is modified such that it can select from a number of possible reference frames (in both the current layer and any lower layer) for use with P-VOP and B-VOP prediction. In MPEG-4 these selections are signalled in the VOP header and cannot be changed during the frame. Table 3.1 lists the four possible references that can be used for P-VOP prediction in an enhancement layer and Tables 3.2 lists the four possible combinations of forward and backward references (Section 2.3.4.3) that can be used with enhancement layer
3.3 Temporal and Spatial Scalability

Figure 3.7: MPEG-4 decoder for generalized scalability

<table>
<thead>
<tr>
<th></th>
<th>References for prediction in MPEG-4 enhancement layer P-VOPs</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The most recent decoded VOP in the enhancement layer</td>
</tr>
<tr>
<td>2</td>
<td>The most recent VOP in display order in the reference layer</td>
</tr>
<tr>
<td>3</td>
<td>The next VOP in display order in the reference layer</td>
</tr>
<tr>
<td>4</td>
<td>The temporally-coincident VOP in the reference layer (motion vectors identical to those in reference layer)</td>
</tr>
</tbody>
</table>

B-VOPs. Figure 3.8 graphically depicts these prediction modes for the P-VOP case and Figure 3.9 shows the B-VOP case. Some of the reference frames can be used in only one configuration or the other, for example there is often no temporally-coincident VOP in temporal scalability, also the next VOP in the reference layer in spatial or SNR scalability may not have been decoded yet and cannot be used. Figure 3.9 shows the forward B-VOP prediction using solid lines and the backward prediction as dashed lines.

The roles of the “mid processor” and “post processor” blocks in Figure 3.7 depends on the type of scalability being used and will be outlined in the following two sections that describe how spatial and temporal scalability are implemented in MPEG-4.
3.3 Temporal and Spatial Scalability

(a) SNR or Spatial Scalability

(b) Temporal Scalability

Figure 3.8: References for prediction in MPEG-4 enhancement layer P-VOPs (corresponds to Table 3.1)

<table>
<thead>
<tr>
<th>Forward Reference</th>
<th>Backward Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 The most recent decoded VOP in the enhancement layer</td>
<td>The temporally-coincident VOP in the reference layer (motion vectors identical to those in reference layer)</td>
</tr>
<tr>
<td>2 The most recent decoded VOP in the enhancement layer</td>
<td>The most recent VOP in display order in the reference layer</td>
</tr>
<tr>
<td>3 The most recent decoded VOP in the enhancement layer</td>
<td>The next VOP in display order in the reference layer</td>
</tr>
<tr>
<td>4 The most recent VOP in display order in the reference layer</td>
<td>The next VOP in display order in the reference layer</td>
</tr>
</tbody>
</table>

Table 3.2: References for prediction in MPEG-4 enhancement layer B-VOPs
Figure 3.9: References for prediction in MPEG-4 enhancement layer B-VOPs (corresponds to Table 3.2)
3.3 Temporal and Spatial Scalability

3.3.1.1 Spatial Scalability

For spatial scalability, the base and enhancement layer decoders output frames at the same rate, but the base layer reconstruction is the wrong size to be used directly for prediction in the enhancement layer. The mid processor in Figure 3.7 produces a spatially-upsampling version of the base layer reconstruction so that it can be used for this purpose. The post processor optionally upsamples the base layer reconstruction for systems that require the frame size be constant even if the enhancement layer is not available.

Figure 3.10 shows one possible arrangement for the prediction references for spatial scalability. The first enhancement layer VOP is a P-VOP which predicts from the temporally-coincident I-VOP which is the fourth mode in Table 3.1 or Figure 3.8(a). The remaining enhancement layer B-VOPs use forward prediction from previous VOP in the same layer and backwards prediction from the temporally-coincident VOP in the base layer (first mode in Table 3.2 or Figure 3.9(a)).
3.3 Temporal and Spatial Scalability

3.3.1.2 Temporal Scalability

In the temporal scalable case only a subset of VOPs in the full sequence are coded in the base layer. Since the base layer VOPs are of the same size as the enhancement layer VOPs, the mid processor does not need to perform any scaling. There are a number of possible schemes for placement of enhancement layer VOPs. The simplest of these would be to code VOPs only in the "gaps" in the base layer. In this case we need the post-processor to interleave the VOPs from the base and enhancement layers for display. Alternatively, Figure 3.11 shows every VOP being coded in the enhancement layer: the existing base layer VOPs are refined as well as the "gaps" being filled. The first enhancement layer VOP (a P-VOP) in Figure 3.11 is coded with respect to the first base layer I-VOP and corresponds to the first prediction mode listed in Table 3.1 and shown in Figure 3.8(b). The "gap" B-VOPs (the second and fourth enhancement VOPs in the figure) use bi-directional prediction with the last frame in the enhancement layer and the next VOP in display order in the base layer being used as
3.3 Temporal and Spatial Scalability

The enhancement layer prediction in MPEG-2 spatial scalability is formed from two components as shown in Figure 3.12: a spatial prediction that is generated by upsampling a recently-received frame from the lower layer, and a temporal prediction that is computed in the same way as in the single-layer case using a reference picture (or pictures in the case of bi-directional prediction) from the same (upper) layer. The macroblock header then codes whether the temporal prediction (weights $A = 1 B = 0$), the spatial prediction ($A = 0 B = 1$) or an interpolated combination of the two ($A = B = \frac{1}{2}$) is to be used for the macroblock.
3.3 Temporal and Spatial Scalability

in question. This last interpolated mode has no similar counterpart in MPEG-4, since the prediction there can be a combination of three reference frames in the case of bi-directional temporal prediction in the enhancement layer. For interlaced coding the weights $A$ and $B$ can be selected independently for each field, $A_T$ for the top field and $A_B$ for the bottom field and similarly for the weight $B$. MPEG-2 explicitly allows for the base layer for spatial scalability to be coded using the earlier ITU H.261 standard or MPEG-1 as well as MPEG-2 itself.

Temporal prediction in MPEG-2 splits an input video sequence into two separate sequences which are then coded by two notionally separate encoders, the even numbered frames become the base layer and the odd numbered frames are coded by an enhancement layer encoder. Note that since the same number of frames are being coded as in the single-layer case, the computational complexity is identical. The only difference is that there is a wider choice of frames for use in prediction. The first three P-VOP prediction modes listed in Table 3.1 are supported in MPEG-2 as are the last three B-VOP prediction modes of Table 3.2. Unlike MPEG-4, temporally-coincident frames are not supported in MPEG-2 temporal scalability.

### 3.3.3 SNR Scalability in MPEG-4

It is worth noting that the MPEG-2 SNR scalable decoder of Figure 3.1 cannot be emulated using the generalized scalability in the MPEG-4 standard. Single-loop SNR scalability can be implemented using P-VOP coding in the enhancement layer predicting from the current base layer VOP (fourth mode in Table 3.1). MPEG-4 spatial scalability (Section 3.3.1.1) can be performed with an upsampling ratio of 1:1 which is then effectively SNR scalability, however the decoder complexity associated with such an approach is very high, requiring one motion-compensated prediction loop and IDCT per layer.
The second amendment to the MPEG-4 standard [26] defines a new type of scalable coding known as Fine Granularity Scalability (FGS) [27], designed for use in broadcasting and packet network environments where bandwidth adaptability is required.

Figure 3.13 shows a schematic of the FGS encoder. FGS is a single-loop scheme: motion-compensated prediction is done only in the base layer. This affects its performance relative to multi-loop schemes, especially at low base-layer bit rates. One advantage of the single-loop schemes is that as long as the base layer is received intact, any amount of the enhancement layer data can be lost without introducing drift.
After the base layer reconstruction is formed, the FGS encoder subtracts this from the original picture to obtain a difference picture. This is passed through a second DCT and the result of this is split into 8x8 blocks and partitioned into bit-planes, e.g. (from [27]) the sequence of residue DCT coefficients

\[(10, 0, 6, 0, 0, 3, 0, 2, 2, 0, 0, 2, 0, 0, 1, 0, \ldots, 0, 0)\]

can be expressed as a sum of powers of two

\[
\begin{align*}
\text{MSB} & : 2^3 \times (1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, \ldots, 0, 0) + \\
\text{MSB} - 1 & : 2^2 \times (0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, \ldots, 0, 0) + \\
\text{MSB} - 2 & : 2^1 \times (1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, \ldots, 0, 0) + \\
\text{MSB} - 3 & : 2^0 \times (0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, \ldots, 0, 0) \\
\end{align*}
\]

which are then each converted to (run,end) symbols in the same manner as described in Section 2.3.3.1 although the level has an implicit value of 1 in this case:

\[
\begin{align*}
\text{MSB} & : (0, 1) \\
\text{MSB} - 1 & : (2, 1) \\
\text{MSB} - 2 & : (0, 0), (1, 0), (2, 0), (1, 0), (0, 0), (2, 1) \\
\text{MSB} - 3 & : (5, 0), (8, 1) \\
\end{align*}
\]

where MSB is the most significant bit-plane.

DCT coefficients can be negative (unlike those chosen for the example) so a sign bit is coded after the most significant bit for each coefficient.

For each level there are three bit-planes: one for luminance and two for the chrominance components. Since the chrominance components normally have a smaller range of values, one or more of their most significant bit-planes may be all zero which is signalled by a special symbol. A standard FGS bitstream is then generated by performing a zig-zag scan on each 8x8 block in each plane,
3.4 MPEG-4 Fine Granularity Scalability

Figure 3.14: MPEG-4 Fine Granularity Scalability decoder

starting with the most significant bit-plane then working down to the least significant bit-plane. Since the bit-plane-coded enhancement texture never enters the decoder frame store, drift can never occur so long as the base layer is always received intact. Indeed, the enhancement layer bitstream can be truncated at any point without adversely affecting the operation of the decoder. This truncation is done by the encoder and/or by sending the FGS enhancement data over a prioritized communication channel. Some FGS application scenarios will be described in the next section.

Figure 3.14 shows the decoder for FGS which is identical to the non-compliant decoder required for the single-loop SNR scalable encoder of Figure 3.5 apart from the bit-plane coding and different enhancement layer syntax.

FGS defines two mechanisms for dealing with perceptual coding as discussed in Section 2.6. *Frequency weighting* allows for various frequency components from all blocks to be moved into higher bit-planes for transmission; this is done to allow for the subjectively more important lower frequency coefficients to appear earlier in the bitstream so that they are more likely to appear at the decoder before the bitstream is truncated. The decoder shifts these coefficients
3.4 MPEG-4 Fine Granularity Scalability

Figure 3.15: Frequency weighting for MPEG-4 FGS

Figure 3.16: Selective enhancement for MPEG-4 FGS
3.4 MPEG-4 Fine Granularity Scalability

back to their original position before applying the IDCT. This is illustrated in Figure 3.15: each position on the zig-zag scan has associated with it a shift, the set of 64 shift values used for each 8x8 block is known as the frequency weighting matrix. The VOP header signals whether this matrix is one defined in the MPEG-4 standard, or a custom matrix that is coded in the bitstream similar to the downloadable matrix support for MPEG quantization (Section 2.3.3). FGS selective enhancement allows for similar shifting to occur for entire macroblocks that are deemed to be subjectively more important, as depicted in Figure 3.16. Note that each macroblock must contain extra syntactic elements for signalling how much shift is needed if selective enhancement is enabled. This will cause an increase in the number of bits generated.

FGS in the MPEG-4 standard also supports temporal scalability: bi-directionally predicted VOPs are inserted between base layer VOPs and then the residue after this prediction is coded using the FGS bit-plane method. These VOPs can be interleaved with the FGS enhancement of the existing base VOPs or coded in a separate bitstream.

A subsequent proposal, Progressive Fine Granularity Scalability (PFGS) [28], is designed to increase the efficiency of FGS by using an extra enhancement layer frame store for prediction which contains some of the bit-plane texture that is used only for intra coding of residuals in standard FGS. This new frame store introduces the possibility for drift if some or all of the enhancement layer data is not received by the decoder. This requires that the contents of the new frame store be refreshed frequently. The decoder for PFGS has even higher complexity than regular FGS, requiring a third IDCT and a second frame store and MCP unit.

In addition to SNR and temporal scalability support in MPEG-4, spatial scalability has also been proposed [29] for possible future inclusion in the standard.
Figure 3.17: Progressive Fine Granularity Scalability encoder
3.4.1 MPEG-4 FGS Applications

Arbitrary bitstream truncation in MPEG-4 FGS is most useful in cases where the encoder or transmitter/server is aware of the amount of bandwidth available and this is stable over the short term. A number of such applications have been identified [30]:

- Bit rate reduction in broadcasting systems that currently require transcoding. For example, many cable television providers receive their content from satellite feeds that are operating at a higher bit rate than the service they are to provide on the cable. The bit rate of the service therefore needs to be reduced, a process which normally requires the video be decoded and re-encoded at the lower bit rate which is a computationally-intensive task. With FGS the bit rate can be reduced by simple truncation.

- Provision of services to many clients over a shared network connection with fixed bandwidth is another application that requires bit-rate adaptability. Consider a content provider that is providing services to \( n \) clients over a fixed bandwidth link. To maximize the quality to those clients the link should be used to 100% capacity however if another service is to be provided to a new client \( n + 1 \) then the rates for the original \( n \) clients must be reduced to “make room” for the additional service.

- Multicasting to clients with different amounts of available bandwidth is another bit-rate reduction application similar to the cable television example above. As the video passes through each node in the multicast tree its rate can be successively reduced by bitstream truncation to the maximum rate required for all clients on each particular branch.

For environments such as the Internet that suffer from significant packet loss, the ability to handle bitstream truncation is not immediately useful given that packets are likely to be lost anywhere within a stream, not just at the end of the stream. Networks that provide packet prioritization, however, allow for
control of where packet loss occurs. For example, if packets belonging to the base layer of an FGS system are tagged as being high priority, the first FGS bit-plane at a lower priority, the second plane at a lower priority still and so on then packet loss will be concentrated in the higher bit-planes, any incomplete plane that is received can be discarded but the lower planes are received intact. This is equivalent to bitstream truncation at the bit-plane level and is useful for applications such as real-time streaming video where feedback to the encoder about network conditions is not possible.

3.5 Experimental Procedure

Experiments were conducted to compare the various SNR scalability schemes that have been discussed. We discuss temporal and spatial scalability again in Chapter 6.

Video coding experiments come in two main types: those which measure objective criteria, often the rate-distortion profile of a particular algorithm when coding a given sequence with a particular set of coding parameters, and those tests which use human observers to make subjective judgements about the quality of sequences displayed before them. The most commonly used objective measure of video quality is the Peak Signal-to-Noise Ratio (PSNR) which is defined as:

\[
\text{PSNR}(\text{dB}) = 10 \times \log_{10} \left( \frac{255^2}{\text{MSE}} \right)
\]  

(3.7)

where MSE is the mean square error calculated between the coded sequence and the original input sequence. Note that while it is common to compute the PSNR on a per-frame basis, the correct method for computing the PSNR over a multi-frame sequence is to calculate the mean square error over all frames and apply the above formula once. This is the method by which sequence PSNR values are calculated throughout this thesis. This is done in preference to computing the mean of the PSNR values for each frame which tends to underestimate the
impact of poor quality frames on the overall sequence\(^1\).

While such objective tests are easy to conduct, they often give misleading results when comparing different coding techniques because different types of artifacts introduced by those techniques may have quite different effects on a human observer while having similar values for the PSNR. Perception of distortion also depends on the content of each frame and how fast it is moving, neither of which is taken into account by the PSNR measurement. These effects are discussed in detail in Chapter 5 in the context of rate control algorithms that seeks to exploit the human visual system's insensitivity to distortion under certain circumstances. In order to "calibrate" the objective measurements, it is therefore necessary to show coded sequences to human observers and ask them to make judgements about the quality of those sequences. This thesis therefore supplies results from both types of test, some initial subjective tests comparing MPEG-2-compliant SNR scalability and MPEG-4 FGS are shown in this chapter while Chapter 5 shows a larger set of results comparing these two techniques to the new stream morphing algorithm. Before we describe the subjective tests the rate-distortion and other objective tests will be discussed.

### 3.5.1 Objective Tests

Three types of scalable systems were implemented: the single-loop (drift free, non-MPEG-2-compliant) system of Figure 3.5, the multi-loop pyramid encoder of Figure 3.4 which is compatible with the MPEG-2 decoder of Figure 3.1 and FGS from MPEG-4 (encoder shown in Figure 3.13 and decoder of Figure 3.14). Each encoder is based on the software which corresponds to MPEG-4 Version 1 with appropriate custom-written decoders except for the case of FGS where the decoder is the MoMuSys software used in the standardization process (the revision of 31/12/2000) [31]. Apart from FGS, we will not consider other forms of SNR scalability that can be implemented in MPEG-4 due to their high decoder

\(^1\)This approach has recently been used by MPEG.
complexity. Each of the schemes tested here has a single motion-compensated prediction loop in the decoder, regardless of the number of layers present and either one or two (in the case of FGS) IDCTs. Note that for the tests involving MPEG-2 SNR scalability the codecs that were implemented use the MPEG-4 bitstream syntax and thus the results can be directly compared with the other systems.

Appendix B describes the test sequences used in this thesis. For this chapter the low-motion “Akiyo” and “Mother & Daughter” sequences along with the high-motion “Foreman” and “Carphone” sequences are used. The common conditions for each test are: input sequences are 100 frames long, CIF size (352x288 pixels), 4:2:0 format at 10 frames/sec. There is a single I-VOP at the start of each sequence followed by 99 P-VOPs. Full search motion estimation is done with a search range of 32 pixels. Half-pel estimation and prediction are enabled along with 8x8 motion vectors however OBMC\(^2\) is not used. The full-pel search is done in all cases using original input frames (i.e. no distortion due to compression) whereas the half-pel estimation is done using the reconstruction. This is in line with the behaviour of the MPEG-4 Verification Model (VM) software. As applications generally seek to optimize the quality at the top layer, the reconstruction used for the half-pel estimation is taken from the frame store in the topmost layer, except in the case of FGS where there is only one frame store (in the base layer).

For the objective tests it will suffice to use only one of the low- and high-motion sequences, “Mother & Daughter” and “Foreman” respectively, to demonstrate the behaviour of these systems. The first set of tests measures the rate-distortion behaviour of the scalable systems and compare these with single-layer coding. The tests are conducted with constant quantizer step size, each with QP=31 in the base layer. For the pyramid and single-layer tests, the enhancement layer quantizers were adjusted to produce four enhancement layers

\(^2\)Overlapped Block Motion-Compensation (OBMC) is an optional feature of MPEG-4 temporal prediction that blends texture across block boundaries to prevent blocking artifacts.
that were approximately evenly spread over twice the base layer bit rate. For FGS the full set of bit-planes was generated and then truncated for testing at various bit rates in this range then decoded. The PSNR of the resulting sequence was measured.

The second set of tests measure bitstream statistics from three- and five-layer pyramid coders and a three-layer single-loop coder operating between the same pair of constant quantizer step size values at both the bottom and top layers. It is important to note that these are not the same conditions as used in the first set of tests. This is designed to show the difference between systems that one might expect to operate in a similar way given the top layers all have the same quantizer setting. A number of different parameters were recorded: the quality (PSNR), bit rate, the number of coefficients coded, blocks coded and macroblocks coded in each layer along with the total scan length for each layer i.e. the sum of the length of the coefficient scan for each block coded from the first coefficient in the block until the last along the zig-zag scan defined in the MPEG-4 standard.

The final set of tests measured the performance of the pyramid encoder and MPEG-2 decoder when frame store mismatch occurs. Five layer sequences were coded and the decoder allowed to run for the first 20 frames and then the top two enhancement layers are discarded for the remainder of the sequence. The PSNRs for each frame were measured at the decoder and compared to those in the original layers at the encoder.

3.5.2 Subjective Tests

In this section a brief overview of the subjective testing process is given. Full details are provided in Appendix C.

The subjective test methodology used in this thesis is a slightly modified version of the Double-Stimulus Continuous Quality-Scale (DSCQS) method as described in ITU-R Recommendation 500 [32]. Observers are shown a number
of test presentations, each presentation is composed of two sequences $A$ and $B$ which are each of 10 seconds duration that are shown twice ($A, B$ followed by $A, B$ again) and then are asked to give a grading to each sequence on a scoresheet similar to that shown in Figure 3.18. Each scale is continuous so a mark can be made anywhere upon it, the tick marks and grading scale is only present as a guide. No explicit statements or demonstrations are given to the observers as to what constitutes “Excellent”, “Good” etc video quality. It is thought that attempting to “calibrate” each viewer's quality rating scale in some way before the test material is presented will force them to change their judgements in an unsatisfactory manner [33]. Some attempt to correct for variation between observers is made later in the statistical analysis of the results, details of which are in Appendix C.

Once all results have been collected the results for $A$ and $B$ are first measured on a scale of 0 (bottom of the “Bad” range) to 100 (top of “Excellent”) then the
difference between the scores is taken. From the distribution of these difference values between observers we can calculate the difference in subjective quality between $A$ and $B$ for the "average" user. As there is often a large spread of values for the subjective quality difference, there may be a significant proportion (but not a majority) of viewers who did not agree with the perception of the "average" viewer that sequence $A$ was better than sequence $B$ (or vice versa). To give an indication of the spread of values for the subjective difference we will also quote the proportions of viewers who fell into each of the three following categories (on a test-by-test basis):

1. Sequence $A$ was better than sequence $B$
2. The two sequences $A$ and $B$ were indistinguishable
3. Sequence $B$ was better than sequence $A$

The standard DSCQS scoresheet (Figure 3.18) does not require the viewer to explicitly mark presentations where $A$ and $B$ were "indistinguishable" and given the marks on the scales were entered by hand we should not rely on the marks coinciding exactly. We define "indistinguishable" to mean that the marks on the scales were within 2mm of each other (the full length of each scale is 109mm on the scoresheets given to the observers). Note that this does not affect the analysis of the perception of the "average" observer.

The tests here deviate from Recommendation 500 in that we are testing video that has been heavily compressed and contains significant artifacts whereas the standard has been largely used to test small magnitude impairments. Rec. 500 advocates the use of the original (uncompressed) sequence as either $A$ or $B$ (chosen in a pseudorandom manner) to act as a reference (Figure 4 of [32]). For heavily degraded sequences, we do not feel that explicit comparison with the original will produce useful results therefore for the tests conducted here $A$ and $B$ will be the same sequence compressed using two different techniques. For comparison of three or more techniques (as will be done in Chapter 5) if we
### 3.5 Experimental Procedure

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Base Layer Rate (kbps)</th>
<th>Total Enhancement Rate (kbps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Akiyo</td>
<td>32</td>
<td>64</td>
</tr>
<tr>
<td>Mother &amp; Daughter</td>
<td>56</td>
<td>112</td>
</tr>
<tr>
<td>Carphone</td>
<td>100</td>
<td>200</td>
</tr>
<tr>
<td>Foreman</td>
<td>130</td>
<td>260</td>
</tr>
</tbody>
</table>

Table 3.3: Subjective test conditions

Test all combinations of the techniques against each other then we can use any of those techniques as a reference for comparison with the other techniques. For example, if we wish to compare the performance of techniques α, β and γ on a particular input sequence we would perform three tests where (A, B) (or (B, A), the order of the sequences is not significant) are: (α, β), (α, γ) and (β, γ). To compare the performance of technique β to the other techniques α and γ we can calculate the mean subjective difference from the first test and the third test (the results of one test will need to be inverted if the reference is A in one test and B in the other) which can both then be plotted in the one graph. If an appropriate confidence interval for each mean score is calculated and the results for one or both of the techniques α, γ do not intercept the zero axis then we can say that the average observer finds β better or worse (depending on which side of the zero axis the mean lies) than α/γ to that level of confidence.

A single set of subjective tests were conducted for this thesis, some of the results will be presented in this chapter and the remainder in Chapter 5. Here we look only at the comparison between MPEG-2-compliant SNR scalability (using the pyramid encoder of Figure 3.4) and MPEG-4 FGS. All tests were conducted at constant bit rate using the perceptual rate control algorithm to be described in Chapter 5 except for the FGS enhancement data which was truncated to meet the target number of bits for that frame (Section 3.4). For MPEG-2-compliant SNR scalability, four enhancement layers were used. All tests are for CIF-sized (352x288 pixels) inputs at 10fps, the sequences and bit rates used are shown in
3.6 Results

Table 3.3.

**Important Note:** The implementation of “MPEG-2-Compliant SNR Scalability” used throughout this thesis differs to that described in MPEG-2 in two ways. While the overall structure of the encoder and decoders is the same, the underlying implementation is that of MPEG-4 (bitstream syntax, prediction, H.263 quantization etc.). Secondly, the differential quantizer step size that is coded in the bitstream (section 2.4.1) is predicted from the corresponding value in the previous layer, a technique with is not used in either MPEG-2 nor MPEG-4. This is done here to reduce the amount of macroblock overhead when using perceptual quantization algorithms at low bit rates, a topic to be discussed in Chapter 5.

3.6 Results

Figures 3.19 and 3.20 show rate-distortion curves and individual frame PSNRs for the three types of scalability and single-layer for “Mother & Daughter” and “Foreman” respectively. As discussed in the previous section, the half-pel motion estimation is done using the top layer frame store and as such the quality of the base layers for the scalable coders will not be the same as the single-layer case with the same quantizer step size. This is due to different motion vectors being calculated in each case. For “Mother & Daughter” (and similar sequences such as “Akiyo”) the overall sequence PSNR depends greatly on the coding of static areas in the first frame. A lower quantizer step size was used in the coding of the first frame in all the layered coding schemes to compensate for the larger dead-zone in the enhancement layers (which use inter-mode H.263 quantization exclusively) and was adjusted to give approximately uniform PSNR over the entire sequence.

The main point to notice is that the performance of FGS is largely the same as the single-loop case and is significantly worse than the multi-loop schemes. For
3.6 Results

Constant Quantizer Results for Mother & Daughter (CIF 10fps)

(a) Comparative performance

Enhancement Layer Frame PSNRs for Mother & Daughter (CIF 10fps)

(b) Video quality in enhancement layers

Figure 3.19: Constant quantizer results for “Mother & Daughter” (CIF 10fps)
3.6 Results

Constant Quantizer Results for Foreman (CIF 10fps)

(a) Comparative performance

(b) Video quality in enhancement layers

Figure 3.20: Constant quantizer results for “Foreman” (CIF 10fps)
### 3.6 Results

<table>
<thead>
<tr>
<th>Layer</th>
<th>PSNR (dB)</th>
<th>Rates (kbps)</th>
<th>Coefficients</th>
<th>Layer PSNR (dB)</th>
<th>MB Overhead</th>
<th>DCT Coeffs</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single-layer</td>
<td>31.95</td>
<td>78.5</td>
<td>86.2</td>
<td>164.7</td>
<td>125788</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

**3-layer MPEG-2-Compliant SNR Scalability (pyramid)**

| Base (QP=31) | 27.64 | 19.8 | 15.2 | 90.0 | 21325 | - | - | - | - |
| 1st enh. (QP=16) | 30.44 | 14.8 | 40.7 | 55.5 | 69592 | 7164 | - | - | - |
| 2nd enh. (QP=12) | 31.88 | 16.4 | 49.0 | 65.4 | 122609 | 13782 | 1315 | - | - |
| **Total** | 51.0 | 104.9 | 210.9 | 154118 | 154118 (137706 discrete) | - | - | - | - |

**5-layer MPEG-2-Compliant SNR Scalability (pyramid)**

| Base (QP=31) | 27.59 | 19.8 | 15.5 | 90.4 | 21774 | - | - | - | - |
| 1st enh. (QP=19) | 29.65 | 13.3 | 27.4 | 40.7 | 54522 | 4869 | - | - | - |
| 2nd enh. (QP=15) | 30.79 | 14.5 | 34.7 | 49.9 | 115178 | 20766 | 1327 | 46 | - |
| 3rd enh. (QP=13) | 31.54 | 13.7 | 25.7 | 39.4 | 132053 | 28196 | 2248 | 104 | 5 |
| **Total** | 76.5 | 136.5 | 268.1 | 195630 | 195630 (162606 discrete) | - | - | - | - |

**3-layer Single-loop Scalability (non-MPEG-2-compliant)**

| Base (QP=31) | 27.73 | 19.8 | 15.0 | 90.1 | 20756 | - | - | - | - |
| 1st enh. (QP=16) | 29.72 | 22.9 | 102.7 | 125.6 | 190830 | 649 | - | - | - |
| 2nd enh. (QP=12) | 7219 | 8481 | 7482 | 28591 | 17992 | 8900 | 8900 | 0 | 5 |
| **Total** | 46627 (23182 discrete) | 91725 (55483 discrete) | 1.45 | - | - | - | - | - | - |

**Table 3.4: Bitstream statistics for Foreman (CIF 10fps)**

<table>
<thead>
<tr>
<th>Layer</th>
<th>Macros</th>
<th>Blocks</th>
<th>Norm. Scan Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single-layer</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>QP=12</td>
<td>21617</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

**3-layer MPEG-2-Compliant SNR Scalability (pyramid)**

| Base (QP=31) | 8228 | - | - | - | - | 13973 | - | - | - | 0.10 |
| 1st enh. (QP=16) | 10128 | 8010 | - | - | - | - | 26040 | 11855 | - | 0.53 |
| 2nd enh. (QP=12) | 7219 | 8481 | 7482 | - | - | - | 28591 | 17992 | 8900 | 373136 | 2021 | 5 | 0.82 |
| **Total** | 46627 (23182 discrete) | 91725 (55483 discrete) | 1.45 | - | - | - | - | - | - | - | - |

**5-layer MPEG-2-Compliant SNR Scalability (pyramid)**

| Base (QP=31) | 8466 | - | - | - | - | 14315 | - | - | - | 0.10 |
| 1st enh. (QP=19) | 8465 | 7674 | - | - | - | - | 21958 | 10099 | - | 0.35 |
| 2nd enh. (QP=15) | 7032 | 6990 | 6808 | - | - | - | 24791 | 14827 | 6640 | - | - | 0.51 |
| 3rd enh. (QP=13) | 5626 | 5833 | 6046 | 6181 | - | - | 24832 | 17573 | 9875 | 4639 | 0.59 |
| 4th enh. (QP=12) | 4679 | 5028 | 4804 | 5262 | 5276 | 23869 | 18534 | 11373 | 6549 | 2743 | 0.46 |
| **Total** | 76575 (25049 discrete) | 134967 (63068 discrete) | 2.01 | - | - | - | - | - | - | - | - |

**3-layer Single-loop Scalability (non-MPEG-2-compliant)**

| Base (QP=31) | 8010 | - | - | - | - | 13469 | - | - | - | 0.10 |
| 1st enh. (QP=16) | 20835 | 7900 | - | - | - | - | 70984 | 10747 | - | 1.23 |
| 2nd enh. (QP=12) | 5637 | 19297 | 7786 | - | - | - | 54482 | 48846 | 8711 | - | - | 1.81 |
| **Total** | 67589 (32720 discrete) | 178307 (112039 discrete) | 3.14 | - | - | - | - | - | - | - | - |

Note: The parameters used for these tests are not the same as for those in Figure 3.20.
<table>
<thead>
<tr>
<th>Layer</th>
<th>PSNR (dB)</th>
<th>Rates (kbps)</th>
<th>Coefficients</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>MB Overhead</td>
<td>DCT Coefs</td>
</tr>
<tr>
<td>Single-layer</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>QP=10</td>
<td>36.40</td>
<td>13.1</td>
<td>22.8</td>
</tr>
<tr>
<td>3-layer MPEG-2-Compliant SNR Scalability (pyramid)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Base (QP=31)</td>
<td>31.09</td>
<td>10.0</td>
<td>2.5</td>
</tr>
<tr>
<td>1st enh. (QP=13,11)</td>
<td>34.86</td>
<td>9.3</td>
<td>13.6</td>
</tr>
<tr>
<td>2nd enh. (QP=10,8)</td>
<td>36.26</td>
<td>10.3</td>
<td>14.6</td>
</tr>
<tr>
<td>Total</td>
<td>29.6</td>
<td>30.7</td>
<td>78.2</td>
</tr>
<tr>
<td>5-layer MPEG-2-Compliant SNR Scalability (pyramid)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Base (QP=31)</td>
<td>31.10</td>
<td>10.1</td>
<td>2.5</td>
</tr>
<tr>
<td>1st enh. (QP=15,11)</td>
<td>34.26</td>
<td>8.5</td>
<td>10.2</td>
</tr>
<tr>
<td>2nd enh. (QP=12,10)</td>
<td>35.17</td>
<td>9.0</td>
<td>9.7</td>
</tr>
<tr>
<td>3rd enh. (QP=11.9)</td>
<td>35.75</td>
<td>9.2</td>
<td>9.4</td>
</tr>
<tr>
<td>4th enh. (QP=10,8)</td>
<td>36.29</td>
<td>9.2</td>
<td>9.9</td>
</tr>
<tr>
<td>Total</td>
<td>46.0</td>
<td>41.7</td>
<td>105.6</td>
</tr>
<tr>
<td>3-layer Single-loop Scalability (non-MPEG-2-compliant)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Base (QP=31)</td>
<td>31.24</td>
<td>10.2</td>
<td>3.0</td>
</tr>
<tr>
<td>1st enh. (QP=13,11)</td>
<td>34.05</td>
<td>21.6</td>
<td>78.1</td>
</tr>
<tr>
<td>2nd enh. (QP=10,8)</td>
<td>35.23</td>
<td>20.5</td>
<td>58.3</td>
</tr>
<tr>
<td>Total</td>
<td>52.3</td>
<td>139.4</td>
<td>208.1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Layer</th>
<th>Macroblocks</th>
<th>Blocks</th>
<th>Norm. Scan</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1x</td>
<td>2x</td>
<td>3x</td>
<td>4x</td>
</tr>
<tr>
<td>Single-layer</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>QP=10</td>
<td>11207</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>3-layer MPEG-2-Compliant SNR Scalability (pyramid)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Base (QP=31)</td>
<td>1886</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>1st enh. (QP=13,11)</td>
<td>6961</td>
<td>1879</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>2nd enh. (QP=10,8)</td>
<td>5986</td>
<td>5254</td>
<td>1610</td>
<td>-</td>
</tr>
<tr>
<td>Total</td>
<td>21324</td>
<td>23650</td>
<td>(12850 discrete)</td>
<td>-</td>
</tr>
<tr>
<td>5-layer MPEG-2-Compliant SNR Scalability (pyramid)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Base (QP=31)</td>
<td>1890</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>1st enh. (QP=15,11)</td>
<td>5603</td>
<td>1867</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>2nd enh. (QP=12,10)</td>
<td>5677</td>
<td>3855</td>
<td>1448</td>
<td>-</td>
</tr>
<tr>
<td>3rd enh. (QP=11,9)</td>
<td>5102</td>
<td>4143</td>
<td>2845</td>
<td>1146</td>
</tr>
<tr>
<td>4th enh. (QP=10,8)</td>
<td>4403</td>
<td>4230</td>
<td>2967</td>
<td>2235</td>
</tr>
<tr>
<td>Total</td>
<td>35429</td>
<td>4780</td>
<td>(14780 discrete)</td>
<td>-</td>
</tr>
<tr>
<td>3-layer Single-loop Scalability (non-MPEG-2-Compliant)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Base (QP=31)</td>
<td>1797</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>1st enh. (QP=13,11)</td>
<td>25470</td>
<td>1787</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>2nd enh. (QP=10,8)</td>
<td>5294</td>
<td>22372</td>
<td>1725</td>
<td>-</td>
</tr>
<tr>
<td>Total</td>
<td>55213</td>
<td>29391</td>
<td>(29391 discrete)</td>
<td>-</td>
</tr>
</tbody>
</table>

Note: The parameters used for these tests are not the same as for those in Figure 3.19.

Table 3.5: Bitstream statistics for Mother & Daughter (CIF 10fps)
3.6 Results

High motion sequences such as “Foreman” the performance difference between the two classes of codec is smaller. Here the gain associated with improved prediction is small compared to the overall magnitude of the prediction difference, due to the constantly-changing nature of the content of the scene.

Tables 3.4 and 3.5 show bitstream statistics for three- and five-layer codecs with the same values for QP in the bottom and top layers. For “Mother & Daughter” the value for QP used in the first frame (the only I frame in the sequence) is shown in the table next to the symbol “[I]”. Again, it is important to note that the parameters used here are not the same as those used for Figures 3.19 and 3.20 and as such no direction comparisons should be made. There is a significant drop in quality for the single-loop coder despite the use of the same value for QP in the top layer, a point which will be discussed further in the next section. The number of non-zero coefficients, blocks and macroblocks coded are counted in each layer and grouped by the number of times these entities have been processed so far. For example, a block in the first enhancement layer (second layer overall) that is listed in the column marked “2x” has been scanned for coefficients in both layers, whereas those marked as “1x” have only been scanned in one layer or the other. Two totals are provided for each: the overall number is the sum of the counts for the topmost layer, weighted by the number of times the entity has been processed, followed by a count of the number of “discrete” entities that are present which is the same sum without the weighting applied. The overall count gives an indication of the volume of symbols that must appear in the bitstreams for all layers to code these entities and the numbers shown here partially explain the large increase in overall bit rate for the layered coding schemes. The discrete count has been included to show that the use of layered coding does not simply split the coding of the symbols present in the single-layer case over the multiple layers present. The use of layered coding generates non-zero coefficients that do not exist in the single-layer case and are often located in blocks and macroblocks that are all zero in the single-layer
Comparing the three-layer and five-layer cases that have the same base layer and top enhancement layer quantizer step sizes we see that the number of coefficients is about one-third higher overall for the five-layer case without any change to the PSNR at the top layer. This indicates that the processes that generate these extra coefficients depend on the number of layers present and not simply the content or amount of motion in the scene.

Figures 3.21 and 3.22 show the results of the frame store mismatch test where the two top enhancement layers of a five layer pyramid coder were lost after 20 frames and the remaining three layers decoded. As predicted in Section 3.2.1, the loss of quality is not catastrophic in this case as has been observed elsewhere for the single-loop encoder.

Figure 3.23 shows the results of the subjective comparisons between MPEG-2-compliant SNR scalability (the reference condition) and MPEG-4 FGS for a full
3.6 Results

MPEG-2 SNR Scalability: Drift after loss of top two layers starting at frame 20 (Foreman)

Figure 3.22: Loss due to drift (Foreman)

<table>
<thead>
<tr>
<th>Sequence</th>
<th>MPEG-2 SNR scal. better</th>
<th>Indistinguishable</th>
<th>FGS better</th>
</tr>
</thead>
<tbody>
<tr>
<td>Akiyo</td>
<td>94.7%</td>
<td>5.3%</td>
<td>0%</td>
</tr>
<tr>
<td>Mother &amp; Daughter</td>
<td>84.2%</td>
<td>5.3%</td>
<td>10.5%</td>
</tr>
<tr>
<td>Foreman</td>
<td>42.1%</td>
<td>36.8%</td>
<td>21.1%</td>
</tr>
<tr>
<td>Carphone</td>
<td>42.1%</td>
<td>21.1%</td>
<td>36.8%</td>
</tr>
</tbody>
</table>

Table 3.6: Spread of subjective test results

<table>
<thead>
<tr>
<th>Akiyo</th>
<th>Mother &amp; Daughter</th>
<th>Foreman</th>
<th>Carphone</th>
</tr>
</thead>
<tbody>
<tr>
<td>99.9998%</td>
<td>99.93%</td>
<td>90.93%</td>
<td>20.14%</td>
</tr>
</tbody>
</table>

Table 3.7: Confidence that MPEG-2-compliant SNR scalability (reference) is subjectively better for the average user
Figure 3.23: Subjective comparison of MPEG-2 SNR Scalability and FGS (95% confidence intervals)
3.7 Discussion

range of four different input sequences. Negative values mean subjective performance that is worse than the reference condition i.e. MPEG-4 FGS is subjectively inferior to MPEG-2-compliant SNR scalability. The units on the vertical scale correspond to the scoresheet of Figure 3.18 being measured from 0 to 100 thus a difference of 20 corresponds to one of the five subjective categories marked on the scoresheet. This is the mean score for the “average” observer along with a 95% confidence interval. While direct comparison between results for different sequences is not useful in itself, all results are presented here in the one figure to show the general trend in the subjective results is the same as shown in the other tests: for low motion sequences such as “Akiyo” and “Mother & Daughter” MPEG-2-compliant SNR scalability performs significantly better than MPEG-4 FGS at similar bit rates. Table 3.6 shows the spread of results between for the same tests shown in Figure 3.23. The central limit theorem states that for a large number of samples the distribution of the mean value here will be approximately normal so we can also calculate given the mean and standard deviation for the “average” observer a likelihood that the mean value will be negative which is shown in Table 3.7. For those tests that are a long way from the zero axis we can state with a very high degree of confidence that the “average” observer will find MPEG-2-compliant SNR scalability to be subjectively superior to MPEG-4 FGS.

3.7 Discussion

At this point we can now look back and summarize the respective strengths and weaknesses of FGS versus multi-loop SNR scalability, leaving aside the single-loop case which has no advantage over either of these schemes.

3.7.1 Critique of FGS

The FGS method has a number of attractive features:
3.7 Discussion

- The encoder complexity of FGS is considerably lower than the multi-loop SNR scalability schemes since motion-compensated prediction is only done once, there is only one frame store and there is a single DCT used (and no IDCT) when generating the enhancement bitstream. For some applications such simple encoders are desirable.

- Recovery from errors in the enhancement layer is instantaneous since the texture there is never stored for use in future prediction.

- Rate control for the enhancement layer is not an issue since the bitstream can be truncated at any point. Rate control in most other methods requires extra overhead for changing the quantizer step size over the frame, which is not required for FGS. Many rate control algorithms also add significantly to encoder complexity. This is especially useful if the amount of available bandwidth to each client is guaranteed.

At the same time there are a number of drawbacks:

- Where no bandwidth or packet loss rate guarantees exist the base layer must be protected from errors by using packet prioritization or by introducing redundancy. The enhancement layer bit-plane texture cannot be used if the base layer has been lost. Neither FGS nor the pyramid coder can be used in networks that suffer significant packet loss but where prioritization is not available.

- Except when it is used with very high base layer bit rates, the performance of FGS is poor; often close to or even worse than two or three layer simulcast ([27], especially Figure 21 therein). This is the price paid for not using enhancement texture in prediction (which is what also gives FGS its excellent error recovery behaviour). PFGS improves performance at the expense of additional complexity.
3.7 Discussion

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Rates (kbps)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Base</td>
</tr>
<tr>
<td>Akiyo</td>
<td>14.9</td>
</tr>
<tr>
<td></td>
<td>20.3</td>
</tr>
<tr>
<td>Foreman</td>
<td>87.1</td>
</tr>
<tr>
<td></td>
<td>111.7</td>
</tr>
<tr>
<td>Mother &amp; Daughter</td>
<td>28.8</td>
</tr>
<tr>
<td></td>
<td>33.1</td>
</tr>
</tbody>
</table>

Table 3.8: FGS bit-plane spacing

- Due to its second IDCT the FGS decoder has significantly higher computational complexity than the MPEG-2 SNR scalable decoder which will make it unattractive for some applications.

- Selective enhancement, as previously described, generates extra overhead when performing perceptual coding.

- The real level of "granularity" achieved by FGS is questionable. While the proponents of FGS will show that arbitrary bitstream truncation can lead to a very fine spread of possible PSNRs, the subjective value in any small increment of extra received FGS texture is certainly much less than a full layer of texture in the case of regular SNR scalability. The use of PSNR in this case as the sole measure of quality can be misleading. For example, comparing two frames where the only difference is the enhancement of one row of macroblocks will indeed result in different PSNR measurements but it is unlikely to be noticeable subjectively. In fact, the quality difference across a partially-enhanced frame may be distracting to the viewer. Furthermore, environments with significant packet loss that provide prioritization cannot limit the amount of data that is discarded when errors occur to less than one bit-plane, as originally discussed in Section 3.4.1.
From the last point it can be argued that the true granularity of FGS does not always extend below that of a single bit-plane. Table 3.8 lists the bit rates of the first four FGS bit-planes for three different sequences at two different base layer bit rates. If we take it to be true that the granularity of FGS does not extend below the bit-plane level it can be seen that the layers are very inconveniently spaced (approximately exponential spacing in the lower layers) and often so far apart that receiving two full planes/layers is not likely.

The bit rates for each bit-plane shown in Table 3.8 were measured without the use of selective enhancement or frequency weighting which can be used to split these large planes into a number of separate planes to increase granularity. The creation of extra planes has a negative effect on coding performance since these require extra scanning passes and results in an increase in the overall sparseness of non-zero bit-plane entries, an issue to be discussed further in Section 3.7.3.1.

### 3.7.2 Critique of Multi-Loop SNR Scalability

Similarly, the MPEG-2 compliant multi-loop SNR scalability has its advantages,

- For error-free operation the bandwidth efficiency of SNR scalability is much higher than FGS for many sequences due to the presence of prediction in the enhancement layers.

- Layering in SNR scalability can be done in a much more flexible way than FGS which is constrained by the locations of the bit-planes.

- The single-loop decoder (Figure 3.1) has very low complexity, using only one IDCT and one motion-compensated prediction block, regardless of the number of layers.

and its disadvantages,
• Random errors introduced in enhancement layers may need to be concealed and are unlikely to be completely corrected until the next I frame. The frame store mismatch experiments conducted here show that the loss of complete enhancement layers, even between I frames, does not lead to significant problems.

• Performance degrades significantly as more layers are added. The reasons behind this degradation are to be discussed in detail in the next section.

• Encoder complexity is significantly higher than FGS, requiring one frame store, motion-compensation unit and a DCT and IDCT per layer.

• Rate control is a challenge, especially for real-time applications that cannot tolerate additional processing delay (see Chapter 5).

3.7.3 Comments on Performance of Scalable Video Coders

Ideally we would like to construct a scalable video system which has performance as close as possible to a “single” layer. Certainly there should be no expectation that splitting into multiple layers somehow makes the whole system more efficient. The prediction in the lower layers is poorer (if indeed there is any at all) and the quantized texture in each layer is coded in essentially the same way in all cases (Huffman-style coding of run/level/end tuples for the non-zero DCT coefficients). Indeed from the results shown in Section 3.6, it can be seen that the methods described here induce a significant performance degradation compared to single-layer at the same total rate. This section describes some of the reasons why this should be the case.

3.7.3.1 Coefficient Sparseness

In the best possible case, we might expect a multi-layer system to generate the same number of non-zero DCT coefficients as a single-layer system (putting
3.7 Discussion

aside for the time being the results shown in Tables 3.4 and 3.5). The overall bit rate is approximately proportional to this coefficient count if we remove the effect of motion vectors, which should be the same in both the scalable and single-layer cases. However, we should still expect the scalable system to require significantly more bits to code since we are partitioning that same number of coefficients over more than one layer. This is because blocks need to be scanned multiple times.

Given a frame with $N$ blocks some subset $n \leq N$ is coded in the single-layer case, now when split over $p$ layers we would expect that coefficients that are coded in the same block in the single-layer case are now coded in separate (co-incident) blocks in different layers. Since we have not affected the frequency content of the original frame we would expect the coefficients to be in roughly the same places along the zig-zag scan however the runs of zeros between non-zero coefficients is longer on average since in many cases coefficients that existed in between others have "moved" to other layers. This results in a change in VLC statistics that cannot be "solved" by simply creating a new VLC table. The total entropy is higher than before despite having the same number of symbols to code.

For progressive still image compression, e.g. JPEG [9] this problem can largely be solved by starting enhancement layer scans where the lower layer scan stopped. However, due to prediction, there is no guarantee that new coefficients do not appear in amongst the coefficients that have already been scanned in the base layer.

Wilson et al [34] propose an arithmetic coding scheme for the coding of enhancement layer coefficients for two-layer SNR scalable coders which overcomes the problem of using the same VLC table in both layers, however, this does not address the sparseness problem. The reduction in bit rate from this scheme is typically less than 10%. Note also that the encoder architecture used in this work is that of Figure 3.2 which is subject to drift.
3.7.3.2 Quantization and Prediction Effects

The SNR scalable coders of Figures 3.2, 3.3, 3.4 and 3.5 form residues in each enhancement layer from the quantization error in the previous layer. Significant errors can accumulate when quantization is done in this way.

Consider a coefficient that is large enough to be non-zero after quantization in a lower layer with a large quantizer step size. Although the coefficient is now non-zero, the quantization error is still large because of the coarseness of the quantization. Given that enhancement layer quantizers tend to be closely spaced due to the non-linear relationship between quantizer step size and bit rate, it is often the case that such coefficients are never refined in higher layers since the residual error is not large enough to create another non-zero quantized coefficient.

How these quantization effects manifest themselves depend on the prediction structure used. For the single-loop case where no enhancement layer prediction is done, Tables 3.4 and 3.5 show a significant drop in PSNR at the top layer compared to single-layer with the same value of QP. Significant coefficients remain coarsely quantized at the top layer and this impacts adversely on quality. While this in itself should decrease the overall number of coefficients coded, the lack of prediction in these upper layers requires that other coefficients be coded and therefore the overall number can be seen to be significantly higher than the corresponding single-layer bitstream. For the pyramid multi-loop encoder, the PSNR drop in the top layer is greatly reduced as some of the "refinement" is done by the prediction added in each layer.

One very important effect that exists only in the single-layer case is that as the bit rate increases for many sequences the quality of the prediction improves to such an extent that coefficients that were needed at low qualities are no longer required. For SNR scalability, the only way to remove a coefficient is to generate another coefficient in an enhancement layer with the opposite sign thus creating two coefficients where there are none in the single-layer comparison. This partly
Figure 3.24: Typical single-layer rate-distortion profile

explains the significantly higher number of discrete coefficients that exist for the scalable coding schemes in Tables 3.4 and 3.5, compared to the single-layer case.

These prediction effects can be appreciated by considering the standard single-layer rate-distortion relationship for video coders with prediction as shown in Figure 3.24. At low bit rates the effect of each increment of new texture coded is much higher than the same increment when then quality is already high. This explains the decreasing slope of the curve as rate increases. New texture not only improves the quality of the current frame but can often also provide better prediction in subsequent frames thus magnifying its effect. This can cause texture to disappear in subsequent frames and those bits can be “re-allocated” to improve quality still further while using the same number of bits. The FGS enhancement bit-planes have a linear rate-distortion profile precisely because there is no prediction involved. Scalable video must create a low-quality version of the sequence being coded and as such cannot take advantage fully of these effects. Indeed, coefficients may disappear as we add layers but they have al-
3.8 Other Approaches to Scalable Video

ready been coded in the lower layers and it is not possible to recover the bits used to code those in any way.

3.8 Other Approaches to Scalable Video

This final section details four other schemes which address the scalable video problem. The first two are closely-related to the MPEG SNR scalable systems we have already described while the last two show some completely different solutions. These are mentioned here for the sake of completeness but will not be referred or compared to in later chapters.

3.8.1 Conditional Replenishment

While the multi-loop SNR scalable encoders of Figure 3.3 and Figure 3.4 offer clearly superior performance to the single-loop encoder of Figure 3.5, it has been shown ([35] and later in [25]) that it is possible to do better with a slightly modified encoder.

Denote $X_{diff}$ as the DCT domain difference signal that is to be quantized in the enhancement layer in the multi-loop schemes. It can be easily shown that

$$X_{diff} = \hat{X}_0 - \hat{X}_1 - Q_{base}(X - \hat{X}_0)$$

where $X$ is the DCT of the original signal, $\hat{X}_0$ is the DCT of the base layer prediction, $\hat{X}_1$ is the DCT of the enhancement layer prediction and $Q_{base}(x)$ is the reconstructed value of the base layer quantizer for a DCT coefficient with value $x$. In the case where the base layer DCT coefficient is non-zero, the enhancement layer is coding the quantization noise that is left over from the base layer, which we would expect to be uncorrelated with anything in the frame store. When the base layer quantized coefficient is zero, however, we are coding the full signal in the enhancement layer where we might expect significant correlation to exist with the contents of the frame store. This leads to the conclusion that where
3.8 Other Approaches to Scalable Video

the base layer quantizer results in a non-zero coefficient, we should avoid the use of the enhancement layer prediction.

Figure 3.25 shows an encoder with a conditional replenishment block added which bypasses the enhancement layer prediction when the quantized base layer signal is non-zero. This has been shown in the previous work to lead to significant performance gains.

The main drawback to this scheme is one of computational complexity in the encoder and especially the decoder as can be seen in Figure 3.26. For each layer in the system we need individual IDCTs per layer after texture decoding, another inverse quantization block in each layer and another three DCT/IDCT units per layer for performing conditional replenishment on the previous layer.

3.8.2 Bitstream Decomposition

Another approach to creating multiple bitstreams from a single video sequence is to split a standard single-layer bitstream into a number of parts.

The work described in [36] and [37] partitions the coefficients from the original single-layer by frequency band (known therein as spectral selection) as well as by bit-plane (successive approximation). To avoid prediction drift in the case where some of all of the enhancement layers are not received, only the base layer texture is used in the frame store. The performance of this scheme should be similar to that of the single-loop scheme of Figure 3.5 given that there are only low quality frame store pictures from which prediction is made. One advantage compared to encoders such as the two-loop case shown in Figure 3.3 is that the signal is only quantized once so the problems described in 3.7.3.2, where significant coefficients in the second layer are not coded because they are contained in the quantizer dead-zone, do not occur. Two IDCTs are required at both the encoder and the decoder since the portion of the received texture that is to go into the frame store must be recovered and added separately from the base layer texture. This number of IDCT operations is independent of the number of
3.8 Other Approaches to Scalable Video

Figure 3.25: Two-layer conditional replenishment encoder
Figure 3.26: Two-layer conditional replenishment decoder
layers the enhancement texture is split into.

[38] describes another approach where the full texture of both sections of a decomposed single-layer stream are used for prediction and a third drift correction signal is sent which can be used in conjunction with the base layer should the enhancement layer be lost. This approach has lower computational complexity than the previous decomposition scheme since there does not need to be a second IDCT outside the motion-compensation loop in the decoder. The drift correction signal is applied in the same way as an enhancement layer in MPEG-2 SNR scalability (Figure 3.1).

3.8.3 Multiple Description Coding

The schemes described so far create layers that are prioritized; if a subset $n$ of $p > n$ original layers is received, only the subset of the $n$ highest priority layers can be decoded successfully. None of these schemes can do anything with any enhancement layers if the base layer is lost. In the case of FGS, each bit-plane contributes more to the decoded image quality than all higher bit-planes combined so if these planes are to be sent over a channel with random packet loss, lower planes need to be protected more (this is not necessary if only bitstream truncation takes place since data is sent in priority order). Transmission of such video over the Internet is, in general, impractical since packet prioritization must be supported in all routers between the source of the video and the decoder. Widespread adoption of new protocols such as IPv6 may eventually solve this problem but there has been no significant roll-out to date of such technology, despite other issues such as a shortage of addresses demanding an update from the current IPv4.

Multiple description coding creates two (or more) layers where any individual layer can be decoded successfully and a higher quality service can be recovered if more layers are received. One approach to this [39] is to take quantized DCT coefficients generated by a standard encoder and to split these using
a multiple description scalar quantizer which maps individual quantized values into one value for each channel (values in bold type below):

\[
\begin{array}{ccccccc}
\ldots & -3 & -2 & -1 & 0 & 1 & 2 & 3 & \ldots \\
\vdots & \ddots & \\
-3 & -6 & -5 \\
-2 & -4 & -3 \\
-1 & -2 & -1 \\
0 & 0 \\
1 & 1 & 2 \\
2 & 3 & 4 \\
3 & 5 & 6 \\
\vdots & \ddots \\
\end{array}
\]

i.e. with a value of 1 received on the first channel (left-hand column) we know the quantized value must be either 1 or 2; the other channel (top row) will be either 0 or 1 which if received will remove this uncertainty.

An alternative to generating multiple descriptions for individual coefficients has been proposed whereby pairs of coefficients are similarly split [40]. Results for this technique on still images show this can be done with around 20% more bits than for the standard single description case.

In the case of video, intra-coded frames can suffer from moderate amounts of random packet loss in any description since any cases where all descriptions are lost will not affect future frames. However, the use of temporal prediction has the same problems with frame store mismatch or drift as the MPEG-style systems. For low rates (< 10%) of packet loss [39] shows that multiple description coding and concealment techniques alone perform well. For prolonged loss of one description or the other drift in the frame store becomes an issue; [41] describes a system that uses separate frame stores that contain only single descriptions for controlling decoder drift.
3.8 Other Approaches to Scalable Video

3.8.4 3D Subband Coding

In recent years subband and specifically wavelet methods have largely displaced the DCT for still image coding [42]. While much effort is currently being spent on developing hybrid wavelet-MCP video schemes which replace the DCT for frame texture coding and leave existing motion-compensated prediction schemes for inter-frame prediction, other researchers are pursuing extensions of 2D wavelet and subband methods into the temporal dimension. This has some attractive features for scalable video coding.

The first stage of a subband image or video coder is to apply one-dimensional filters to the input to separate the low-pass (LP) and high-pass (HP) components. These filters are applied recursively on the low-pass components to split these further since this is where the significant information is held. Figure 3.27 shows
3.8 Other Approaches to Scalable Video

A typical arrangement for the encoder analysis filter bank [43] of a 3D subband video coder that groups several successive frames together for coding. Six different filters are used: low \((LP_t)\) and high pass \((HP_t)\) in the temporal direction plus two each for the horizontal and vertical spatial dimensions \((LP_h, HP_h, LP_v, HP_v)\). Early schemes code the resulting coefficients using vector quantization [44]; later wavelet methods use extensions to the existing zerotree [45] and SPIHT [46,47] techniques to exploit commonality between representations of the same image at the different scales and to prioritize coefficients as they are put into a bitstream.

Partitioning into spatio-temporal bands is attractive for scalable coding since the omission of some bands does not cause decoder drift as is the case with the hybrid systems previously described when not all texture is received. MPEG-4 FGS avoids this problem by limiting the amount of texture that is in the motion loop. However, this is done at the expense of decreased coding efficiency.

For applications such as videoconferencing, where there is typically very little motion between frames, the framework described performs adequately. A number of schemes exist [48,49] which shift the pixels in the input frames using standard motion-compensated prediction in order to increase temporal similarity between frames to be coded. This is not as straightforward as it might seem. Since the frames are not coded independently, the fact that the prediction of one frame generally uses only a subset of the previous frame's pixels (since motion is non-uniform across a frame) causes problems since there is no longer a one-to-one correspondence between pixels in the two frames and as such some may not otherwise be coded.

For some applications the extra complexity, memory and latency requirements associated with processing groups of several frames (typically two or four but possibly more) at once is unacceptable. [49] and [50] shows results comparable with single-layer H.263 and MPEG-1 while offering scalability features which as we have seen come at significant extra cost.
3.9 Conclusion

This chapter has outlined the various forms of scalable video coding that are available in the current MPEG-2 and MPEG-4 video standards. For transport media that lack prioritization, none of these schemes work well due to their strictly hierarchical nature. Multiple description coding, which is not part of any current MPEG standard, is useful in this case. Where prioritization does exist, it has been shown that MPEG-4 Fine Granularity Scalability has poor performance compared to multi-loop MPEG-2-compatible SNR scalability due to its lack of prediction above the base layer. While FGS has excellent error recovery properties, it has been shown that the effect of frame store mismatch due to lost enhancement layer information in the MPEG-2 case is minimal if a multi-loop encoder is used.

MPEG-2 SNR scalability therefore provides an excellent basis for new methods such as stream morphing which will be introduced in the following chapters.
Chapter 4

Stream Morphing

4.1 Introduction

MPEG-2-compliant SNR scalability and MPEG-4 FGS are two examples of what we might choose to call *scalability by texture summation*. The common feature of these schemes is that enhancement layer data is decoded to form additional texture which is added to that already received. Indeed these two methods are quite similar: MPEG-4 FGS differs primarily in the fact that it fixes the quantization step sizes to be powers of two and prevents drift by using only a single frame store, while processing all enhancement layer texture outside the motion-compensated prediction loop.

The central idea of this thesis is that a scalable video system can be constructed from a hierarchy of single-layer bitstreams of increasing quality, the base layer being the single-layer bitstream with the lowest quality and the enhancement layers, rather than coding texture, specify the operations required to *transform* a stream that has been received into another of higher quality. Such an approach has a number of advantages: at each layer the signal is only quantized once thus avoiding the problems with repeated quantization that were observed for MPEG-2-compliant SNR scalability. In addition, the generation of the scalable representation from the set of single-layer bitstreams need not be done concurrently with coding of those streams. Similarly, the recovery
of single-layer streams from the scalable representation can be done upstream of the final decoder. The use of standard single-layer encoders and decoders as well as the decoupling of the single-layer and scalable forms of the bitstream hierarchy enable some attractive new architectures for scalable video.

Section 4.2 describes this stream morphing process in detail for the SNR scalable case (temporal and spatial scalability will be discussed in Chapter 6). This description is further refined in Section 4.3, which goes into the detail of how each of the components in a single-layer MPEG-4 bitstream (as described in Chapter 2) is processed by the stream morphing system. Section 4.4 describes the application scenarios where stream morphing can be used. Section 4.5 analyses the computational complexity of the new scheme. Sections 4.6, 4.7 and 4.8 show and comment upon experimental results for stream morphing in the constant quantizer case. Finally, Section 4.9 draws conclusions from this chapter. Appendix A shows the complete enhancement layer bitstream syntax for the experimental system used.

### 4.2 Overview of Stream Morphing

Figure 4.1 shows a schematic diagram of an $n$-layer stream morphing encoder. It consists of $n$ parallel single-layer encoders (we will use MPEG-4 for our experiments but this method could be similarly applied to other similar standards such as H.263) which are unmodified from their standard form except that only the base layer needs to be coded using the standard syntax (e.g. MPEG-4). For the enhancement layers, the stream morphing process uses its own syntax and can take input data in the form of raw quantized DCT coefficients plus the other macroblock parameters such as quantizer step size and prediction mode. In Chapter 6, we will see that temporal and spatial scalability generally requires different motion vectors to be used in consecutive layers so motion vectors must be morphed in a manner similar to the quantized DCT coefficients. While it is
also possible to morph motion vectors in SNR scalability, experience has shown that the overhead associated with motion vector morphing outweighs any savings that can be achieved and as such the same motion vectors are always used in all layers.

Once the quantized DCT coefficients have been determined in each layer, they are used as input to the stream morphing encoder, along with side information such as the sequence and VOP header parameters and the quantizer step size values for each macroblock. This encoder creates a bitstream describing the same fundamental entities that exist in the standard (single-layer) MPEG-4 description at each layer however the number of bits required to code this information is greatly reduced (when compared to regular single-layer) by using the contents of the previous layer as a form of "prediction". An alternate view of the stream morphing process is as a method for achieving efficient simulcast of multiple video streams.

The sequence and VOP header information are largely identical for all layers.
4.2 Overview of Stream Morphing

Figure 4.2: Morphing between macroblocks in successive layers

in the SNR scalable case. All VOP timing information is identical, all layers are of the same size and all layers should be coded using the same picture mode (I-VOP, P-VOP or B-VOP). The only significant parameter that varies between layers, and thus needs to be coded in the enhancement layer picture header, is the value of the quantizer step size to be used in the first macroblock.

Once this fixed information has been coded, the VOPs in all layers are processed one macroblock at a time. The content of each enhancement layer macroblock is coded with respect to the corresponding element in the coincident macroblock in the previous layer, as shown by the arrows in Figure 4.2. To prevent the need to store large amounts of data, it is desirable to process coincident macroblocks in all layers simultaneously. The following list summarizes the quantities that must be coded in each macroblock, as first described in Section 2.4:

- The quantizer step size for the macroblock which is expressed as a differential value relative to either the value used in the last macroblock, or in
the case of the first macroblock, the step size encoded in the VOP header.

• The prediction mode for the macroblock. For I-VOPs this parameter is not required since there are no alternate modes to choose from.

• The coded block pattern which is notionally one bit per block denoting whether there are any non-zero coefficients in that block. For intra-coded macroblocks, DC coefficients are assumed to be non-zero and are always coded and as such are not included in the calculation of the coded block pattern.

• Any motion vectors associated with the macroblock. In I-VOPs and intra-coded macroblocks in P-VOPs, no motion vectors are present. For SNR scalability, the motion vectors used in all co-incident inter-coded macroblocks are the same, as previously stated. The only occasion where motion vectors need to be explicitly coded in enhancement layers is where previous layers were all intra-coded and motion vectors appear for the first time above the base layer.

• The quantized coefficient values themselves.

Note that the above list is not in the same order or form as the MPEG-4 macroblock syntax. For efficiency reasons, when using variable-length codes MPEG-4 groups some of these symbols together, notably the macroblock mode and coded block pattern for the chrominance blocks are specified using one variable-length codeword. The luminance coded block pattern, which could be expressed as four different VLCs, is similarly grouped into a single VLC. Figure 4.2 can be considered to be an expanded form of the MPEG-4 macroblock syntax: it encodes the same information but has been completely broken down into its component parts. Only one 8x8 texture block is shown in Figure 4.2; the process is repeated for the other 8x8 blocks in the macroblock. An intra-coded macroblock is shown in the figure; inter-coded blocks are identical except they
do not have a DC coefficient nor do they have the intra AC coefficient prediction flag.

The decoder corresponding to Figure 4.1 is shown in Figure 4.3. The quantized DCT coefficients and other macroblock data are decoded from the base layer bitstream and then the first enhancement layer bitstream is used to morph this low quality service into another single-layer bitstream of higher quality and so on for all available enhancement layers. This results in the full set of single-layer bitstreams generated at the encoder except for those that correspond to enhancement layers that have been lost. Normally only the highest quality bitstream is then used. Figure 4.3 shows this being decoded with the remainder of a standard MPEG-4 single-layer decoder. While the other bitstreams are not used in this example, it is important to appreciate that these are available if required. Section 4.4 will describes some application scenarios where this is desirable.
4.2 Overview of Stream Morphing

4.2.1 Arithmetic Coding

Variable-length codes of the type discussed in Section 2.3.5 are suited to coding symbols drawn from alphabets of a reasonable size and whose symbols do not have skewed probability distributions, i.e. one symbol does not have a very much higher probability of occurrence than any other. If this is not the case then the fact that each symbol must occupy an integer number of bits can result in significant expansion of the number of bits required to code a typical message compared with the actual entropy of that message. For example, applying the Huffman procedure to an alphabet with three symbols A, B and C with probabilities of 90%, 9% and 1% respectively would yield a symbol for A that is one bit long and symbols for B and C that are each two bits long, giving an average number of bits per symbol of 1.1. The actual entropy $E$ of such a source is much lower:

\[
E = \sum_i p_i \times \log_2 \left( \frac{1}{p_i} \right) \\
= 0.9 \times \log_2 \left( \frac{1}{0.9} \right) + 0.09 \times \log_2 \left( \frac{1}{0.09} \right) + 0.01 \times \log_2 \left( \frac{1}{0.01} \right) \\
\approx 0.52 \text{ bits/symbol}
\]

In this case, the use of variable-length codes requires more than twice as many bits as the theoretical minimum.

Stream morphing requires the coding of many symbols (each enhancement layer has as many symbols as the corresponding single-layer service in the expanded form of Figure 4.2) that are drawn from small alphabets that typically have heavily biased distributions. For example, we would intuitively expect that given two single-layer bitstreams generated from the same input sequence that use very similar values for the quantizer step size then there is a very high probability that any coefficient present in the lower quality layer would also be present in the second higher-quality service. To morph the coefficients in these two layers, we encode a symbol for each lower layer coefficient which describes any coefficient magnitude change in the upper layer plus a description
of any new coefficients that were not present in the lower layer. For closely-spaced layers, we would expect that the symbols for each existing coefficient are heavily skewed towards the symbol for "no change" rather than the other possibilities such as the coefficient decreasing in magnitude or disappearing altogether. Using variable length codes in this case would be extremely inefficient and ultimately impractical for this application.

Arithmetic coding [51] is a technique with higher computational complexity than variable-length coding, however its efficiency can approach the Shannon limit without any restrictions on the alphabets used. Figure 4.4 illustrates the coding of two symbols with an arithmetic coder. Both symbols are drawn from an alphabet with four symbols whose probabilities are listed in Table 4.1. Both encoder and decoder maintain a range of values \([L, H]\), starting at 0.0 to 1.0 at the beginning of a sequence. As each symbol is coded the range of values is narrowed according to the cumulative probability of that symbol in the given probability model. In this example, the first symbol coded is \(b\) which contracts the range from \(L_0 = 0.0, H_0 = 1.0\) to \(L_1 = 0.125, H_1 = 0.525\). After the second symbol \(c\) the range contracts further to \(L_2 = 0.125 + 0.525 \times (0.525 - 0.125) = \)
Table 4.1: Symbol probabilities for arithmetic coding example

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Probability</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>0.125</td>
<td>[0,0.125)</td>
</tr>
<tr>
<td>b</td>
<td>0.4</td>
<td>[0.125,0.525)</td>
</tr>
<tr>
<td>c</td>
<td>0.3</td>
<td>[0.525,0.825)</td>
</tr>
<tr>
<td>d</td>
<td>0.175</td>
<td>[0.825,1.0)</td>
</tr>
</tbody>
</table>

0.335, $H_2 = 0.125 + 0.825 \times (0.525 - 0.125) = 0.455$ and so on. After all the symbols have been encoded, any value in the final range can be transmitted to the decoder which performs the same progressive contraction of the possible range of values as the encoder and with reference to the value sent to it by the encoder can determine the correct symbol that was encoded at each stage.

For practical implementations this approach using real numbers has a number of problems. The most serious of these is the need to store and perform arithmetic on values to hundreds or thousands of decimal places, even for relatively short messages. In practice, the endpoints of the range are stored as integers (typically 32 bits long which is the native format for many current computers). When using integers, the range must not be allowed to shrink so far that different symbols map to the same values in the range. When the range of values drops below half the initial range ($2^{31}$ for 32-bit integers) renormalization is performed at the encoder and decoder which expands the range by a factor of two. Renormalization events come in three categories:

1. If the current range is entirely within the bottom half of the initial range the encoder signals to the decoder that the value is within the bottom half of the range by emitting a single bit with a value of zero. The endpoint values of the range are multiplied by two and encoding or decoding can continue. Figure 4.5(a) shows how the endpoints of the range change when renormalization occurs.

2. Similarly, if the current range is in the top half then a bit with a value of one
is emitted and the endpoints are doubled and shifted down by the width of the initial range to return them to the valid range. This corresponds to Figure 4.5(a) with the ranges mirrored about the half-way point at 0.5.

3. If the range spans the half-way point of the initial range then no bits are emitted immediately but the encoder increments its count of delayed bits and the endpoints of the range are doubled and half the initial range is subtracted to bring them into the valid range (Figure 4.5(b)). The delayed bits are flushed when the range contracts into one either half of the initial range (items 1 and 2 above) at which time a one bit or zero bit, as previously described, are emitted into the bitstream followed by the delayed bits which have the opposite value. For example, if the range contracts into this central region twice so that two renormalization events of the type shown in Figure 4.5(b) are required and then upon coding of further symbols the range contracts into the bottom half then the three bits 011 are placed into the bitstream. Proof of the validity of this will not be shown here but can be found in [51].

When coding symbol $n$ the lower and upper ends of the range are calculated...
by:

\[ L_n = L_{n-1} + \left( (H_{n-1} - L_{n-1}) \times l \right) \text{div} t \]  
\[ H_n = L_{n-1} + \left( (H_{n-1} - L_{n-1}) \times h \right) \text{div} t \]

(4.1)

(4.2)

where \( l/t \) is the low end of the cumulative frequency range for the symbol being coded and \( h/t \) is the upper end (\( l \) and \( h \) are the numbers in the third column of Table 4.1 and \( t = 1 \)). In *adaptive arithmetic coding*, statistics are gathered during the coding process in which case \( t \) is the total number of symbols so far counted and \( l \) and \( h \) are cumulative symbol counts. So that the product in each expression here can fit into a finite precision \( w \)-bit value, the number of bits \( f \) used for frequency counts must be restricted such that \( w \geq 2^f + 1 \) [52]. For \( w = 32 \) this restricts frequency counts to less than \( 2^{15} \). This can be a significant restriction for large alphabets, which are typically required for coding of text where one symbol is used for each distinct word. [52] describes a modified method which raises the frequency count limit to \( 2^{30} \), however, we will not need this for stream morphing as our largest alphabet has only 64 possible values. For this work we will use a slightly modified form of arithmetic coding known as *range coding* [53] which has a different renormalization procedure which is done at the byte level rather than for each bit which reduces computational requirements. To reduce computational complexity further we will not use adaptive arithmetic/range coding; fixed models will be used at all times and total frequency counts are normalized to \( 2^{15} - 1 \). This removes the need for integer division to be performed when coding symbols in the encoder.

**4.2.1.1 Resynchronization**

For compatibility with existing MPEG bitstreams, it is desirable to insert start codes as discussed previously in Section 2.4.2. If errors occur and the decoder loses track of its position in the bitstream, it can then skip to the start of the next video packet or VOP by looking for a sequence of 23 zero bits followed by
a one bit which can only occur at the start of an MPEG header. To prevent start code emulation when using arithmetic coding the encoder and decoder must keep a count of the number of consecutive zeros seen and insert a stuffing bit with value 1 if 22 consecutive zeros are counted. In the standard MPEG bitstream formats this counting is unnecessary since the location of stuffing bits are specified in the standard to cover all possible situations where start code emulation could occur. This type of analysis is not possible when using arithmetic coding since symbols do not use integer numbers of bits.

4.3 Morphing of Bitstream Components

The following sections describe how the morphing process works on each of the bitstream components shown in the expanded format of Figure 4.2. A full description of the stream morphing bitstream syntax and a list of the probability models used for arithmetic coding is given in Appendix A.

Most symbol types to be described here are associated with more than one probability model, the actual model that is used at any given time is determined by some additional context information that both the encoder and decoder can compute. The context information is designed to give some indication as to the likely probability distribution for the symbol to be coded, the use of separate models that are a better match for the statistics of the source results in a lower number of bits generated. The downside to the use of many models is one of complexity: not only is extra processing required to compute the context, and thus which model is to be used, but also in extra storage for the probability tables and logic to index alternate models. This is especially critical for hardware implementation, which is outside the scope of this thesis. As such, the number of models to use and the model selection rules in these cases have been determined by empirical means only. Primarily, this involved observation of the number of bits generated by symbols encoded using each probability model.
and comparing this to the entropy of those symbols. Any significant mismatch (≥ 15 – 20%) that was observed during testing on a variety of video sequences indicated that the number of models and/or the model selection logic for that symbol needed to be re-appraised.

### 4.3.1 Quantization Step Size

The MPEG-4 standard allows for the macroblock quantizer to be changed by at most ±2 relative to value used in the previous macroblock, or in the case of the first macroblock in the VOP, the initial quantizer value coded in the VOP header. There is one exception to this rule: the quantizer step size cannot be changed in inter-coded macroblocks with four motion vectors (Section 2.3.4). To allow for recovery of single-layer bitstreams in MPEG-4-compatible form we must code enhancement layer quantizer step size changes to conform with this ±2 difference limit.

While it is possible to vary the quantizer step size independently in each layer the overhead associated with doing this for perceptual rate control algorithms over many layers is large. As far as possible, the step size value should vary in
4.3 Morphing of Bitstream Components

Quantizer step size clipped at 31 in these regions for layer $n$ and possibly $n+1$.

Figure 4.7: Clipping of quantizer step size at low rates in MPEG-4

the same way as the base layer so that the differential value there can be used to predict the differential value in each enhancement layer. The absolute (rather than differential) values for the quantizer step sizes in each layer are therefore separated by an approximately constant offset as shown in Figure 4.6. Rate control algorithms being run in each layer need to depart from this convention occasionally in order to meet their respective rate targets so other values for the differential quantizer step size are supported. However, the probability models used for arithmetic coding are heavily biased towards the differential value in the current layer being identical to that in the previous layer.

The MPEG-4 standard [54] also limits the maximum value for the quantizer step size to be 31, which is a significant constraint for perceptual rate control algorithms operating at low bit rates. In such cases there will be areas within a frame where the rate control algorithm would like to vary the quantizer step size in a continuous manner but must use a constant value of 31 for the quantizer step size because larger values are not permitted. As enhancement layers are added the amount of clipping is reduced as the step size moves away from the threshold, as shown in Figure 4.7. As such there will be regions where the quantizer step size is not predicted well from the previous layer where clipping has occurred and the differential quantizer step size is always zero. To account
4.3 Morphing of Bitstream Components

for this effect, the stream morphing algorithm uses a different arithmetic coding probability model to describe the differential quantizer step size where the value of the previous layer quantizer is near this limit (≥ 30). In this way, variation can be introduced in the enhancement layers without having to change the statistics used for the standard model. A flag is defined in the sequence header to turn off the use of the different model which is useful when using a constant quantizer step size where we do not expect the quantizer value to change when it is near this threshold.

4.3.2 Macroblock Mode

MPEG-4 P-VOPs allow for individual macroblocks to be coded in intra mode, which can be used for areas where the prediction is poor and coding the motion-compensated prediction difference would most likely require nearly as many (or possibly more) bits than coding in intra mode. In the lower layers, where the quality of the video (and hence the inter-frame prediction) is poorer, the use of intra macroblocks is often more widespread than in higher layers where the prediction is of better quality. Stream morphing therefore allows for the use of different macroblock modes in different layers for P-VOPs. One symbol is coded in each enhancement layer P-VOP macroblock to signal whether the mode is different from that used in the coincident macroblock in the previous layer. Note that the choice of macroblock mode is non-normative and so some implementations may choose to use the same mode in all layers.

Where a mode change has taken place and the coincident macroblock in the previous layer was coded in intra mode and the current layer in inter mode (or vice versa), the rest of the information in the macroblock cannot be morphed from the contents of the previous layer macroblock. Intra-coded blocks tend to have many more coefficients than inter blocks and as such predicting the locations and magnitudes of the coefficients in one form is not possible given the previous layer is coded in the other form. Similarly, if the current
4.3 Morphing of Bitstream Components

layer is to be coded in inter mode then motion vectors will need to be specified since they are not present in the previous layer. Once such a macroblock mode change has been signalled, the contents of the macroblock are passed through the arithmetic coder in their original syntax (i.e. a sequence of VLCs). Stopping the arithmetic coder to transmit a short sequence of VLCs belonging to a single macroblock is an expensive operation so each VLC bit is coded using the arithmetic coder and a uniform binary model (two possible values corresponding to bit values of 0 or 1 with 50% probability for each).

4.3.3 Skipped Macroblocks

Each macroblock in MPEG-4 P-VOPs and B-VOPs contains a single bit marker denoting whether the macroblock is "skipped". A skipped macroblock requires no other information to be transmitted: it has (implicitly) no non-zero DCT coefficients, the same quantizer step size value as the previous macroblock and a zero (full, not necessarily zero differential) motion vector. At low to moderate bit rates, a large proportion of macroblocks in a sequence will be skipped and the fact that such macroblocks can be coded with a single bit rather than three or more separate VLCs is a significant gain in efficiency.

Stream morphing also defines the notion of a skipped macroblock for its enhancement layers, albeit in a slightly different form. Firstly, unlike the standard MPEG-4 case where the macroblock skipped VLC is the first VLC in the macroblock, for stream morphing any quantizer changes have already been signalled before the macroblock skipped symbol appears. This is done to support the use of different probability models depending on the quantizer difference between the current layer and the previous layer, a point which is to be discussed further in Section 4.3.7. Similarly, since any motion vectors used in enhancement layers have already been coded in a previous layer and are not allowed to change between layers, the definition of macroblock skipped does not include any reference to the magnitudes of motion vectors in the macroblock. The mac-
roblock skipped symbol in stream morphing is only used to indicate whether a macroblock has no non-zero DCT coefficients; all other macroblock parameters have been coded explicitly (at near zero cost since they are almost always correctly predicted from the previous layer) or have been previously coded and cannot change.

The macroblock skipped symbol is used only for macroblocks where the coincident macroblock in the base layer had no non-zero DCT coefficients. In other cases, the macroblock is always scanned for changes. Since coded macroblocks normally appear in contiguous groups rather than scattered randomly around the frame, a count is made of the number of macroblocks surrounding the current macroblock that are coded in the same or previous layer and a different model is used depending on the value of this count. To simplify processing and remove the need for additional storage only the causal macroblocks that have been previously decoded are counted (Figure 4.8).

### 4.3.4 Intra AC Coefficient Prediction

Intra-coded macroblocks in MPEG-4 contain a single bit flag to indicate whether coefficient prediction, as previously described in Section 2.3.2, is enabled or disabled. Stream morphing allows for this parameter to change between layers and
4.3 Morphing of Bitstream Components

A single symbol is generated in each intra-coded enhancement layer macroblock to signal its value.

### 4.3.5 Signalling of Coded Blocks

For intra-coded macroblocks and inter macroblocks that are not skipped, the MPEG-4 syntax codes which of the six 8x8 blocks (for 4:2:0 format) in the macroblock have any non-zero coefficients. VLC coding of this information using six discrete bits is not efficient so the coded block pattern is split into two VLCs, one for luminance (CBPY) and another for chrominance (CBPC), the latter is also combined with the specification of the macroblock prediction mode. Since stream morphing uses arithmetic coding, it is not necessary to combine symbols together in this way but rather they can be coded independently. Indeed, the probability model used for each block depends on the number of surrounding coded blocks in a way similar to the previously described macroblock skipped symbol (Section 4.3.3 and Figure 4.8) so each symbol must be coded separately. As for the macroblock skipped case, only those blocks that are skipped in the base layer use coded block symbols; those with non-zero coefficients in the previous layer are always scanned for changes.

### 4.3.6 Morphing of DCT Coefficients

The process of scanning an 8x8 block occurs in two stages: the first stage iterates over all the non-zero coefficients in the coincident block in the previous layer and describes any changes to those coefficients for the current layer. How this is done depends on whether the macroblock is coded in intra or inter mode and each case will be discussed individually in the next two sections.

The second stage of block scanning involves identifying any new coefficients that are non-zero in the current layer but which were zero in the previous layer. This is done in a similar way to standard MPEG-4: the scan starts at the first
4.3 Morphing of Bitstream Components

coefficient and counts the number of zero coefficients before a new coefficient appears. The magnitude of the coefficient (almost always ±1) is then coded. This is repeated until all new coefficients have been coded. For efficiency, MPEG-4 uses single VLCs for (run, level, end of scan) tuples. However, to reduce the size of the probability models involved without requiring complex escape codes, stream morphing splits these components and uses separate symbols for each. Rather than coding the end of the scan at each coefficient the number of new coefficients for the block is coded at the start of the scan.

Splitting the scan into two sections in this way increases efficiency for blocks where no new coefficients are added but some coefficients may have updated values as this requires no scan be made through the block. A scan need only be performed in that subset of blocks where new non-zero coefficients that did not exist in the previous layer have appeared.

In Figure 4.2 the layer \( n \) coefficients \( AC_{1,n} \) through \( AC_{c,n} \) are scanned first and in this example all coefficients except \( AC_{4,n} \) are non-zero in the enhancement layer and any change in value is signalled. A symbol is generated for \( AC_{4,n} \) that indicates the coefficient has been deleted, i.e. it had a non-zero value in the base layer but is zero in the current layer. Once these existing coefficients have been scanned the existence of three new coefficients is signalled and finally those new coefficients \( AC_{2,n+1} \), \( AC_{c'-2,n+1} \) and \( AC_{c',n+1} \) are located and their magnitudes coded.

4.3.6.1 Coding of Existing Intra Coefficients

Since there is no prediction in the case of intra-coded macroblocks and the input sequence is assumed to be the same in all layers, we know that coincident unquantized DCT coefficient values in different layers are identical. With different quantization step sizes being used in each layer, the quantization bins may overlap, which allows for the possible range of values that the coefficient may have to be narrowed to such an extent that in some layers it is not necessary to
Figure 4.9: Coding of intra coefficients

code a symbol to determine the value the coefficient has with respect to a finer quantization step size.

Figure 4.9 shows an example of this process. We wish to code the coefficient whose unquantized value is marked in the figure. For the first $n$ layers this value is zero after quantization and for layers $n + 1$ and $n + 2$ the quantized value is 1. The grey regions in the figure denote the range of possible unquantized values that correspond to the quantized value in each layer. The quantization bins for layers $n$ and $n + 1$ overlap and since the true value of the coefficient must be the same, the decoder can therefore infer that after layer $n + 1$ the value really lies within the smaller range marked to the right of the layer. In other words, the range of possible values that a coefficient can take in layer $n + 1$ is the intersection of the quantization bins for that layer and all previous layers. Comparing this range to the quantization bins for layer $n + 2$ (which we can compute if the quantization step size for this layer has been already decoded) we see that in this example the coefficient value lies entirely within the range for a quantized value of 1 therefore it is not necessary to code *anything* in this layer for the given choice of quantizer step size. Layers 1 to $n - 1$ are not shown here since their range of possible values for a quantized value of zero
are supersets of the range for layer \( n \) and do not narrow down the range after their intersection is calculated. Note that this quantization bin overlap is useful only for systems that have at least three layers.

Most intra coefficients, especially those in lower layers, are not able to be skipped due to the effect just described. These coefficients each span \( m \geq 2 \) quantization regions (that each correspond to different quantized coefficient values), and a symbol must be coded to identify which of these regions contains the coefficient. This can be done with a single arithmetic coded symbol with \( m \) possible values. The statistics of these symbols depend on the location of the transitions between quantization regions. In each case an integer quantity \( l \in [0, 31] \) is computed which describes the location of the first transition (closest to zero), and is used to choose which model is used:

\[
l = \left(31 \times (t - r_{\text{bottom}})\right) / (r_{\text{top}} - r_{\text{bottom}})
\]  

(4.3)

where \( t \) is the largest unquantized value (in absolute terms) in the valid quantization bin that is closest to zero, \( r_{\text{bottom}} \) is the bottom of the quantization range left from the previous layer (after intersection with the regions from all layers) and \( r_{\text{top}} \) is the top of that region. The division operation in equation 4.3 is again an integer division that always rounds towards zero. Figure 4.10 illustrates this for three different transition locations \( t_1, t_2, t_3 \) for two quantization regions. It can be clearly seen that if the probability density function for coefficient magnitudes is that shown on the left-hand side of the figure, then we will want to skew the models for arithmetic coding to favour the upper bin (value '1' as marked in the figure) for the first two examples (low values of \( l \)) and towards '0' for the third example which has a high value for \( l \). Our stream morphing implementation uses six different models for the \( m = 2 \) case with model selection being done according to the value of \( l \) as shown in Table 4.2. A similar process is used for the case of three quantization regions (the right-most example in Figure 4.10) although the first transition will tend to be near the bottom of the region given that a full bin lies above it. Model selection for the \( m = 3 \) case is
4.3 Morphing of Bitstream Components

Two Quantization Regions

\[T_{top}\]

Valid Quantization Region

\[T_{bottom}\]

Three Quantization Regions

Figure 4.10: Effect on region transition location on symbol statistics

<table>
<thead>
<tr>
<th>( l \in )</th>
<th>Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0,3]</td>
<td>1</td>
</tr>
<tr>
<td>[4,7]</td>
<td>2</td>
</tr>
<tr>
<td>[8,15]</td>
<td>3</td>
</tr>
<tr>
<td>[16,21]</td>
<td>4</td>
</tr>
<tr>
<td>[22,27]</td>
<td>5</td>
</tr>
<tr>
<td>[28,31]</td>
<td>6</td>
</tr>
</tbody>
</table>

Table 4.2: Probability model selection for intra coefficients (two quantization regions)

done by the relation shown in Table 4.3. For the rare \( m = 4 \) case (which only can occur if there is a very large difference in quantizer step size values used in adjacent layers) two models are used, the first for \( 0 \leq l < 3 \) and the other for the remaining values of \( l \). Tables 4.2 and 4.3 were devised using empirical means as stated at the beginning of Section 4.3.

Note that in order to perform these calculations any DC or AC coefficient prediction used for coding (Section 2.3.2) must be removed and the full values of each coefficient used. This also means that if the use of AC prediction is changed between layers (Section 4.3.4) this has no effect on the morphing process for the individual coefficients.

The statistics for coding DC and AC intra coefficients are different and use
different models. AC coefficients are typically spread over a much narrower range of values than the DC coefficients and as such the probability of a value being non-zero away from the quantization boundary is smaller.

### 4.3.6.2 Coding of Existing Inter Coefficients

For macroblocks with motion-compensated prediction, there is no guarantee that the quantization regions for coincident coefficients overlap and thus the technique described above for intra coefficients cannot be used. For each non-zero coefficient in the previous layer in an inter-coded macroblock, one symbol needs to be generated that describes how the quantized value has changed for the current layer. This symbol takes one of five possible values whose semantics are:

- The quantized value is the same as the previous layer.
- The absolute value of the quantized coefficient has increased by one compared to its value in the previous layer.
- The absolute value of the quantized coefficient has decreased by one compared to its value in the previous layer. For coefficients with values of ±1 this means that the value in this layer is zero. This condition must be explicitly coded if the coefficient had a non-zero value in the previous layer, unlike other coefficients that are also zero in the current layer.

\[
\begin{array}{|c|c|}
\hline
l \in & \text{Model} \\
\hline
[0,1] & 1 \\
[2,3] & 2 \\
[4,7] & 3 \\
[8,31] & 4 \\
\hline
\end{array}
\]

Table 4.3: Probability model selection for intra coefficients (three quantization regions)
• The absolute value of the quantized coefficient has increased by more than one compared to its value in the previous layer.

• The absolute value of the quantized coefficient has decreased by more than one compared to its value in the previous layer. This may also result in a zero-valued coefficient in this layer.

The last two possibilities listed here are comparatively rare and do not themselves specify exactly the value of the quantized coefficient in the current layer. In these cases, an escape sequence is generated in the form of one or more extra binary symbols, each symbol received with a value '0' denotes an additional increase in coefficient magnitude of 1 and a symbol with value '1' denotes the end of the escape sequence. For example, if a coefficient has value 1 in layer $n$ and 5 in layer $n + 1$ then a symbol corresponding to the fifth possibility in the above list (i.e. the coefficient magnitude has increased by at least two) is generated followed by three escape sequence symbols with values '0', '0' and '1' to specify the magnitude increase to be four rather than two. Note that the last item in the list is used for cases where the sign of the coefficient changes i.e. the magnitude of the change in the coefficient's value is larger than the coefficient's magnitude in the previous layer.

With the relatively coarse quantization used at low bit rates the majority of non-zero inter coefficients have values of $\pm 1$. Coefficients with larger absolute values were found to be more volatile and to have a significantly lower probability of retaining the same value between adjacent layers. To exploit this difference, different probability models are used, one for coefficients with values of $\pm 1$ and another for coefficients with larger absolute values.

4.3.6.3 Inter Coefficient Restoration

Coefficients in inter-coded macroblocks whose values go to zero in one layer after being non-zero in a lower layer often reappears again at a higher layer
where the finer quantizers used induces a non-zero coefficient even if improved prediction has (at least temporarily) removed the need for a coefficient at that location for some intermediate layer(s). These coefficients are included in the scan over existing coefficients, and a symbol is coded for each to indicate if the coefficient has been restored (i.e. having a non-zero value in the current layer). These symbols take one of four possible values:

- The coefficient is not restored, i.e. its value is still zero in the current layer.
- The coefficient has been restored and has the same value as the layer in which it last had a non-zero value.
- The coefficient is non-zero in this layer and has an absolute value that is larger than its last non-zero value in a previous layer.
- The coefficient is non-zero in this layer and has an absolute value that is smaller than its last non-zero value in a previous layer.

The precise offset for the last two cases here is coded using the same escape sequence that was defined in the previous section for coding the refinement of coefficients whose values had changed by more than ±1 with respect to the previous layer.

4.3.6.4 Preventing Frequent Coefficient Value Changes

Stream morphing avoids problems associated with quantization of the same signal multiple times by quantizing the full difference signal in each layer once, and then working on the quantized coefficient values to eliminate redundancy between the layers. A consequence of this for inter-coded texture is that the quantized values in each layer can change with very small differences in prediction between two adjacent layers. For example, in one layer the unquantized DCT coefficient might be slightly above the threshold for creating a non-zero coefficient however in the next layer up the prediction may be slightly better
and this value might drop to just below the quantization threshold and so the deletion of this coefficient would need to be signalled. In a higher layer with a finer quantization step size this value may again rise above the (now lower) threshold and the coefficient would be restored. This can occur for very small changes in the prediction difference yet this induces a number of extra symbols to be encoded. This effect does not occur for MPEG-2-compliant SNR scalability since for a second coefficient to be generated in the upper layer requires a significant extra difference to exist in that layer (enough to generate another non-zero coefficient with the finer quantizer).

A simple but effective strategy for dealing with this effect is for the encoder to detect when coincident quantized coefficients in some layers $i$ and $i + 2$ are identical but with a different value in the intermediate layer $i + 1$. In such cases the encoder forces the quantized coefficient in layer $i + 1$ to be identical to the surrounding layers. This avoids a pair of symbols being generated to describe the opposing changes in layers $i + 1$ and $i + 2$. Use of this technique has been found to decrease the overall bit rate for typical three layer sequences by 5%-10%. Experimental results to be presented later in this chapter indicate no significant deterioration in the quality of the intermediate layers when this technique is applied. When used with rate control (Chapter 5), significant performance gains have been observed as the bits required to code these unnecessary changes can be used to code new coefficients.

### 4.3.7 Layer Separation

The statistics of many of the symbol types mentioned above are very dependent on the "spacing" of the current layer from the previous layer, i.e. the difference between the quantizer step size values between the two layers. As was mentioned first in Section 4.2.1, we would expect that quantized coefficients in two layers that have quantizer step size values that are very close to each other to almost always have the same value. Conversely if the current layer has a
quantizer step size that is much smaller than the previous layer (i.e. finer quantization) then we would expect that many coefficients will have larger absolute values.

To compensate for this important effect, most symbols coded use probability models that are selected based on one of two measures of adjacent layer spacing. For symbols that refer to only one coefficient, which are the update (Section 4.3.6.2) and restoration (Section 4.3.6.3) of inter coefficients, a quantity $D_{qp}$ that summarizes the difference between the current layer quantizer step size and the step size when the coefficient's value last changed is calculated from the following expression:

$$D_{qp} = 8 \times q_{\text{lowprev}} \text{ div}(q_{\text{lowprev}} - q_{\text{current}})$$

(4.4)

where $q_{\text{lowprev}}$ is the value of the quantizer step size in the lowest layer where the coefficient had the same quantized value as in the (immediate) previous layer (this must be determined on a coefficient-by-coefficient basis) and $q_{\text{current}}$ is the quantizer step size in the current layer. The model to use is then determined according to the relation shown in Table 4.4. $q_{\text{lowprev}}$ is used instead of the quantizer step size in the previous layer as we would expect a coefficient is more likely to change if it has had the same value for a number of layers that span a large range of quantizer step size values. Note that since intra coefficients are coded only with respect to the location of the (overlapping) quantization boundaries (Section 4.3.6.1), reference to the quantizer step size is not required.

Table 4.4: Probability model selection for coefficient-oriented symbols

<table>
<thead>
<tr>
<th>$D_{qp}$ ∈</th>
<th>Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>(128,∞)</td>
<td>1</td>
</tr>
<tr>
<td>[73, 128]</td>
<td>2</td>
</tr>
<tr>
<td>[26, 72]</td>
<td>3</td>
</tr>
<tr>
<td>[16, 25]</td>
<td>4</td>
</tr>
<tr>
<td>(0, 15]</td>
<td>5</td>
</tr>
</tbody>
</table>

4.3 Morphing of Bitstream Components

139
For other symbols that have meaning at the block or macroblock level rather than the coefficient level, a different layer spacing metric is used which varies quadratically and is approximately proportional to the bit rate difference between adjacent layers:

\[ D_{rate} = q_{prev}^2 \text{div}(q_{prev} - q_{current}) \]  

(4.5)

where \( q_{prev} \) is the quantizer step size in the previous layer. The model to use is then determined by the relation shown in Table 4.5. This metric is used to select models when coding macroblock skipped/not skipped (Section 4.3.3), block coded/not coded (Section 4.3.5), the number of new coefficients in a block (Section 4.3.6), any change in intra AC prediction (Section 4.3.4) and for macroblock mode change (Section 4.3.2). These values for these symbols are dependent upon the number of non-zero coefficients present in the block or macroblock which is not a linear function of the quantizer step size so using a metric such as \( D_{qp} \) at the block or macroblock level (rather than the coefficient level) is not appropriate.

\( D_{qp} \) and \( D_{rate} \), like the intra coefficient processing models presented earlier, have been developed by empirical means only. As a large number of other models depends upon it, \( D_{rate} \) should have as a small number of entries otherwise the overall set of arithmetic coding probability models will grow too large.

Appendix A describes all symbol types, the number of different models that are used for each and how these are selected.
4.4 Applications

The previously-described $n$-layer stream morphing encoder (Figure 4.1) and decoder (Figure 4.3) can be used as direct replacements for the MPEG-2-compliant SNR scalable encoder (e.g. Figure 3.4) and decoder (Figure 3.1). $n$ bitstreams are generated and transmitted across some channel (with priority) to the decoder where the streams that are received intact are decoded, combined and displayed.

As stream morphing operates on individual single-layer bitstreams that are largely independent of each other (other than using the same motion vectors), unlike the layers of an MPEG-2-compliant SNR scalable system, the formation of the scalable representation of the bitstream hierarchy and its recovery can be decoupled from the original encoding and ultimate decoding of those bitstreams.

Figure 4.11 shows an alternative arrangement on the encoder side of a stream morphing system. The first stage of the encoding process generates $m$ single-layer bitstreams which are then stored, e.g. on disk. At some later time a subset $n \leq m$ of those streams is selected and passed through a stream mor-
Figure 4.12: Upstream recovery for single-layer switching

phing post-processor which parses the bitstreams and recovers the coefficient values and other data such as motion vectors. The post-processor generates $n - 1$ enhancement layers which are then transmitted over the channel with the base layer which is unmodified by the morphing process. Since stream morphing works without requiring any DCT/IDCT or motion-compensated prediction operations this post-processing is computationally-inexpensive to perform.

Figure 4.12 shows a similar decoupling that can be achieved for recovery of the single-layer bitstreams and the decoding of one of those streams. The $n$ layers that were generated by a stream morphing encoder, either in the on-line configuration (Figure 4.1) or by an off-line post-processor (Figure 4.11) can be converted back into the set of original single-layer bitstreams by a proxy that is located upstream of the final destination. Normally the highest quality single-layer stream that is received intact will be forwarded to the receiver, however since we are able to recover all bitstreams up to that point it may be desirable to use more than one of these, for example if there are multiple clients who have different bandwidth or computational constraints. Indeed with feedback from the decoder it is possible to do conventional single-layer switching [55] at this point rather than at the encoder. The downstream clients need not be aware that scalable coding has been used; they receive a standard format single-layer bitstream at the normal bit rate. Another advantage of this approach is that the client's networking software does not need to be modified to support
quality of service guarantees that is normally required for enhancement layer prioritization in scalable coding. So long as the path from the original server to the proxy is over one or more networks that do provide appropriate quality of service guarantees and this is where most of the network congestion is experienced then the effect will be very similar to complete end-to-end scalable coding.

The upstream recovery process of Figure 4.12 is shown in a "real time" configuration where the single-layer streams are immediately forwarded on to their destination(s). Another possibility is for the proxy machine to store the bit-streams locally for onward distribution at a later time, either in single-layer form or perhaps in another scalable form. Note that with local storage the proxy can also act as a post-processor (Figure 4.11). Stream morphing is therefore a very efficient method for distributing video inside of Content Delivery Networks [55] where numerous "edge" servers are placed throughout the Internet to serve nearby clients rather than all clients trying to access the same single central server which can easily become overloaded (in terms of network traffic and/or computational load). For switched single-layer systems, the edge servers must store a number of different versions of the same video sequence that have been coded at different bit rates. Rather than "simulcasting" these streams from the central server to the edge servers (albeit not in real time), a scalable representation, which consumes considerably less data than all of the single-layer streams, is formed that can then be transmitted and all the single-layer streams are recovered at the edge servers for storage. Note that since there is often no need for data to be transmitted in real time, a reliable transport (such as TCP/IP) can sometimes be used and layer prioritization is not an issue.

The two different application scenarios described above require two different types of rate control for use in CBR environments. For decoders that recover stream morphing enhancement layers and decode simultaneously (Figure 4.3),
a rate control algorithm would seek to generate a constant number of bits per second for each individual layer (base layer plus stream morphing enhancement layers). Chapter 5 describes a rate control algorithm that deals with this case. For upstream recovery (Figure 4.12) all of original single-layer bitstreams should be coded CBR which will result in stream morphing enhancement layers that will be VBR. Existing single-layer rate control techniques can be used in this case. It is not possible to have CBR stream morphing enhancement layers that are recovered to form CBR single-layer bitstreams.

4.5 Computational Complexity

Stream morphing is based upon the use of standard MPEG-4 encoders and decoders, the computational complexity of which has been analysed elsewhere [15]. Excluding the operations performed by the stream morphing post-processor, which will be discussed in this section, the computational requirements for an $n$-layer stream morphing encoder are $n$ times those for a single-layer MPEG-4 encoder except that motion estimation needs to be performed in one layer only given that the same set of motion vectors must be used in all layers. As motion estimation makes up a significant proportion of the total computational cost of an MPEG-4 encoder, the actual increase in complexity is therefore far less than a full factor of $n$. Similarly, at the decoder side the cost is the same as a single-layer MPEG-4 decoder plus that incurred by the stream morphing pre-processor in forming the input to that decoder.

4.5.1 Methodology

We consider the stream morphing encoder (Figure 4.1) and decoder (Figure 4.3) where the enhancement layer bitstreams are generated and recovered at the same time as the constituent single-layer bitstreams are encoded or decoded. The off-line encoder (Figure 4.11) and the proxy decoder (Figure 4.12) have
slightly higher computational requirements due to the generation and recovery of intermediate MPEG-4 bitstreams, the overhead for which can be estimated using existing techniques [15].

The stream morphing encoder operates a macroblock at a time processing each layer in turn and generating symbols for each coefficient that was present in the previous layer and identifying and coding new symbols in the current layer. Since the stream morphing operation is co-located with the original encoder, we assume that a buffer containing the quantized DCT coefficients for each layer is already available. Similarly, at the decoder the stream morphing pre-processor generates coefficients that can be immediately placed into a buffer where the single-layer MPEG-4 decoder can then perform the remaining inverse quantization, IDCT etc operations to form the decoded output. We seek only to count significant integer operations used in the morphing process: multiplies, divides, additions, subtractions, shifts etc. Due to the small amounts of data involved and the fact that this data is normally cached in the single-layer encoders and decoders, the memory bandwidth requirements for the algorithm are minimal and will not be considered further.

The computational cost will be divided into two components: the cost associated with the generation or recovery of arithmetic coded symbols and secondly the cost of calculating which probability model is required for the decoding of those symbols. While the cost of processing a symbol is dependent on whether it is being encoded or decoded, the model calculation overhead is identical between the encoder and the decoder since the same operations are involved in both.

Sections 4.5.2-4.5.5 detail the operations required in each of these categories and a tally of each is presented in Table 4.6. Later in this chapter, counts of each operation type will be presented that are gathered from profiling data generated from encoder and decoder runs which will allow estimation of the absolute number of multiply, add etc. operations on typical video sequences.
4.5 Computational Complexity

4.5.2 Coding and Decoding of Symbols

With non-adaptive normalized probability models (Section 4.2.1) integer division is not required to encode a symbol. Two integer multiplies, two right shifts (by a fixed number of bits which is known in advance and depends on the normalization) and a single integer addition are required for each symbol encoded.

At the decoder two shifts (again by a fixed number of bits) and a single integer divide are required to determine the cumulative frequency of the symbol being decoded. To update the decoder state a further two integer multiplications and an integer subtraction are required.

Furthermore the arithmetic encoder and decoder requires periodic renormalization. This operation requires three shifts and an addition in the worst case. Encoding and decoding a given sequence will require the same number of renormalization operations at both the encoder and decoder.

4.5.3 Per Macroblock Operations

After the decoding of the differential quantizer step size and calculation of its full value (requiring one addition) the quantity $D_{rate}$ (equation 4.5) needs to be computed into order to determine the correct models to use for other symbols in the macroblock. Determining $D_{rate}$ requires an integer multiplication, one subtraction and one division.

4.5.4 Intra Coefficients

Each non-zero intra-coded DCT coefficient in the previous layer is associated with a range of possible unquantized values for that coefficient for which it is necessary to calculate the number of quantization bins in the current layer this range spans (Section 4.3.6.1). Quantization must therefore be performed on both ends of this range of values. The implementation of the H.263 quantiza-
tion operation in the Microsoft software for the MPEG-4 Verification Model [31] (since there is no normative definition for the quantization we must choose a specific method of implementation) requires two integer division operations, one multiply, two shifts and an add for a DC coefficient and one division and one shift for an AC coefficient. For coefficients that span more than one region, the location of the first transition point must be computed in order to evaluate \( l \) (equation 4.3). Determining the endpoint of a quantization region takes, in the worst case, one multiply, one add, one shift and one subtraction for a DC coefficient and two additions, two subtractions one multiply and one shift for an AC coefficient. Evaluation of equation 4.3 itself requires one integer multiply, one divide and two subtractions.

### 4.5.5 Inter Coefficients

New inter-coded coefficients that were not present in any previous layer require only symbols that have single models and do not require any extra computation. For coefficients that are non-zero in the previous layer or are candidates for restoration (are zero in the immediately previous layer but were non-zero in a lower layer) the quantity \( D_{qp} \) needs to be computed (equation 4.4) which requires one integer division and a subtraction.

### 4.5.6 Memory Requirements

Apart from buffers containing the quantized DCT coefficients in each layer (which will most likely already exist in the single-layer MPEG-4 encoders or decoders where they are generated or are to be used) and any output bitstream buffering, the only other significant piece of state information required in either the encoder or the decoder is the storage of the current unquantized intra coefficient range (Section 4.3.6.1). This amounts to two DCT coefficients worth of storage (normally 12 bits per value) for each coefficient in the macroblock.
4.5 Computational Complexity

<table>
<thead>
<tr>
<th>Operation</th>
<th>Multiply</th>
<th>Divide</th>
<th>Add</th>
<th>Subtract</th>
<th>Shift</th>
</tr>
</thead>
<tbody>
<tr>
<td>Symbol Encode</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>Symbol Decode</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Renormalization</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>QP and $D_{rate}$ calculation</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$D_{qp}$ for existing/restored inter</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Existing intra DC (one region)</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>Existing intra DC (&gt;1 region)</td>
<td>4</td>
<td>5</td>
<td>3</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>Existing intra AC (one region)</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>Existing intra AC (&gt;1 region)</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>4</td>
<td>3</td>
</tr>
</tbody>
</table>

Table 4.6: Operation summary for stream morphing

which is 1152 bytes for a 16x16 pixel macroblock in 4:2:0 format. This storage can be re-used in each layer of a multi-layer encoder or decoder. If processing is done on a block-by-block basis this can be reduced to one sixth of this value (192 bytes). In-memory compression of these values should be possible given that many quantization ranges are identical, corresponding to the central (zero) quantization bin.

A small amount of extra memory is also required to keep track of the number of surrounding coded blocks and macroblocks for the purpose of model selection when signalling macroblock skipped and the coded block pattern (Sections 4.3.3 and 4.3.5 respectively). A record of coded blocks and macroblocks needs to be kept back as far as the macroblock marked as '2' in Figure 4.8 which is $N + 1$ macroblocks away from the current macroblock (where $N$ is the width of the frame in pixels divided by 16, or the number of macroblocks per row of macroblocks) along the row-wise scan used in MPEG-4. For each of these macroblocks we need to store one bit to specify if the macroblock was coded and another four bits to signal whether the bottom two luminance blocks and each of the chrominance blocks were coded (the top two luminance blocks are...
not adjacent to any blocks further along in the scan and can be ignored). For CIF-sized frames this amounts to \((4 + 1) \times (22 + 1) = 115\) bits of storage per layer.

For the off-line encoder (Figure 4.11) and decoder (Figure 4.12) extra storage must be allocated to perform intra coefficient prediction. For the on-line configurations this information is already stored in the corresponding single-layer encoder or decoder, however, this will need to be kept separately if the morphing process is occurring outside of the context of these entities. In the worst case (every macroblock in the last row was coded in intra mode) the first row and column of unquantized DCT coefficients must be stored for the previous macroblock where either horizontal or vertical prediction may be performed, 15 values in all at 12 bits per value or 180 bits, and the first row of coefficients \((8 \times 12 = 96\) bits per macroblock) for the previous \(N\) macroblocks where only vertical prediction is possible, plus another bit of storage to indicate whether each macroblock was coded in intra or inter mode. For CIF-sized frames (22 macroblocks per row) this is \(180 + 21 \times 96 + 22 = 2218\) bits of storage per layer.

### 4.5.7 Optimizations

It should be noted that many (but not all) of the computationally-expensive division operations described here, notably the calculation of \(D_{rate}\) and \(D_{qp}\), work with divisors that are within a limited range (less than 100) and as such can be implemented using multiplications via a lookup table containing fixed point representations of the inverses of all possible divisor values. This is also true of the quantization operations used on the intra coefficient ranges in order to determine the number of quantization bins the range spans in the current layer (Section 4.3.6.1). If we replace each of these division operations by a fixed-point approximation we can replace each of them with a single multiply and a shift which reduces the computational requirements summarized in Table 4.6 to that shown in Table 4.7.
4.6 Experimental Procedure

This section will present experimental results for the stream morphing system as just described under conditions of constant quantization in each layer. Comparisons are provided with MPEG-2-compliant SNR scalability and MPEG-4 FGS. The test conditions are as follows:

- Five layers are used for those techniques with discrete layers (stream morphing and the MPEG-2-compliant SNR scalability).
- All sequences begin with a single I-VOP with P-VOPs for all frames thereafter. Intra macroblocks are used in P-VOPs where the sum of absolute difference of the prediction difference in the luminance blocks is larger than the sum of absolute difference of the original pixels relative to the mean pixel value by some threshold value (this is the method used in the MPEG-4 VM software [31]).
- H.263 quantization is used.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Multiply</th>
<th>Divide</th>
<th>Add</th>
<th>Subtract</th>
<th>Shift</th>
</tr>
</thead>
<tbody>
<tr>
<td>Symbol Encode</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>Symbol Decode</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Renormalization</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>QP and $D_{rate}$ calculation</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$D_{qp}$ for existing/restored inter</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Existing intra DC (one region)</td>
<td>6</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>8</td>
</tr>
<tr>
<td>Existing intra DC (&gt;1 region)</td>
<td>9</td>
<td>0</td>
<td>3</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>Existing intra AC (one region)</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>Existing intra AC (&gt;1 region)</td>
<td>5</td>
<td>0</td>
<td>2</td>
<td>4</td>
<td>6</td>
</tr>
</tbody>
</table>

Table 4.7: Operation summary with fixed-point implementation of division with restricted divisors
• Motion estimation is done using full search in all cases with search ranges of 16 pixels (QCIF), 32 pixels (CIF) or 64 pixels (720x480 or 720x576).

For the base layer, the lowest possible bit rate (quantizer step size of 31) is used in all cases. Since this layer needs to be protected when transmitted over the channel, it is desirable that the cost of this protection (which is proportional to the bit rate) be as small as possible. The quantizer step sizes in the other four layers are adjusted to spread those layers over approximately twice the rate of the base layer.

All results presented here are from bitstreams that have been successfully decoded; in addition for the techniques with discrete layers the encoder reconstruction has also been verified to be identical to the decoder output.

Sequences such as “Akiyo” which have complex static backgrounds need to be treated with care. Those static areas are generally identical in all frames, since the error signal in predicted frames is not large enough to cause any new texture to be coded in those frames. As such, a poor quality first frame has a disproportionate impact on the overall quality of the entire sequence. This is important for the MPEG-2-compliant SNR scalability since the quantization effects, as previously described, result in a poor quality first frame if the same quantizer is used there as for the remainder of the sequence. For “Akiyo” as well as “Mother & Daughter” the quantizer step size used in the first frame of the MPEG-2-compliant SNR scalable case has been adjusted independently of the one used for the remainder of the sequence so that the PSNR of that frame is approximately identical to those in the other frames.

4.7 Results

Figures 4.13, 4.14, 4.15, 4.16 and 4.17 show rate-distortion curves for the sequences “Akiyo”, “Carphone”, “Coastguard”, “Foreman” and “Mother & Daughter” respectively at 10fps CIF. For other frame sizes and rates we only show one
4.7 Results

sequence with a relative low bit rate ("Mother & Daughter") and one sequence requiring higher rates ("Foreman"). Figures 4.18 and 4.19 show results for the selected sequences at 30fps CIF while Figures 4.20 and 4.21 give results for QCIF at 30fps. Finally, Figures 4.22 and 4.23 show performance at full Rec. 601 (720x576 25fps or 720x480 30fps).

For the 10fps tests stream morphing is shown twice, once with the coefficient adjustment algorithm of Section 4.3.6.4 used to prevent opposite coefficient value changes in consecutive layers and a second time with this disabled. The results indicate a significant improvement in the performance at the top layer and at most a negligible drop in quality in the lower enhancement layers.

Note that as with the results shown in the previous chapter, half-pel motion estimation is done using the reconstruction from the top layer for stream morphing and MPEG-2-compliant SNR scalability. The FGS results presented here use the base layer reconstruction as the half-pel reference which is the behaviour of the MPEG-4 VM implementation and is consistent with the fact that prediction is only ever done in the base layer. As such the quality of the base layers for stream morphing and MPEG-2-compliant SNR scalability are not the same as to single-layer or FGS as different motion vectors are being used.

Figures 4.24 and 4.25 show the effect of the loss of the top two layers after 20 frames on the decoding of five layer stream morphing sequences corresponding to "Foreman" in Figure 4.16 and "Mother & Daughter" in Figure 4.17. In the stream morphing case it is possible to simulate the loss of the top two layers by splicing together the bitstream segment corresponding to the first 20 frames of the top layer and the last 80 frames of the middle layer bitstream. This is equivalent to the operation performed on the same sequences in MPEG-2-compliant SNR scalability in Figures 3.21 and 3.22 and shows that stream morphing has a similar lack of drift when not all layers are received. These encoder architectures are very similar in the sense that they both have one motion-compensated prediction loop per layer, the primary difference with stream morphing is how
4.7 Results

Constant Quantizer results for Akiyo (GIF 10fps)

(a) Comparative performance

(b) Video quality in enhancement layers

Figure 4.13: Constant quantizer results for "Akiyo" (CIF 10fps)
Figure 4.14: Constant quantizer results for “Carphone” (CIF 10fps)

(a) Comparative performance

(b) Video quality in enhancement layers
4.7 Results

Constant Quantizer results for Coastguard (CIF 10fps)

(a) Comparative performance

(b) Video quality in enhancement layers

Figure 4.15: Constant quantizer results for “Coastguard” (CIF 10fps)
4.7 Results

Constant Quantizer results for Foreman (GIF 1Ofps)

(a) Comparative performance

(b) Video quality in enhancement layers

Figure 4.16: Constant quantizer results for “Foreman” (CIF 10fps)
4.7 Results

Constant Quantizer results for Mother & Daughter (CIF 10fps)

(a) Comparative performance

(b) Video quality in enhancement layers

Figure 4.17: Constant quantizer results for “Mother & Daughter” (CIF 10fps)
4.7 Results

Constant Quantizer results for Foreman (GIF 30fps)

32.5
32
31.5
31
30.5
OC 30 z CO D.

(a) Comparative performance

(b) Video quality in enhancement layers

Figure 4.18: Constant quantizer results for “Foreman” (CIF 30fps)
4.7 Results

Constant Quantizer results for Mother & Daughter (CIF 30fps)

(a) Comparative performance

(b) Video quality in enhancement layers

Figure 4.19: Constant quantizer results for "Mother & Daughter" (CIF 30fps)
4.7 Results

Constant Quantizer results for Foreman (QCIF 30fps)

(a) Comparative performance

(b) Video quality in enhancement layers

Figure 4.20: Constant quantizer results for “Foreman” (QCIF 30fps)
4.7 Results

Constant Quantizer results for Mother & Daughter (QCIF 30fps)

(a) Comparative performance

(b) Video quality in enhancement layers

Figure 4.21: Constant quantizer results for "Mother & Daughter" (QCIF 30fps)
4.7 Results

Constant Quantizer results for Foreman (Rec. 601 [720x576] 25fps)

(a) Comparative performance

(b) Video quality in enhancement layers

Figure 4.22: Constant quantizer results for "Foreman" (720x576 25fps)
4.7 Results

Figure 4.23: Constant quantizer results for “Mother & Daughter” (720x480 30fps)

(a) Comparative performance

(b) Video quality in enhancement layers
Stream Morphing: Drift after loss of top two layers starting at frame 20 (Foreman)

Figure 4.24: Loss due to drift (Foreman)

Stream Morphing: Drift after loss of top two layers starting at frame 20 (Mother & Daughter)

Figure 4.25: Loss due to drift (Mother & Daughter)
### Operation Count

<table>
<thead>
<tr>
<th>Operation</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Symbol Encode/Decode</td>
<td>1618850</td>
</tr>
<tr>
<td>Renormalization</td>
<td>183322</td>
</tr>
<tr>
<td>QP and $D_{rate}$ calculation</td>
<td>158400</td>
</tr>
<tr>
<td>$D_{qp}$ for existing/restored inter</td>
<td>237710</td>
</tr>
<tr>
<td>Existing intra DC (one region)</td>
<td>5590</td>
</tr>
<tr>
<td>Existing intra DC (&gt;1 region)</td>
<td>17024</td>
</tr>
<tr>
<td>Existing intra AC (one region)</td>
<td>7914</td>
</tr>
<tr>
<td>Existing intra AC (&gt;1 region)</td>
<td>13534</td>
</tr>
</tbody>
</table>

Table 4.8: Operation counts for Foreman (CIF 10fps)

the prediction difference is coded. The behaviour of these systems is therefore very similar when not all layers are received.

Tables 4.8 and 4.9 show measured operation counts for the morphing of five layer "Foreman" and "Mother & Daughter" using the same parameters as the results shown in Figures 4.16 and 4.17 respectively. These have been compiled using the categories defined originally in Tables 4.6 and 4.7. Combining the counts shown in Table 4.16 for the "Foreman" sequence with Table 4.7 allows the calculation of the number of primitive operations used in encoding and decoding the sequence which is shown in Table 4.10. Likewise, the counts for the lower rate sequence "Mother & Daughter" are shown in Table 4.9 resulting in the primitive operation counts shown in Table 4.11.

Finally, Tables 4.12 and 4.13 show bitstream statistics for stream morphing with a comparison against 5 layer MPEG-2-compliant SNR scalability. Tables 3.4 and 3.5 showed additional statistics for the same sequences in the 3 layer cases of the MPEG-2 scalability as well as single-loop. To reiterate from the previous chapter: taking these counts is designed to show that scalable systems are subject to an inevitable "spreading" of coefficients into blocks that are not coded in the corresponding single-layer equivalent at the level of quality of the top
4.7 Results

<table>
<thead>
<tr>
<th>Operation</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Symbol Encode/Decode</td>
<td>896276</td>
</tr>
<tr>
<td>Renormalization</td>
<td>63871</td>
</tr>
<tr>
<td>QP and $D_{\text{rate}}$ calculation</td>
<td>158400</td>
</tr>
<tr>
<td>$D_{dq}$ for existing/restored inter</td>
<td>59437</td>
</tr>
<tr>
<td>Existing intra DC (one region)</td>
<td>3373</td>
</tr>
<tr>
<td>Existing intra DC (&gt;1 region)</td>
<td>6731</td>
</tr>
<tr>
<td>Existing intra AC (one region)</td>
<td>3456</td>
</tr>
<tr>
<td>Existing intra AC (&gt;1 region)</td>
<td>5226</td>
</tr>
</tbody>
</table>

Table 4.9: Operation counts for Mother & Daughter (CIF 10fps)

<table>
<thead>
<tr>
<th></th>
<th>Multiply</th>
<th>Divide</th>
<th>Add</th>
<th>Subtract</th>
<th>Shift</th>
</tr>
</thead>
<tbody>
<tr>
<td>Encoder</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Overall (10s)</td>
<td>4062464</td>
<td>0</td>
<td>2049892</td>
<td>467270</td>
<td>4511596</td>
</tr>
<tr>
<td>Approx. Rate</td>
<td>406k/s</td>
<td>0</td>
<td>205k/s</td>
<td>47k/s</td>
<td>451k/s</td>
</tr>
</tbody>
</table>

|                                |          |        |         |           |       |
| Decoder                        |          |        |         |           |       |
| Overall (10s)                  | 4062464  | 1618850| 431042  | 2086120   | 4511596 |
| Approx. Rate                   | 406k/s   | 162k/s | 43k/s   | 209k/s    | 451k/s |

Table 4.10: Overall operation summary for Foreman (4 enhancement layers, CIF 10fps)
<table>
<thead>
<tr>
<th></th>
<th>Multiply</th>
<th>Divide</th>
<th>Add</th>
<th>Subtract</th>
<th>Shift</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Encoder</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Overall (10s)</td>
<td>2282648</td>
<td>0</td>
<td>1155938</td>
<td>245472</td>
<td>2341476</td>
</tr>
<tr>
<td>Approx. Rate</td>
<td>228k/s</td>
<td>0</td>
<td>116k/s</td>
<td>25k/s</td>
<td>234k/s</td>
</tr>
<tr>
<td><strong>Decoder</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Overall (10s)</td>
<td>2282648</td>
<td>896276</td>
<td>259662</td>
<td>1141748</td>
<td>2341476</td>
</tr>
<tr>
<td>Approx. Rate</td>
<td>228k/s</td>
<td>90k/s</td>
<td>26k/s</td>
<td>114k/s</td>
<td>234k/s</td>
</tr>
</tbody>
</table>

Table 4.11: Overall operation summary for Mother & Daughter (4 enhancement layers, CIF 10fps)

layer, as originally discussed in Section 3.7.3.2. These tables show that stream morphing limits the extent of this spreading more effectively than the MPEG-2 approach, evident here in the reduced numbers of discrete non-zero coefficients, blocks and macroblocks which acts to reduce the number of symbols that must be coded and therefore the overall bit rate. Note that unlike the results for MPEG-2 SNR scalability shown here and also Tables 3.4 and 3.5 no sums are shown for stream morphing that are weighted by the number of layers the entity is present in. The true cost of coding each discrete entity in the new system is proportional to the number of times that changes must be signalled, not the number of layers in which they are non-zero and as such the weighted measure is not useful. Tables 4.12 and 4.13 also break down the bit rates for each layer to show the contributions of macroblock overhead separately from the coding of new coefficients in the layer. For stream morphing enhancement layers, the portion of the bitstream used in the updating of coefficients already present in previous layers is also listed. Note that the number of bits required to code motion vectors is approximately the same across all methods and are included in the totals but are not included in the breakdowns for the base layers.
4.7 Results

<table>
<thead>
<tr>
<th>Layer</th>
<th>QP=12</th>
<th>5-layer MPEG-2-Compliant SNR Scalability (pyramid)</th>
<th>5-layer Stream Morphing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base (QP=31)</td>
<td>27.59</td>
<td>27.75</td>
<td>27.75</td>
</tr>
<tr>
<td>1st enh. (QP=19)</td>
<td>29.65</td>
<td>30.65</td>
<td>29.45</td>
</tr>
<tr>
<td>2nd enh. (QP=15)</td>
<td>30.79</td>
<td>31.28</td>
<td>30.65</td>
</tr>
<tr>
<td>3rd enh. (QP=13)</td>
<td>31.54</td>
<td>31.28</td>
<td>31.28</td>
</tr>
<tr>
<td>4th enh. (QP=12)</td>
<td>31.95</td>
<td>31.95</td>
<td>31.95</td>
</tr>
<tr>
<td>Total</td>
<td>76.5</td>
<td>51.6</td>
<td>76.5</td>
</tr>
</tbody>
</table>

Table 4.12: Bitstream statistics for Foreman (GIF 10fps)
### 4.7 Results

#### Layer PSNR (dB) Rates (kbps) Coefficients
<table>
<thead>
<tr>
<th>Layer</th>
<th>PSNR (dB)</th>
<th>Overhead</th>
<th>Existing</th>
<th>New</th>
<th>Total Ix</th>
<th>1x</th>
<th>2x</th>
<th>3x</th>
<th>4x</th>
<th>5x</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single-layer</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>QP=10</td>
<td>36.40</td>
<td>13.1</td>
<td>-</td>
<td>22.8</td>
<td>54.5</td>
<td>34795</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

#### 5-layer MPEG-2-Compliant SNR Scalability (pyramid)

| Base (QP=31) | 31.10 | 10.1 | - | 2.5 | 30.5 | 2819 | - | - | - | - |
| 1st enh. (QP=15,11) | 34.26 | 8.5 | - | 10.2 | 18.7 | 15589 | 1611 | - | - | - |
| 2nd enh. (QP=12,10) | 35.17 | 9.0 | - | 9.7 | 18.7 | 27008 | 2960 | 220 | - | - |
| 3rd enh. (QP=11,9) | 35.75 | 9.2 | - | 9.4 | 18.6 | 34493 | 5865 | 513 | 6 | - |
| 4th enh. (QP=10,8) | 36.29 | 9.2 | - | 9.9 | 19.1 | 41019 | 9508 | 754 | 31 | 0 |
| Total | 46.0 | - | 41.7 | 105.6 | 62421 | 51312 discrete |

#### 5-layer Stream Morphing

| Base (QP=31) | 31.25 | 10.3 | - | 3.3 | 30.7 | 2498 | - | - | - | - |
| 1st enh. (QP=15) | 34.06 | 4.8 | - | 0.9 | 6.6 | 12.3 | 13279 | 2071 | - | - |
| 2nd enh. (QP=12) | 35.24 | 4.0 | - | 1.2 | 6.9 | 12.1 | 10685 | 12628 | 1960 | - |
| 3rd enh. (QP=11) | 35.63 | 2.6 | - | 1.5 | 4.7 | 8.8 | 6883 | 11279 | 11299 | 1878 |
| 4th enh. (QP=10) | 36.40 | 3.9 | - | 2.5 | 8.3 | 14.7 | 12362 | 10420 | 8501 | 9895 | 1783 |
| Total | 25.6 | 6.1 | - | 29.8 | 78.6 | 42961 discrete |

#### Table 4.13: Bitstream statistics for Mother & Daughter (CIF 10fps)
4.8 Discussion

The results presented above show that stream morphing outperforms MPEG-2-compliant SNR scalability for all sequences tested. There are two primary reasons for this:

1. Tables 4.12 and 4.13 clearly show that stream morphing can achieve similar objective qualities as MPEG-2-compliant SNR scalability while using fewer discrete coefficients over all layers that are located in fewer 8x8 blocks. While this comes at the expense of additional bandwidth required to update coefficient values between layers this additional cost is small.

2. Macroblock overhead is significantly reduced in moving to the arithmetic coded bitstream syntax. As we will show in Chapter 5, this is even more crucial when rate control is added and the overhead associated with quantizer step size change on a per-macroblock basis is included. “Low-motion” sequences where macroblock overhead is a larger proportion of the overall bitstream such as “Akiyo” and “Mother & Daughter” benefit most from this reduction. These gains come at the expense of additional implementation complexity required for arithmetic coding compared with VLCs which is detailed in Tables 4.10 and 4.11. Encoding and decoding VLCs is a relative simple operation which does not require complex operations such as integer multiplication (and also division in the decoder) that are associated with arithmetic or range coding. While the complexity of stream morphing is higher than MPEG-2-compliant SNR scalability, we believe that the difference is small in the context of a complete system which contains DC-T/IDCT operations and motion-compensation. When compared to MPEG-4 FGS, stream morphing and MPEG-2-compliant SNR scalability have a significant advantage in terms of reduced decoder complexity due to the use of only one IDCT rather than two.
Over the range of frame sizes and rates tested, the relative performance of the two scalable coding methods are very similar. For "Mother & Daughter" the test at 10fps CIF resulted in the best relative performance for MPEG-2-compliant SNR scalability (measured by considering the gain over the four enhancement layers compared to that achieved by stream morphing). This is due to the fact that the macroblock overhead is a smaller proportion of the total bitstream: consider that at 10fps CIF having one bit per macroblock for the macroblock skipped VLC requires 3.96kbit/s per layer which is approximately one third of the enhancement layer rates whereas for the other cases the macroblock skipped overhead is closer to a half of the total enhancement layer rate. As such more bits are available in this case for coding coefficients.

Stream morphing only quantizes the input signal once, inter-layer correlation is exploited at the quantized coefficient level without having to compute an unquantized error signal which then needs to be quantized again. In this way, all coefficients are refined in enhancement layers with a smaller quantizer step size, unlike MPEG-2-compliant SNR scalability where significant coefficients remain coarsely quantized due to the fact that the error signal must be larger than the quantizer dead zone in order for a coefficient to be updated (Section 3.7.3.2). For this reason the subjective performance of stream morphing is better than the objective measurements shown here would suggest which will be demonstrated with subjective tests in Chapter 5.

Finally, the comparisons provided with MPEG-4 FGS in Figures 4.13-4.23 show that in every case tested (except for very low rates just above the base layer rate) the performance of stream morphing is higher even if at intermediate bit rates we must "back off" to the rate corresponding to the nearest complete layer and not use all the available bandwidth. While stream morphing does not provide "fine" granularity in the sense that FGS does it can be seen that this is not necessary to produce superior performance at almost any operating point we choose.
4.9 Conclusion

This chapter introduced the basic concepts behind the stream morphing method and provided results for the constant quantizer case that show it achieves consistently superior performance to MPEG-2-compliant SNR scalability. This is in addition to the increased flexibility that results from the use of standard single-layer bitstreams as inputs and outputs in all layers and the ability to generate and recover those streams outside the context of their encoding and decoding. The next chapter considers the application of rate control to stream morphing for CBR applications.
Chapter 5

Rate Control for Stream Morphing and MPEG-2-Compliant SNR Scalability

5.1 Introduction

Section 2.6 introduced the need for control over quantization to manage the volume of compressed data generated by a video encoder, while maintaining uniform quality over each individual video frame and ideally over the whole sequence. For stream morphing, as described in the previous chapter, to be applied to scenarios where bandwidth limitations exist, a rate control method must be developed to suit it. Stream morphing has been formulated in a very different way from other forms of scalability, such as the MPEG-2-compliant SNR scalability which retains an almost identical syntactic structure to a standard single-layer encoder and as such we should not expect existing rate control algorithms to be useful for the new scheme.

Section 5.2 begins by describing previous approaches to the rate control problem for single-layer video encoders. A new single-layer rate control algorithm based on pre-analysis of unquantized DCT coefficients is then developed, which is designed for use at low bit rates where the volume of side information, such as motion vectors and coded block patterns, often exceeds that generated by DCT coefficients alone. This rate controller forms the basis for a framework that is used with other encoder architectures. Section 5.3 describes the exten-
sion of the single-layer rate control scheme for use with MPEG-2-compliant SNR scalability. In this case only minor changes to the single-layer rate model are needed due to the similarities between the coding of enhancement layers of this type and the basic single-layer encoder. Section 5.4 goes on to define a rate model for stream morphing which is then inserted into the basic framework. Sections 5.5, 5.6 and 5.7 describe and provide experimental results comparing the various scalability schemes under CBR conditions. Section 5.8 draws final conclusions from this chapter.

Although we are primarily interested in the behaviour of stream morphing, having implementations of an essentially identical algorithm for the single-layer and MPEG-2-compliant SNR scalable cases provides a useful basis for comparisons between these methods. The single-layer rate control scheme developed here can also be used for the base layer in an MPEG-4 FGS system which also assists in making comparisons with the other scalable coding schemes. Recall that rate control is not an issue in the enhancement layer for MPEG-4 FGS since the bitstream can be truncated at any point without introducing drift.

5.2 Rate Control for Single-layer MPEG-4 Encoders

5.2.1 Previous Approaches

The rate control problem is necessarily a trade-off between pre-analysis (i.e. any processing done before the choice of macroblock quantizer step sizes is made), allowing for more accurate characterization of the source material, and the desire to keep computation and encoder latency to a minimum.

At one end of the spectrum [56] demonstrates a rate control scheme based on dynamic programming [57] which requires the input sequence to be coded and analysed many times before an "optimal" solution to the rate control problem is achieved. A number of other schemes have been proposed with lower complexity which, however, require a number of passes through a given video
5.2 Rate Control for Single-layer MPEG-4 Encoders

frame or GOP in order to model the rate-distortion relation [58] or the single quantization value required to meet a given bit target [59].

In contrast, Test Model 5 (TM5) for MPEG-2 [60] provides what is perhaps one of the least computationally-expensive rate control algorithms possible. The quantizer step size is calculated solely as a function of decoder buffer fullness $d_i$ (Section 2.6); for macroblock $i$ of $N$ total:

$$Q_i = \frac{d_i \times 31}{r}$$  \hspace{1cm} (5.1)

where $r = 2^\frac{R}{F}$ is the so-called reaction parameter, $R$ is the target bit rate and $F$ is the frame rate. If the buffer fills faster than the channel rate then $d_i$ increases which causes the later values for $Q_i$ to increase, which acts to lower the bit rate and thus empty the buffer. Likewise, a lower than expected rate empties the buffer and lower $Q_i$ causing the instantaneous bit rate to rise. Ideally $Q_i$ remains as constant as possible over a frame (an additional term is added to code perceptually important macroblocks more accurately) which assumes the number of bits generated per macroblock is approximately constant over the frame.

TM5 was originally designed for MPEG-2, which is optimized for high bit rates and qualities (low quantizer step size $Q_i$). The simple relationship between $Q_i$ and the buffer fullness given in equation 5.1 does not take into account the nonlinear relationship between quantizer step size and output bit rate. For effective control at large quantizer step size values (low quality), $Q_i$ needs to vary over a wider range than when small step sizes are used, where slight changes to $Q_i$ have a much greater effect on the output rate. As such there is insufficient feedback from the buffer fullness at low bit rates which means that the overall bit target for each frame is rarely achieved. This is shown in Figure 5.1 which shows the number of bits per frame generated in the encoding of the "Mother & Daughter" sequence using TM5 rate control at a low rate of 56kbps (CIF 10fps) where there is significant variation in the number of bits produced from frame-to-frame. Such variation necessitates a large amount of decoder
5.2 Rate Control for Single-layer MPEG-4 Encoders

Figure 5.1: TM5 rate control: example of non-uniform bit allocation at low rates

buffering (and associated delay) when transmitting over a CBR channel. On the positive side, the only pre-analysis done by TM5 are variance measurements in the pixel domain on the original input frame (for use with perceptual quantization) which can be done as soon as a new input frame is received and does not require any other work to be done before decisions on quantizer step size values can be made. This serves to decrease the latency of the system when compared to other techniques that require information about the prediction difference and/or the DCT coefficients.

Rather than solely using the buffer fullness to regulate the quantizer step size, many other algorithms attempt to model the rate-distortion relationship of the source material. [61], which forms the basis for a number of the multiple video object rate control techniques trialed during the MPEG-4 standardization process [62], attempts to model the number of bits generated by a frame $R$ using:

$$R = \alpha Q^{-1} + \beta Q^{-2}$$ (5.2)
where $Q$ is the average quantization step size in the frame and $\alpha$ and $\beta$ are parameters calculated using linear regression on all previous frames' actual bit counts and average quantizer step size used.

[63] describes a more elaborate scheme that specifies the quantizer step size to be used at the macroblock (rather than frame) level which also incorporates rate feedback that is not present in the previous method. The number of bits generated by each macroblock is modelled by:

$$B_i = A \left( K \frac{\sigma_i^2}{Q^2} + C \right)$$

(5.3)

where $\sigma_i^2$ is the variance of the motion-compensated prediction difference in the macroblock $i$ and $A$, $K$ and $C$ are constants. This eventually leads to the following closed-form expression for an individual macroblock quantizer step size $Q_i$:

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

(5.4)

where $L$ is the number of bits left in the frame for texture (computed from the total bit target for the frame from which is subtracted the predicted amount of macroblock overhead, proportional to $C$, expected for the rest of the frame), $\alpha_i$ is a macroblock weighting factor that can be used to ensure perceptually-important areas of the image are coded more accurately than others and $S_i$ is the sum of all $\sigma_i \alpha_i$ products over the remaining macroblocks after the current one. It is important to note that this algorithm, unlike the TM5 rate control, introduces a single frame processing delay in most implementations since $\sigma_i$ needs to be evaluated for all macroblocks before any decisions on the value of any quantization step sizes can be made as $S_i$ (i.e. $S_i$ in the first macroblock as required to evaluate equation 5.4) is not available until this is done. The delay is due to the fact that the motion-compensated prediction difference is required for the entire frame which means that the motion estimation process must also be complete for the entire frame. On systems with dedicated hardware for performing motion estimation, it is undesirable to have this idle for...
any significant period of time. Therefore, in order to pipeline the various encoder stages, it would be necessary to perform motion estimation, prediction and computing the difference for one frame while performing the rest of the operations (DCT, quantization etc.) on the previous frame after the information for performing rate control on that frame has become available. Even for systems without dedicated hardware, such as workstations or PCs, delaying the output of the bitstream until motion estimation has finished requires a delay which is a significant fraction of the frame period. This may be unacceptable for some applications, such as videoconferencing, where end-to-end delays can be distracting for users. Note also that the prediction difference needs to be stored between stages if it is not to be recalculated, which requires extra storage and memory bandwidth at the encoder.

The use of spatial-domain measures, such as pixel variance used in [63], is simple to implement but is often a poor predictor of the number of bits that are generated by typical image and video coders [64]. [65] advocates the measurement of the number of non-zero quantized DCT coefficients for the estimation of the rate-distortion behaviour of video. This is based on the fact that there is a one-to-one relation between quantized coefficients and VLCs for non-zero coefficient (run,level,end) tuples (Section 2.3.3.1) in the bitstream. Since there is a strong correlation between the number of these VLCs being generated in a given period of time and the overall bit rate, there must be a similar correlation between the number of coefficients and the bit rate. The number of quantized coefficients for a particular quantizer step size can be calculated by comparing the absolute value of each unquantized coefficient to the threshold value corresponding to the edge of the central (zero) quantization bin. This thresholding operation is easy to compute for many possible values of the step size since it is only one comparison per threshold plus one absolute value operation per coefficient. Like the previous method, the analysis of unquantized coefficients requires that the encoding process be split into two stages and to prevent
Figure 5.2: Modified single-layer encoder with rate control

having to re-compute the DCT coefficients when generating the bitstream the unquantized coefficients must be stored after thresholding.

The new rate control algorithm to be presented here is based on a similar analysis of unquantized DCT coefficients. It differs from similar complete algorithms that have been previously developed [66,67] in that it models macroblock header information as well as DCT coefficients and also reduces the computational requirements associated with thresholding by using a subset of possible thresholds and interpolating between the results obtained at those points. The algorithm will also be applied to scalable video encoders in addition to the single-layer case where it can be argued that DCT-based techniques must be used. The pyramid configuration of the MPEG-2-compliant SNR scalable encoder of Figure 3.4 does all enhancement layer processing in the DCT domain and stream morphing itself does no processing whatsoever in the pixel domain.
5.2 Rate Control for Single-layer MPEG-4 Encoders

5.2.2 Encoder Architecture

Figure 5.2 shows a modified version of the standard MPEG-4 single-layer encoder originally shown in Figure 2.11 which collects unquantized DCT coefficients in a second frame store then completes the encoding of the frame in a second pass. This structure will also be used for each layer of the stream morphing encoder shown in Figure 4.1. The rate controller is explicitly marked in this version of the diagram. As well as the unquantized DCT coefficients, the rate control block has a number of other inputs: the original image and motion-compensated prediction difference for use with perceptual quantization (to be described in Section 5.2.4 below) and also the motion vectors. Motion vectors sometimes make up more than half the bitstream at low bit rates and since we are completing the motion estimation process in advance of doing any rate control, it is possible to know in advance almost exactly how many bits are required to code these in the final bitstream.

A second motion-compensated prediction block has been added to regenerate the prediction for determining the reconstruction which is now performed in the second pass through the data. The rate control algorithm to be presented has a feature which attempts to control the number of bits used for motion vectors by disabling the use of four motion vectors per block if the number of bits to code the motion vectors requires too large a fraction of the overall bit budget. If this is done then the motion-compensated prediction used in the thresholding process is no longer valid and must be re-computed before the second pass. If this feature is disabled then it would be possible to store the prediction between encoder stages rather than to regenerate it.

5.2.3 Bit Production Model

To model the number of bits generated by a single video frame, we divide up the bits generated by the MPEG-4 encoder into three categories:
5.2 Rate Control for Single-layer MPEG-4 Encoders

Motion vectors: "Foreman" GIF 10fps, QP=31

Figure 5.3: Non-zero motion vector bits

Macroblock overhead: "Foreman" CIF 10fps, QP=31

Figure 5.4: Bits for macroblock overhead
5.2 Rate Control for Single-layer MPEG-4 Encoders

1. Bits generated in coding non-zero (differential) motion vectors.

2. Bits attributed to macroblock overhead: coded block patterns (which includes the signalling of the macroblock prediction mode), differential quantizer step size VLCs and the macroblock skipped bit in each P-VOP or B-VOP macroblock.

3. VLCs describing non-zero quantized DCT coefficients.

Sequence and VOP headers are ignored since they make up a negligible fraction of the total number of bits for typical sequences.

The reason for distinguishing between zero and non-zero motion vectors is that in the MPEG-4 syntax, a zero motion vector may not be explicitly coded at all if the macroblock is skipped, which only occurs if the full motion vector is zero (note that the corresponding differential motion vector that is coded in the not-skipped case may not be zero). After motion estimation has been
5.2 Rate Control for Single-layer MPEG-4 Encoders

performed, the rate controller knows which macroblocks have a zero motion vector but cannot tell which of these macroblocks will be skipped since this depends on the number of non-zero coefficients which is itself a function of the as-yet-unselected quantizer step size for each of those macroblocks.

Figure 5.3 shows a plot of the bits required for coding differential motion vectors with a single-layer MPEG-4 encoder on the "Foreman" sequence (at 10fps CIF) with a constant quantizer step size of 31. Figures 5.4 and 5.5 show the number of bits required for macroblock overhead and DCT coefficient VLCs for the same sequence. Note that for this "high-motion" sequence at such a low bit rate motion vectors make up more than half of the overall bitstream and the macroblock overhead is also of significant size in proportion to the bits required for DCT coefficients.

This example indicates that for low bit rates if an application can tolerate a single frame processing delay an approximate number of bits required for motion vectors can be determined which can make the rate control task much easier given that a large proportion of the bitstream is effectively taken out of the rate control problem. Indeed we would expect that any rate control scheme that does not attempt this type of analysis will experience problems with sudden changes in the amount of movement (and hence the number of bits required to code motion vectors) in a video sequence.

As described previously, thresholding of the unquantized DCT coefficients can be used before quantization to estimate how many non-zero coefficients are going to be created. Figure 5.6 shows the number of non-zero coefficients for intra- and inter-coded macroblocks for the coding of the "Foreman" sequence at the minimum rate. Figures 5.7 and 5.8 show the number of bits generated per intra and inter coefficient respectively. These plots are obtained by dividing the total number of bits for DCT coefficients shown in the appropriate part of Figure 5.5 by the number of non-zero coefficients in each frame in Figure 5.6. Note that some frames do not have any intra-coded coefficients: Figure 5.7 only
5.2 Rate Control for Single-layer MPEG-4 Encoders

Non-zero coefficients: "Foreman" CIF 10fps, QP=31

Figure 5.6: Number of non-zero coefficients

Intra texture per coefficient: "Foreman" CIF 10fps, QP=31

Figure 5.7: Bits per non-zero intra coefficient
5.2 Rate Control for Single-layer MPEG-4 Encoders

Figure 5.8: Bits per non-zero inter coefficient

plots values for those frames which have at least one intra coefficient. Note also that the discontinuities in Figure 5.7 occur for macroblocks where there are very few coefficients and as such the average is only taken over a small number of macroblocks. These results indicate that a simple linear model relating non-zero quantized DCT coefficients and the number of bits used to code texture is likely to work well in practice. If the gradient of the linear relationship is updated at the end of each frame the use of a linear relationship does not require that the number of bits per coefficient remain constant over time (which it clearly does not in this case, compare frames 70 and 90 in Figure 5.8) but rather that the value of this quantity in the previous frame is a good predictor of its value in the current frame.

Similarly, it was observed that the macroblock overhead is approximately proportional to the number of coded macroblocks of each type. Note that all intra macroblocks are deemed to be coded even if there are no non-zero coef-
5.2 Rate Control for Single-layer MPEG-4 Encoders

Macroblocks: "Foreman" CIF 10fps, QP=31

Figure 5.9: Number of coded macroblocks

Intra overhead per macroblock: "Foreman" CIF 10fps, QP=31

Figure 5.10: Overhead per intra macroblock
5.2 Rate Control for Single-layer MPEG-4 Encoders

Inter overhead per macroblock: "Foreman" CIF 10fps, QP=31

Figure 5.11: Overhead per inter macroblock

coefficients since there is no macroblock skipping mechanism for these and coded block patterns are generated in all cases. Figure 5.9 shows the number of intra-and inter-coded macroblocks in the example sequence and Figures 5.10 and 5.11 show the macroblock overhead data of Figure 5.4 normalized by these macroblock counts.

The results of Figures 5.7, 5.8, 5.10 and 5.11 show that both the macroblock overhead and number of bits required for DCT coefficients can be effectively estimated by linear functions of quantities that a rate control algorithm can measure in the first pass of an encoder similar to that shown in Figure 5.2.

Many macroblocks, especially at low bit rates, will be assured of having no non-zero coefficients given particular choice for $Q_i$, the quantizer step size in the $i$th macroblock. Even if $Q_i$ does not correspond exactly to one of the threshold quantizer step size values used, we can still be certain there are no non-zero coefficients if the coefficient counts made at the closest surrounding thresholds
5.2 Rate Control for Single-layer MPEG-4 Encoders

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>No</td>
<td>No</td>
<td>1</td>
</tr>
<tr>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>9</td>
</tr>
<tr>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>3</td>
</tr>
<tr>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>5</td>
</tr>
<tr>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>9</td>
</tr>
</tbody>
</table>

Table 5.1: MPEG-4 fixed overhead for inter macroblocks with no non-zero DCT coefficients

are zero due to the monotonicity of the count of non-zero DCT coefficients with respect to quantizer step size. The number of bits required to code these macroblocks is entirely dependent on the macroblock overhead which is itself dependent only on whether the motion vector in the block is non-zero and whether the differential quantizer step size for this macroblock is to be non-zero, both of which are known by the rate control algorithm when the choice of $Q_i$ is made. The amount of overhead can vary considerably: for example, if a macroblock is skipped then it requires only one bit to code whereas a macroblock with no texture and no quantizer step size change but a non-zero motion vector requires four bits (one bit each for macroblock skipped and the coded block pattern (chrominance) and two bits for the coded block pattern (luminance)) plus those required to code the differential motion vectors themselves. As discussed previously, the number of bits for motion vectors could be calculated in advance and removed from the rate control problem. We can tally up this fixed overhead in a similar way since we can compute it exactly, rather than relying on models using measurements of the pixel domain input frame which are in general not very reliable.
5.2 Rate Control for Single-layer MPEG-4 Encoders

To calculate the fixed overhead, those macroblocks with no non-zero DCT coefficients should be classified according to Table 5.1 which lists the number of bits to count towards each macroblock of a specific type. Note that no entries exist in the table for a zero motion vector and four motion vectors per block (a zero vector can always be expressed as one vector) nor for four motion vectors per macroblock and a non-zero differential quantizer step size since this is not supported by the MPEG-4 syntax.

The single macroblock skipped bit that is always present in every P-VOP and B-VOP macroblock is always considered to be a fixed overhead, even if there are non-zero DCT coefficients and as such the rest of the overhead cannot be exactly determined in advance.

Our final single-layer MPEG-4 bit production model can be written as:

\[ B^* = \alpha_1 \times c_d^* + \alpha_2 \times c_e^* + \alpha_3 \times m_d^* + \alpha_4 \times m_e^* + o_f + v \]  

(5.5)

where \( B^* \) is the estimated\(^1\) number of bits to be generated by the frame in question, \( c_d^* \) and \( c_e^* \) are the estimated number of intra and inter coefficients for the frame respectively while \( m_d^* \) and \( m_e^* \) are the estimated number of intra and inter coded macroblocks in the frame respectively. \( o_f \) is the total fixed macroblock overhead as discussed above and \( v = \sum_{i=1}^{N} v_i \) is the total number of bits required to code non-zero differential motion vectors. At the start of a sequence the \( \alpha \) parameters are initialized to \( \alpha_1 = 8, \alpha_2 = 7, \alpha_3 = 11 \) and \( \alpha_4 = 7 \). For the experiments to be conducted later, we have chosen to keep the intra parameters \( \alpha_1 \) and \( \alpha_3 \) constant since \( \alpha_1 \) is subject to volatile changes in macroblocks with few coefficients and \( \alpha_3 \) has been observed to not vary greatly between frames. The remaining parameters \( \alpha_2 \) and \( \alpha_4 \) have their values recomputed at the end of each frame using the statistics gathered in the second coding stage when the bitstream is generated.

\(^1\)All symbols used in this chapter marked with * are estimates.
5.2 Rate Control for Single-layer MPEG-4 Encoders

5.2.4 Perceptual Quantization

The use of constant quantization over a whole video frame is known to produce uneven quality when perceived by human viewers [68]. As such, we will want to choose quantizer step sizes that incorporate a perceptual component, i.e. in the $i$th macroblock

$$Q_i = \max(1, \min(31, Q_{frame} + \Delta_i + \delta_i))$$

where $Q_{frame}$ is constant throughout the frame and $\Delta_i$ is the perceptual offset value for macroblock $i$. For convenience, we define $\Delta_i \geq 0$ in all cases. Note that for very high bit rates or for a very large range of $\Delta_i$ values over a frame, $Q_{frame}$ may be negative. $\delta_i$ is an extra term that compensates for any observed deviations from the predicted number of bits generated through to macroblock $i$, whose calculation will be described in Section 5.2.7 below. The clipping operations in equation 5.6 ensure the quantizer step size is within the limits imposed by MPEG-4.

How the offsets $\Delta_i$ are chosen is not considered in detail here: we use the method specified by Pickering [68]. Other methods can be "plugged in" to the rate control algorithm to replace this if needed.

Two main perceptual effects are identified in [68]: spatial masking is the relative insensitivity of the human visual system to distortion in areas containing many luminance edges. Temporal masking is the insensitivity to distortion in areas of a sequence that are changing rapidly. A brief summary of how these factors are used in [68] to calculate $\Delta_i$ is given below.

To estimate the amount of quantization distortion, a weighted sum of the unquantized DCT coefficient values over each of the four luminance blocks in a macroblock is taken. A parameter $D_q$ is defined to be the maximum of these values (i.e. the worst case distortion) over the four luminance blocks. A weighting is used to de-emphasize high frequency contributions to $D_q$ where the human visual system is less sensitive to noise.
The amount of spatial masking is estimated from the inter-pixel correlation which is determined by calculating histograms of the difference between closely separated pixels (in the original frame before any prediction is done). If this correlation is high then this part of the frame is flat and distortion will be noticeable. A parameter $\delta_s$ is calculated from these histograms which are calculated for offsets in both the vertical and horizontal direction and various steps are taken to reduce the effect of noise, the details of which can be found in [68].

Temporal masking is estimated from the motion-compensated prediction difference by measuring the temporal correlation between coincident pixels in different frames by calculation of histograms in a similar manner to that used for estimating the amount of spatial masking. If the prediction difference has low spatial correlation then that part of the frame is changing quickly and distortion will not be as noticeable as in more static areas.

The final offset is calculated in terms of the quantization distortion, spatial and temporal masking parameters.

$$\Delta_i = -0.5\delta_s D_q + 12\delta_t\delta_s + 0.5D_q - 18\delta_s - 12\delta_t$$  \hspace{1cm} (5.7)

which is then normalized so the minimum value of $\Delta_i$ over the whole frame is zero in line with its earlier definition.

These raw $\Delta_i$ values may need to be modified to ensure they are used effectively in the context of the MPEG-4 syntax. Firstly, if a macroblock has four motion vectors then the MPEG-4 standard does not allow for the quantizer step size to change relative to the previous macroblock. This can cause problems when high priority macroblocks (i.e. those that are subjectively important and are associated with low values of $\Delta_i$) which require four motion vectors but whose effective value of $\Delta_i$ is that of the last coded macroblock with a single vector. To correct for this, if a macroblock with a single vector is followed by one or more macroblocks with four vectors thus forming a group of macroblocks which share the same quantizer step size, its priority is raised to that of the
highest in the group to ensure that the most important macroblock is coded accurately. Secondly, due to the restriction in the MPEG-4 syntax that the macroblock quantizer can change by at most ±2 relative to the last macroblock not all choices for \( \Delta_i \) will be "reachable". After the initial computation of the values for \( \Delta_i \) using equation 5.7, a number of passes are made through the set of \( \Delta_i \) values starting with those of the highest priority and any of those macroblocks that cannot be reached given the choices already made for macroblocks with higher priority have their priority raised (\( \Delta_i \) lowered) until the all macroblocks can be reached with a maximum quantizer step size difference of ±2 per (single motion vector) macroblock.

5.2.5 Thresholding

While it is possible to compute the number of non-zero coefficients (\( c_{a}^* \) and \( c_{e}^* \) in equation 5.5) by thresholding for every possible value of the quantizer step size \( Q_i \in [1, 31] \), this is computationally expensive. To reduce the number of comparisons required, a subset of eight quantizer step size values \{31, 25, 19, 15, 12, 9, 6, 4\} is used where counts of the number of non-zero coefficients are taken, and the counts at other values of the quantizer step size are estimated using spline interpolation [69]. As we are not using a uniform value for the quantizer step size over the whole frame, it is necessary to keep separate counts for discrete values of \( \Delta_i \). Many modern CPU architectures support parallel comparison between multiple pairs of values in a single machine instruction; notably the Intel Pentium III (and later) processors [70] support the comparison between four pairs of 16-bit signed integers (each suitable for storage of an unquantized DCT coefficient value that requires at least 12 bits) packed into a 64-bit MMX register. The choice of eight thresholds was made so that the thresholding of each coefficient requires two machine instructions. It was found during development that using only four thresholds was prone to significant errors in the interpolated values, especially where the number of coefficients is close to zero.
for a particular value of $\Delta_t$.

As well as estimating the number of DCT coefficients to be generated, the thresholding process is also used to estimate the number of coded blocks and macroblocks in the frame ($m_a^*$ and $m_e^*$ in equation 5.5). The number of coded blocks is not required for the single-layer rate model but will be used later with stream morphing. If we are certain that there is at least one unquantized coefficient whose value is above the threshold then we know that the block it is in will be coded and also that the corresponding macroblock will be coded. As for the DCT coefficient case, the number of coded blocks and macroblocks is interpolated for quantizer step size values not in the set used for thresholding.

### 5.2.6 Frame-Level Rate Control

The target number of bits for the current frame $B_t$ is calculated not only from the average number of bits per frame at the nominal bit rate but also requires a term for the control of the fullness of the decoder buffer. To minimize the possibility of buffer underflow, it is desirable to keep the buffer partly full at all times. We have chosen a target buffer fullness of $\frac{1}{3}$ the maximum buffer size which leads to the following expression for $B_t$:

$$B_t = \begin{cases} \frac{R}{F} + \frac{1}{6} \left( \frac{D_{\text{max}}}{3} - D \right) & \text{for } D < \frac{D_{\text{max}}}{3} \\ \frac{R}{F} - \frac{1}{4} \left( D - \frac{D_{\text{max}}}{3} \right) & \text{otherwise} \end{cases}$$

(5.8)

where $R$ is the target bit rate, $F$ is the frame rate, $D$ is the current fullness of the decoder buffer and $D_{\text{max}}$ is the overall size of the decoder buffer, i.e. $0 \leq D \leq D_{\text{max}}$.

Buffer overflow is a rather more serious condition than underflow and if the decoder buffer was to fill completely the coding of the frame would have to stop at that point since no more bits can be added. Underflow can be compensated for by adding extra stuffing bits at the end of the coding of the current frame (which have no purpose other than to consume the extra bits). Equation 5.8
therefore attempts to empty the buffer more quickly if it is more than $\frac{1}{3}$ full and is thus closer to the overflow condition.

In cases where the number of non-zero motion vector bits $v$ (Section 5.2.3) is close to, or indeed higher than $B_t$ one additional measure that is taken to prevent the current frame from overshooting the target number of bits is to convert all macroblocks with four motion vectors to use only a single vector. This reduces the quality of the prediction but in these cases preventing a significant excess of bits (and possibly buffer overflow) is the more important goal.

5.2.7 Monitoring of the Coding Process

After the target number of bits for the frame $B_t$ has been calculated using equation 5.5, these bits are divided up over the frame so the progress of the encoding of the frame and the number of bits generated can be compared to a prediction from the rate model. TM5 rate control works on the premise that the bits are distributed evenly over the frame which is usually a poor assumption, especially at low bit rates where the bits may be spent in only a few macroblocks with most of the macroblocks requiring only one bit (for the macroblock skipped flag).

The estimated number of bits for macroblock $i$, $B_i^*$ is calculated as follows. If $Q_i$ is one of the eight quantizer step size values (index $t$ where $t \in [0, 7]$) where thresholding was carried out then:

$$B_i^* = \begin{cases} 
\alpha_1 \times \frac{C_{i,t}}{T_{\Delta_i,t,a}} \times c_i^* + \alpha_3 & \text{intra} \\
\alpha_2 \times \frac{C_{i,t}}{T_{\Delta_i,t,e}} \times c_i^* + \alpha_4 + v_i & \text{inter with } C_{i,t} > 0 \\
v_i & \text{otherwise}
\end{cases} \tag{5.9}$$

where $C_{i,t}$ is the number of non-zero coefficients in macroblock $i$ for threshold $t$, $T_{\Delta_i,t,a}, T_{\Delta_i,t,e}$ are the total number of non-zero intra and inter coefficients respectively for all macroblocks with the same value of $\Delta_i$ as macroblock $i$ for threshold $t$, i.e. $T_{\Delta_i,t,a} + T_{\Delta_i,t,e} = \sum_{j: \Delta_j = \Delta_i} C_{j,t}$ where the sum is over all macroblocks sharing the same offset, and $v_i$ is the number of bits for any non-zero motion vector in the macroblock ($v_i = 0$ if the motion vector is zero). The
condition in the second case of equation 5.9 means that the macroblock overhead term is only added for macroblocks with non-zero coefficients which is something of an approximation since headers are required if the differential quantizer step size is non-zero or \( v_i \neq 0 \). Note that this is already accounted for since we are modelling separately (Section 5.2.3, the term \( \sigma_f \) in equation 5.5) the fixed overhead associated with macroblocks with no coefficients. If \( Q_i \) does not correspond to a threshold quantizer and as such \( T_{\Delta_i,t,a} \) or \( T_{\Delta_i,t,e} \) were interpolated then the estimate must also interpolate between sample counts for thresholds \( t \) and \( t + 1 \):

\[
B_i^* = \begin{cases} 
\alpha_1 \times \frac{C_{i,t} \times T_{\Delta_i,t+1,a} + C_{i,t+1} \times T_{\Delta_i,t,a}}{T_{\Delta_i,t,a} T_{\Delta_i,t+1,a}} \times c_d^* + \alpha_3 & \text{intra} \\
\alpha_2 \times \frac{C_{i,t} \times T_{\Delta_i,t+1,e} + C_{i,t+1} \times T_{\Delta_i,t,e}}{T_{\Delta_i,t,e} T_{\Delta_i,t+1,e}} \times c_e^* + \alpha_4 + v_i & \text{inter with } C_{i,t+1} > 0 \\
v_i & \text{otherwise}
\end{cases} \tag{5.10}
\]

Note that \( Q_{i,t+1} > Q_i \) since the number of non-zero coefficients in a macroblock can never fall with the choice of a smaller value for the quantizer step size.

After the encoding process has been completed, the rate controller calculates the difference \( \Delta_{b,i} \) between the number of bits that have been generated and how many were predicted by the rate control algorithm at macroblock \( i \):

\[
\Delta_{b,i} = \left( \sum_{i' = 0}^{i} B_{i'}^* \right) - (B_{\text{total}} - B_{\text{start}}) \tag{5.11}
\]

where \( B_{\text{total}} \) is the total number of bits for the whole bitstream so far and \( B_{\text{start}} \) was the bit count at the start of the current frame. \( \epsilon_i \) is then defined as the ratio of \( \Delta_{b,i} \) to the target number of bits for the frame, i.e. \( \epsilon_i = \frac{\Delta_{b,i}}{B_i} \).

A proportional-plus-integral (PI) controller is used with \( \epsilon_i \) as its input signal to correct for any deviation from the expected number of bits which calculates a correction term \( \delta_i \) which is then added to the macroblock quantizer step size (equation 5.6):

\[
\delta_i = \beta_1 \times \epsilon_i + \beta_2 \times \sum_{i' = 1}^{i-1} \epsilon_{i'} \tag{5.12}
\]
5.2 Rate Control for Single-layer MPEG-4 Encoders

The summation (integral) term is required because in general it is not possible to choose a value for $Q_{\text{frame}}$ that results in $B^*$ being sufficiently close to the desired target $B_t$ since $Q_{\text{frame}}$ can only be chosen from a relatively small set of integer values which result in widely spaced values for $B^*$. To avoid this steady-state error (ignoring any other disturbances due to inaccuracies the rate model) at least one integrator is required in the control system [71].

For our experiments, the parameter values $\beta_1 = 30$ and $\beta_2 = 300$ were used. Higher parameter values result in more accurate tracking of the number of target bits however if the correction signal $\delta_i$ is changed too frequently the overhead associated with these changes will become too great.

5.2.8 Intra-Coded Frames

While the rate control algorithm that has been described models the use of intra-coded macroblocks, it is not designed to work on the first frame of the sequence that must be coded entirely in intra mode. The first frame normally requires more bits than will fit into the decoder buffer and needs to be transmitted over more than one frame period. Similarly, the first P frame after a scene change will be effectively an I frame as all macroblocks should be forced to be coded in intra mode due to large differences in the motion-compensated prediction. Here also extra time needs to be taken to code the frame, possibly by dropping the following frame and allocating two times the normal frame bit allocation for the coding of the frame in intra mode.

5.2.9 Algorithm

The complete rate control algorithm is as follows:

1. Initialize model parameters: $\alpha_1 = 8$, $\alpha_2 = 7$, $\alpha_3 = 11$, $\alpha_4 = 7$.

2. Calculate the target number of bits $B_t$ for the frame based on the desired rate and buffer fullness using equation 5.8.
3. Perform motion estimation and tally the number of bits required to code differential motion vectors in macroblocks with non-zero full motion vectors. If there are too many motion vector bits, i.e. \( \nu > c \times B_t \) where \( c \) is some threshold \( (c = \frac{2}{3} \) is used for our experiments) then collapse the motion vectors in macroblocks that use four vectors per macroblock into a single vector per macroblock.

4. Perform motion-compensated prediction then apply the DCT to the prediction difference.

5. Analyse each macroblock in the original frame and corresponding prediction difference macroblock to calculate perceptual offsets \( \Delta_i \), applying any corrections required to conform with the MPEG-4 syntax as discussed in Section 5.2.4.

6. Applying thresholding (Section 5.2.5) to the unquantized DCT coefficients from each macroblock and update total counts of non-zero coefficients, coded blocks etc. which are kept for each discrete value of \( \Delta_i \) used in the frame.

7. Place the unquantized coefficients into a separate frame store for use in the second stage.

8. Apply the bit production model (equation 5.5) to the statistics gathered and find the value of \( Q_{frame} \) which predicts the largest value of \( B^* \) that does not exceed \( B_t \).

9. Calculate \( B^*_i \) for all macroblocks using equation 5.9 or 5.10 as appropriate.

10. Take the unquantized coefficients from their frame store, quantize and encode them using macroblock quantizer \( Q_i = Q_{frame} + \Delta_i + \delta_i \). Monitor the number of bits generated and number of coefficients actually coded and adjust \( \delta_i \) as per Section 5.2.7. While more macroblocks require second-stage processing repeat this step.
5.3 Rate Control for MPEG-2-Compliant SNR Scalability

11. Update model parameters $\alpha_1$-$\alpha_4$ if desired from data collected during previous step:

\[
\begin{align*}
\alpha_1 &= \frac{N_{c,a}}{c_a} \\
\alpha_2 &= \frac{N_{c,e}}{c_e} \\
\alpha_3 &= \frac{N_{m,a}}{m_a} \\
\alpha_4 &= \frac{N_{m,e}}{m_e}
\end{align*}
\]

where $N_{c,a}$ and $N_{c,e}$ are the number of bits used to code intra and inter DCT coefficients respectively, $N_{m,a}$ and $N_{m,e}$ are the number of bits used to code intra and inter macroblock overhead, $c_a$ and $c_e$ are the number of intra and inter DCT coefficients and $m_a$ and $m_e$ are the number of coded intra and inter macroblocks. Recalculation of $\alpha_1$ and $\alpha_3$ may be skipped since they normally contribute only a limited number of bits and $\alpha_1$ tends to be volatile. If $c_a = 0$ then $\alpha_1$ cannot be recalculated in this frame and its value remains unchanged. This is also the case for $\alpha_3$ if $m_a = 0$.

12. While more frames are to be coded return to step 2.

Later extensions to this algorithm to support scalable encoders only modify the steps relating to the bit production model (steps 1, 8 and 11). All other steps remain the same.

5.3 Rate Control for MPEG-2-Compliant SNR Scalability

MPEG-2-compliant SNR scalability uses an almost identical syntax to the base layer case, the only difference being that no motion vectors need to be coded in the enhancement layers, as it is a requirement that the motion vectors are
5.3 Rate Control for MPEG-2-Compliant SNR Scalability

the same in all layers for use with a single-loop decoder. As such the bit production model described in the previous section can be used, the only change being the $v_i$ terms in equations 5.9 and 5.10 are zero for all enhancement layer macroblocks.

For an $n$-layer scalable encoder ($n > 1$), the rate control algorithm as just described is run independently in each of the layers. As in the stream morphing case, the differential quantizer step size for each macroblock is predicted from the layer below and to minimize the amount of overhead associated with changing the quantizer step size it is desirable for the step size in consecutive layers to be separated by an offset, which is constant over as many macroblocks as possible (Section 4.3.1 and Figure 4.6). Using the notation of equation 5.6, we want to use the same values for $\Delta_i$ in each layer and to minimize the number of times the value of $\delta_i$ changes while still meeting bit rate targets.

Ideally a set of $\Delta_i$ values would be calculated at the top layer which is where we normally want to optimize the quality of the coded sequence. These $\Delta_i$ values would be then passed to the rate controllers running in the lower layers. In the MPEG-2-compliant SNR scalable case, this is not possible because the calculation of $\Delta_i$ in the top layer requires the unquantized DCT coefficients to be available in that layer. These coefficients are predicted from the previous layer in the pyramid configuration (Figure 3.4) which can only be known after the rate control decision in that layer is made, which is impossible if we are trying to use the same values for $\Delta_i$ in all layers since the lower layers are waiting for $\Delta_i$ to be calculated in the top layer. Therefore in the MPEG-2-compliant SNR scalable case the set of $\Delta_i$ values must be calculated at the base layer and passed upwards to the rate controllers in the higher layers.

The fact that the proposed rate control algorithm requires all unquantized DCT coefficients to be available in a given layer which itself depends on the rate control decisions in the previous layer(s) has the effect of significantly increasing the encoder latency. The first pre-analysis stage (thresholding etc.) in layer $n + 1$
can only be conducted during the second stage (quantization, coding, IDCT etc.) of coding layer $n$ as the quantized DCT coefficients become available. As more layers are added, the coding of those layers is further delayed (in comparison to the base layer) while waiting for the lower layers to complete. The length of this delay depends on the implementation. However, the most significant factor affecting the total delay is the motion estimation process which is only done once so the overall latency for an $n$-layer encoder will be $\ll n$ frames.

### 5.4 Rate Control for Stream Morphing

The rate control problem for multi-layer stream morphing encoders is significantly different from MPEG-2-compliant SNR scalability as discussed in the previous section. The main differences are:

- Coding of macroblock overhead (primarily coded block patterns, macroblock skipped and differential quantizer step sizes) is done in a quite different way in stream morphing. Blocks that were coded in the previous layer are automatically scanned for coefficient value changes and require no coded block pattern signalling. Unlike the standard MPEG-4 syntax, the differential quantizer step size can be non-zero without inducing significant extra overhead in otherwise empty macroblocks as shown in Table 5.1.

- The cost associated with coding individual non-zero coefficients depends on whether a coincident non-zero coefficient existed in a previous layer. The cost of coding a new coefficient that does not appear in any previous layer is similar to the single-layer and MPEG-2-compliant SNR scalability cases previously discussed whereas if the coefficient is simply an update of a previous coefficient then it requires significantly fewer bits on average.

- As the prediction mode can change between layers in stream morphing,
those macroblocks that are passed through the arithmetic coder in their original syntax (Section 4.3.2) must be considered.

5.4.1 Bit Production Model

In this section we develop a new bit production model that can be inserted into the rate control framework developed in Section 5.2.9 to replace the original single-layer model of Section 5.2.3. The overall number of bits required to code a frame of video in a stream morphing enhancement layer is divided up into its logical components in line with the list above and the original stream morphing bitstream description given in Section 4.3. These are:

- New coefficients not present in any previous layer.
- Existing coefficients that are being updated for the current layer.
- Macroblock skipped and coded block pattern overhead to signal which of the macroblocks or blocks that were skipped in the previous layer must now be scanned for non-zero coefficients.
- Bits required to pass through macroblocks where the prediction mode has changed.

Models for each of these components are developed in the following sections. New and existing coefficients require different models for intra- and inter-coded coefficients.

It is not possible to model the number of bits required to code the differential quantizer step size symbols since this is primarily a function of how poor the rate model is and as such how many times the correction factor $\delta_i$ in equation 5.6 changes which is clearly not known in advance. Ideally the contribution from these symbols is negligible as the differential quantizer step size is almost always predicted correctly from the previous layer.
5.4 Rate Control for Stream Morphing

(a) Average number of bits required for each new coefficient in inter macroblocks

(b) Bits required for new coefficients in intra macroblocks

Figure 5.12: Stream morphing: number of bits required to code new coefficients
("Foreman", 1st enhancement layer of Figure 4.16)
5.4 Rate Control for Stream Morphing

5.4.1.1 New Coefficients

As originally described in Section 4.3.6, new coefficients that are zero-valued in all previous layers are scanned in a similar way to coefficients in the single-layer case. As such, the single-layer model with a linear relationship between number of coefficients and the number of bits required to code those coefficients (Section 5.2.3 and Figures 5.7 and 5.8) is valid for these classes of coefficients in stream morphing. The statistics plotted in Figure 5.12 (taken from the first enhancement layer of the coding of "Foreman" in Figure 4.16) show this to be the case. As in previous cases the intra macroblock data in Figure 5.12 is shown as a scatter plot rather than a continuous line to avoid discontinuities at frames that have no intra macroblocks.

The thresholding operation that is done in the first stage of the encoder produces statistics on the estimated number of coefficients in each layer for the whole frame. It is not possible, however, to split up these counts into new and existing coefficients since whether a coefficient is coded in a previous layer is dependent on the choices for quantizer step size in those layers which have not been made yet as well as any manipulation of the quantized coefficients to prevent equal and opposite changes in successive layers as described in Section 4.3.6.4 (which depends not only on previous layers but also higher layers as well). The number of new intra and inter coefficients in layer \( n \), \( c_{\text{new},a,n} \) and \( c_{\text{new},e,n} \) are estimated using the following expressions:

\[
\begin{align*}
    c_{\text{new},a,n} &= c^*_{a,n} - c^*_{\text{thru},a,n} - (c^*_{a,n-1} - c^*_{\text{thruprev},a,n}) \\
    c_{\text{new},e,n} &= c^*_{e,n} - c^*_{\text{thru},e,n} - (c^*_{e,n-1} - c^*_{\text{thruprev},a,n})
\end{align*}
\]

where \( c^*_{a,n}, c^*_{e,n} \) are the overall intra and inter coefficient counts for the whole frame, \( c^*_{\text{thru},a,n}, c^*_{\text{thru},e,n} \) are the number of intra and inter coefficients in layer \( n \) that are in macroblocks that will be passed through due to prediction mode change and \( c^*_{\text{thruprev},a,n}, c^*_{\text{thruprev},e,n} \) are the number of intra and inter coefficients in the previous layer in macroblocks that are coincident with those in
layer $n$ that are to be passed through. Recall that the statistics used here are dependent on the choice of $Q_{\text{frame}}$ in the current layer and the set of $\Delta_i$ values.

The results from applying equations 5.17 and 5.18 are, by necessity, estimates only. They assume that all non-zero coefficients in the layer $n - 1$ will remain non-zero in layer $n$ which is not always true. In Figure 5.12(b) the number of new intra coefficients estimated by application of equation 5.17 is occasionally a negative number which is clearly impossible and this is an indication that the coefficient count generated by equation 5.17 is only approximate. Fortunately in these cases there are very few new intra coefficients (if we could obtain a true count at this time) and so clipping this count to zero does not result in a significant error.

It is theoretically possible to obtain accurate non-zero coefficient statistics by coding the layers one at a time as is done in the MPEG-2-compliant SNR scalable case at the expense of extra encoder latency as described in Section 5.3. If the collection of statistics for layer $n$ does not start until the coding of layer $n - 1$ is complete then accurate counts can be obtained and the locations of all coded coefficients in layer $n - 1$ could be used to determine the split between new and existing coefficients. The fact that the final texture in layer $n$ of a stream morphing encoder is not independent and must be coded with respect to the previous layer means that the bitstream for layer $n - 1$ would need to be stored after it was coded to be used again during the coding of layer $n$ which would add significantly to the complexity and memory requirements of the encoder. Estimating the number of new and existing coefficients allows for all coincident macroblocks to be processed simultaneously which reduces complexity and latency at the expense of some decrease in the accuracy of the bit production model.
5.4 Rate Control for Stream Morphing

Stream Morphing: Existing INTER Coefficients

(a) Average number of bits per coefficient for updates in inter macroblocks

Stream Morphing: existing INTRA coefficients

(b) Number of bits required to update coefficients in intra macroblocks

Figure 5.13: Stream morphing: number of bits required to code existing coefficients ("Foreman", 1st enhancement layer of Figure 4.16)
5.4.1.2 Existing Coefficients

In a manner similar to the case of new coefficients as described in the last section, the number of existing coefficients $c_{\text{exist},a,n}^*$, $c_{\text{exist},e,n}^*$ must be estimated using the following expressions:

\begin{align}
  c_{\text{exist},a,n}^* &= c_{a,n-1}^* - c_{\text{pthruprev},e,n}^* \\
  c_{\text{exist},e,n}^* &= c_{e,n-1}^* - c_{\text{pthruprev},a,n}^*
\end{align}

(5.19)  
(5.20)

Figure 5.13 shows plots of the number of bits per existing coefficient using the same linear model as used previously. Volatility in the estimates $c_{\text{exist},a,n}^*$ and $c_{\text{exist},e,n}^*$ affect the rate model to a greater extent here although note that updates of existing coefficients contribute considerably fewer bits to the overall bitstream than new coefficients (see Tables 4.12 and 4.13). Similar to the case of new coefficients described in the previous section, Figure 5.13 indicates that first-order models can approximate well the number of bits required for existing coefficients in intra- and inter-coded macroblocks.

5.4.1.3 Coded Block Patterns and Macroblock Skipped

The overhead required for signalling which of the blocks and macroblocks that were skipped in the base layer are coded in the current layer was (empirically) found to be approximately proportional to the square root of the difference between the number of coded blocks in the current and previous layers $\Delta_{b,n}^*$:

\[
\Delta_{b,n}^* = b_{a,n}^* + b_{e,n}^* - (b_{\text{pthru},a,n}^* + b_{\text{pthru},e,n}^*) - (b_{a,n-1}^* + b_{e,n-1}^*)
\]

(5.21)

where $b_{a,n}^*$, $b_{e,n}^*$ are the number of coded intra and inter macroblocks estimated by the thresholding process in layer $n$ and $b_{\text{pthru},a,n}^*$, $b_{\text{pthru},e,n}^*$ are the numbers of those blocks which are to be passed through in the current layer.

Sections 4.3.3 and 4.3.5 described how the number of surrounding blocks or macroblocks are used to alter the statistics of the symbols in the arithmetic coder to favour those new blocks or macroblocks that appear near existing ones.
Figure 5.14: Stream morphing: coded block pattern overhead ("Foreman", 1st enhancement layer of Figure 4.16)
As such, the subset of new blocks or macroblocks that appear out on their own or are adjacent to existing texture but are not in the causal set of adjacents used for probability model selection (Figure 4.8) will require a higher number of bits to code than those which are in areas where we might "expect" new blocks to be found. As such, the number of bits required to code these does not rise linearly with the number of new blocks but rather more slowly.

Figure 5.14 shows a plot of the ratio of the number of bits required to code this information to $\sqrt{\Delta b_{t,n}}$. Again, the value of this quantity in the previous frame is a good predictor of its value in the current frame so a linear relationship can be used.
5.4.1.4 VLCs for Pass Through Macroblocks

For macroblocks using a different prediction mode to the coincident macroblock in the previous layer, the VLCs from the corresponding single-layer bitstream are passed through the arithmetic coder using a uniform binary model. As the same number of bits is required for coding these macroblocks as was the case when using standard VLCs, the same bit production model can be used for those macroblocks as was used in the single-layer case (Section 5.2.3). The number of bits required for all macroblocks that are passed through in layer \( n \) is therefore approximated by:

\[
B_{\text{passthru},n}^* = \sum_{i: \text{passthru}} \left( 4.5 \times c_{\text{passthru},a,i,n}^* + 7.2 \times c_{\text{passthru},e,i,n}^* + 10 + v_i \right)
\]  

(5.22)

where \( c_{\text{passthru},a,i,n}^* \) and \( c_{\text{passthru},e,i,n}^* \) are the number of intra and inter non-zero coefficients estimated to be present in macroblock \( i \) of layer \( n \). The addition of ten bits per macroblock is to account for the predicted amount of macroblock overhead. Given the small number of macroblocks usually involved, the constants in the expression are fixed rather than being updated at the end of each frame. Figure 5.15 shows a plot of the number of bits predicted by this model and the number actually used for the example stream morphing enhancement layer which shows this model works well.

5.4.1.5 Complete Model

Combining the four components outlined in the previous sections gives the following expression for the predicted number of bits for a given frame in layer \( n \):

\[
B_n^* = \alpha_1 \times c_{\text{new},a,n}^* + \alpha_2 \times c_{\text{new},e,n}^* + \alpha_3 \times c_{\text{exist},a,n}^* + \alpha_4 \times c_{\text{exist},e,n}^* + \alpha_5 \times \sqrt{\Delta_{b,n}^*} + \sum_{i: \text{passthru}} \left( 4.5 \times c_{\text{passthru},a,i,n}^* + 7.2 \times c_{\text{passthru},e,i,n}^* + \right)
\]
This can be used as a substitute for the single-layer model of equation 5.5 in step 8 of the rate control algorithm outlined in Section 5.2.9.

At the start of the simulation (step 1 in Section 5.2.9) the rate control parameters are initialized to $\alpha_1 = 7.0$, $\alpha_2 = 7.5$, $\alpha_3 = 0.5$, $\alpha_4 = 2.25$ and $\alpha_5 = 50.0$ (these values were determined from empirical data gathered from a number of tests on different video sequences). At the end of each frame these parameters are re-calculated (step 11 in Section 5.2.9) using statistics gathered during the encoding process:

\[
\begin{align*}
\alpha_1 &= \frac{B_{\text{new},a,n}}{C_{\text{new},a,n}} \\
\alpha_2 &= \frac{B_{\text{new},e,n}}{C_{\text{new},e,n}} \\
\alpha_3 &= \frac{B_{\text{exist},a,n}}{C_{\text{exist},a,n}} \\
\alpha_4 &= \frac{B_{\text{exist},e,n}}{C_{\text{exist},e,n}} \\
\alpha_5 &= \frac{B_{\text{cbp},n}}{\sqrt{\Delta b,n}}
\end{align*}
\]

where $B_{\text{new},a,n}$, $B_{\text{new},e,n}$, $B_{\text{exist},a,n}$, $B_{\text{exist},e,n}$, $B_{\text{cbp},n}$ are the number of bits that were used to code new intra coefficients, new inter coefficients, existing intra coefficients, existing inter coefficients and coded block patterns/macroblock skipped respectively.

## 5.5 Experimental Procedure

Four codecs were constructed to evaluate the performance of the new rate control architecture: two for the single-layer case described in Section 5.2, one with the MPEG-4 FGS enhancement layer and one without, a third MPEG-2-compliant SNR scalable system with rate control as described in Section 5.3 and a fourth for the stream morphing case of Section 5.4. The experimental conditions are
the same as for the previous chapter as detailed in Section 4.6 with low bit rates being selected for the base layer and in the case of those codecs with discrete enhancement layers four enhancement layers are used which are spread over approximately twice the bit rate used by the base layer. The maximum buffer size in each layer was selected to produce a maximum delay of $\frac{1}{4}$s. Constant quantization is used in the first frame and has been adjusted for "low-motion" sequences to produce uniform quality over the sequence as was done in the experiments of the previous two chapters.

In addition, some subjective tests were conducted to compare the performance of stream morphing to the MPEG-2-compliant SNR scalability and MPEG-4 FGS for which preliminary results were presented in Chapter 3. The test sequences and rates used are the same as those for the Chapter 3 tests listed in Table 3.3. The stream morphing and MPEG-2-compliant SNR scalability tests both use four enhancement layers spread over the specified total enhancement layer rate. For the low motion sequences "Akiyo" and "Mother & Daughter" some additional comparisons are made with MPEG-2-compliant SNR scalability and FGS whose total enhancement layer rates have been adjusted to give top layer PSNR values approximately equal to stream morphing (whose total rate is unmodified). This "normalized" or "same PSNR" case is included to attempt to measure how much (if any) of the difference in subjective quality is intrinsic to the type of scalability used and how much is due to other factors such as syntactic efficiency, especially for macroblock overhead which uses arithmetic coding in stream morphing but less efficient (and less computationally-expensive) VLC coding in the single-layer and MPEG-2 SNR scalable cases. This is important as it changes the number of bits available for coding DCT coefficients in a CBR environment. If the subjective qualities of two techniques are substantially different at the same PSNR, then this is an indication that other factors are affecting performance. For MPEG-2-compliant SNR scalability, the fact that coefficients from lower layers are not updated in higher layers due to the need to produce
non-zero coefficients in those layers as previously discussed (Section 3.7.3.2) may produce a noticeable effect on the subjective quality that is not reflected in PSNR results. Similarly for MPEG-4 FGS it has been noted that in very slow moving areas (especially at edges of objects) combined with low base layer bit rates (i.e. poor quality base layer) a "shimmering" effect can be seen where the FGS residue signal is re-computed in each frame and can appear as time-varying noise. As both of these effects are caused by the addition of enhancement layer data to the existing base layer, these extra experiments seek to highlight any subjective advantages that stream morphing has given it creates a single-layer bitstream in all layers.

5.6 Results

Figures 5.16 and 5.18 show results for CIF 10fps using the sequences "Foreman" and "Mother & Daughter" respectively with base layer bit rates of 130kbps and 56kbps respectively. To reduce clutter in the plots in the case of "Foreman" where the sequences are more closely spaced in terms of PSNR only the first and fourth (topmost) enhancement layers are shown. Figure 5.17 shows the number of bits used per frame for the stream morphing and MPEG-2-compliant SNR scalable results of Figure 5.16. Figures 5.19 and 5.20 show results for the same sequences at 30fps CIF with base layer bit rates of 220kbps and 80kbps respectively. Figures 5.21 and 5.22 show results for 30fps QCIF with base layer rates of 70kbps and 18kbps respectively.

Figure 5.23 displays the subjective test results for the "average" observer with the reference condition being stream morphing. Negative values denote subjective quality that is worse than the reference condition. Table 5.2 shows the spread of subjective test results for all observers and Table 5.3 shows the probability that the "average" observer will not find stream morphing to be subjectively superior to the other specified coding technique. Other details
5.6 Results

Rate Control results for Foreman (CIF 10fps)

(a) Comparative performance

(b) Video quality in enhancement layers

Figure 5.16: Rate control results for “Foreman” (CIF 10fps)
Figure 5.17: Bits used (cumulative) for discrete scalable encoders ("Foreman", CIF 10fps)
Results

Rate Control results for Mother & Daughter (GIF 10fps)

37.5
36.5
35.5
34.5
33.5
80 100 120 140 180
Bit rate (kbps) [excluding first frame]

(a) Comparative performance

Enhancement Layer Frame PSNRs for Mother & Daughter (Rate Control — CIF 10fps)

(b) Video quality in enhancement layers

Figure 5.18: Rate control results for “Mother & Daughter” (CIF 10fps)
5.6 Results

Rate Control results for Foreman (CIF 30fps)

(a) Comparative performance

(b) Video quality in enhancement layers

Figure 5.19: Rate control results for “Foreman” (CIF 30fps)
5.6 Results

Rate Control results for Mother & Daughter (GIF 30fps)

(a) Comparative performance

(b) Video quality in enhancement layers

Figure 5.20: Rate control results for “Mother & Daughter” (CIF 30fps)
5.6 Results

Rate Control results for Foreman (QCIF 30fps)

32.5
32
31.5
31
CD
DC 30.5
CO
30
29.5
29
28.5
-e- single Layer
-X- Stream Morphing
*- MPEG-2-Compliant SNR Scalability
MPEG-4 FGS

Bit rate (kbps) [excluding first frame]

Frame PSNRs for Foreman (QCIF 30fps)

35
34
33
32
31
30
29
28
27
0 50 100 150 200 250 300
Frame Number

(a) Comparative performance

(b) Video quality in enhancement layers

Figure 5.21: Rate control results for “Foreman” (QCIF 30fps)
5.6 Results

Rate Control results for Mother & Daughter (QCIF 30fps)

(a) Comparative performance

(b) Video quality in enhancement layers

Figure 5.22: Rate control results for “Mother & Daughter” (QCIF 30fps)
5.6 Results

Subjective Performance of Stream Morphing (Akiyo CIF 10fps)
Base layer rate = 32kbps

Subjective Performance of Stream Morphing (Mother & Daughter CIF 10fps)
Base layer rate = 56kbps

(a) Akiyo

(b) Mother & Daughter

Figure 5.23: Subjective comparisons with stream morphing (95% confidence intervals)
5.6 Results

Subjective Performance of Stream Morphing (Foreman CIF 10fps)
Base layer rate = 130kbps, total enh. rate = 260kbps

Subjective Performance of Stream Morphing (Carphone CIF 10fps)
Base layer rate = 100kbps, total enh. rate = 200kbps

(c) Foreman

(d) Carphone

Figure 5.23 (continued)
### 5.6 Results

<table>
<thead>
<tr>
<th>Technique</th>
<th>Stream morphing better</th>
<th>Indistinguishable</th>
<th>Other technique better</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Akiyo</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MPEG-2 SNR</td>
<td>94.7%</td>
<td>5.3%</td>
<td>0%</td>
</tr>
<tr>
<td>MPEG-4 FGS</td>
<td>100%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>MPEG-2 SNR (same PSNR)</td>
<td>68.4%</td>
<td>10.5%</td>
<td>21.1%</td>
</tr>
<tr>
<td>MPEG-4 FGS (same PSNR)</td>
<td>36.8%</td>
<td>21.1%</td>
<td>42.1%</td>
</tr>
<tr>
<td><strong>Mother &amp; Daughter</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MPEG-2 SNR</td>
<td>89.5%</td>
<td>0%</td>
<td>10.5%</td>
</tr>
<tr>
<td>MPEG-4 FGS</td>
<td>100%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>MPEG-2 SNR (same PSNR)</td>
<td>89.5%</td>
<td>5.3%</td>
<td>5.3%</td>
</tr>
<tr>
<td>MPEG-4 FGS (same PSNR)</td>
<td>31.6%</td>
<td>15.8%</td>
<td>52.6%</td>
</tr>
<tr>
<td><strong>Foreman</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MPEG-2 SNR</td>
<td>47.4%</td>
<td>26.3%</td>
<td>26.3%</td>
</tr>
<tr>
<td>MPEG-4 FGS</td>
<td>68.4%</td>
<td>15.8%</td>
<td>15.8%</td>
</tr>
<tr>
<td><strong>Carphone</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MPEG-2 SNR</td>
<td>63.2%</td>
<td>21.1%</td>
<td>15.8%</td>
</tr>
<tr>
<td>MPEG-4 FGS</td>
<td>84.2%</td>
<td>10.5%</td>
<td>5.3%</td>
</tr>
</tbody>
</table>

Table 5.2: Spread of subjective test results

<table>
<thead>
<tr>
<th></th>
<th>Akiyo</th>
<th>Mother &amp; Daughter</th>
<th>Foreman</th>
<th>Carphone</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPEG-2 SNR</td>
<td>0.001%</td>
<td>0.15%</td>
<td>11.59%</td>
<td>0.52%</td>
</tr>
<tr>
<td>MPEG-4 FGS</td>
<td>0.0001%</td>
<td>0.0002%</td>
<td>0.37%</td>
<td>0.03%</td>
</tr>
<tr>
<td>MPEG-2 SNR (same PSNR)</td>
<td>2.6%</td>
<td>0.09%</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>MPEG-4 FGS (same PSNR)</td>
<td>25.4%</td>
<td>89.23%</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 5.3: Probability of average user finding stream morphing (reference condition) worse than other tested technique.
such as the bit rates for the “same PSNR” tests and the PSNR values at the top layer are listed in Appendix C.

5.7 Discussion

The rate-distortion performance of the scalable codecs under CBR conditions is very similar to the constant quantizer case from the previous chapter. For low bit rates (especially the 30fps results) “Mother & Daughter” has significantly higher overhead when using MPEG-2-compliant SNR scalability. Indeed in these cases the quantizer step size is clipped at the maximum value of 31 for much of the time in the base layer and a sizeable proportion of any extra bits added are used to vary the quantizer step size and with any extra overhead associated with such variations. This is evident in both the single-layer and scalable results where the gradient of the lower end of the rate-distortion curve increases with increasing bit rate which is opposite of the normal behaviour, indicating that very few bits are available to be used in coding new texture and are being used instead to code macroblock overhead. Those schemes that use the MPEG-4 syntax are particularly affected by the overhead associated with macroblocks with only a quantizer step size change but no non-zero coefficients (see Table 5.1). The slightly higher bit rates at the start of the sequences which are evident in Figure 5.17(b) are due to the encoder filling the buffer until it is approximately one-third full as described in Section 5.2.6.

Figure 5.17(a) shows that the previously-described inaccuracy in the bit production model for stream morphing resulting from the rate controller not knowing how many of the coefficients in the current layer were coded in a previous layer does affect the uniformity of the allocation of bits in each frame. This is the price paid for the reduced latency that can be achieved with this formulation of rate control for stream morphing. Both stream morphing and MPEG-2-compliant SNR scalability have trouble meeting their bit rate targets in the
middle section of "Foreman" (frames 50 through 80) where rapid camera panning introduces a large amount of new material into the frame that needs to be coded and increases significantly the number of bits required for motion vectors, even if all macroblocks with four motion vectors are converted to use only one vector (Section 5.2.6).

In many of the objective rate-distortion tests MPEG-4 FGS compares much more favourably to the discrete layered systems than in the previous chapter's results for the constant quantizer case. This is due to the fact that in the FGS case no attempt is being made to code perceptually-important areas more accurately and much of the gain in terms of PSNR is in areas where distortion is not as visible as elsewhere. Ideally, FGS selective enhancement would be used to code more accurately those areas that are quantized finely in the base layer however, the software required to perform this was not available and this would have required extra overhead in signalling which macroblocks were to be enhanced. The true impact of perceptual quantization is, however, reflected in the subjective test results in Figure 5.23 and Tables 5.2 and 5.3 where in every case stream morphing has a significant advantage in terms of subjective quality to the "average" user. While this may (at least partly) be due to the lack of perceptual quantization being done in the FGS case, there was also a significant advantage seen for the comparison with MPEG-2-compliant SNR scalability where perceptual quantization was being performed.

The results for the "same PSNR" tests on the low motion sequences "Akiyo" and "Mother & Daughter", where the bit rates were adjusted to give the same PSNR values at the top layer as stream morphing (the second and fourth intervals shown in parts (a) and (b) of Figure 5.23) indicate that there is no subjective performance disadvantage to MPEG-4 FGS at the same PSNR when compared to stream morphing/single-layer coding. For both sequences the confidence intervals for the "average" user contained the zero axis. These increases in quality have come at a very large cost in terms of the number of bits required to code
the sequences using FGS: a more than 300% increase for "Akiyo" and more than 90% for "Mother & Daughter". For "Mother & Daughter" coded using MPEG-2-compliant SNR scalability the results indicated only a very small improvement in subjective performance despite a 30% increase in the enhancement layer bit rate which resulted in the same PSNR value in the top layer as stream morphing. In this sequence blocking artifacts are very noticeable in the light-coloured background (unlike "Akiyo" where the background is mostly dark), especially at the borders with the heads of the persons in the scene. As the accuracy of the (mostly low frequency) coefficients in these areas is not being increased significantly as new layers are added due to the residue not being large enough to create another non-zero coefficient, the amount of visible blocking does not decrease significantly as the bit rate is increased. In contrast, all coefficients in stream morphing are refined as the quantization step size is lowered in the enhancement layers and the discontinuities at block boundaries are reduced.

5.8 Conclusion

This chapter completes the development of stream morphing for SNR scalability by describing a rate control algorithm for use in CBR environments. This algorithm is effective for applications that can tolerate a small encoder processing delay and the lower rates of overhead for stream morphing compared to MPEG-2-compliant SNR scalability have been shown to lead to significantly better performance at low bit rates. The results shown here confirm the superiority of stream morphing in terms of subjective quality where all other techniques tested at the same bit rates were judged to be of poorer quality by almost all observers. The next chapter will describe the application of the concept of stream morphing to spatial and temporal scalability.
Chapter 6

Stream Morphing for Spatial and Temporal Scalability

6.1 Introduction

For scalable video systems operating over a wide range of bit rates from the base layer to the top-most enhancement layer, it is undesirable to be restricted to the use of SNR scalability alone. To achieve low bit rates in the lower layers generally requires the use of very coarse quantization, which is unsatisfactory for users who must use such a service for any period of time. A more balanced system would scale the frame size and/or rate as well as distortion as new layers are added. The existing MPEG-2 and MPEG-4 video standards provide for spatial and temporal scalability in addition to SNR scalability. This chapter describes modifications to the stream morphing technique introduced in Chapter 4 to provide similar functionality. This chapter is structured as follows: Section 6.2 describes spatial scalability, Section 6.3 follows with a description of temporal scalability, Sections 6.4 and 6.5 show results from some comparative experiments of these techniques and MPEG-4, Section 6.6 discusses these results and finally Section 6.7 draws conclusions from this chapter.
6.2 Spatial Scalability

Spatial scalability in MPEG-4 was originally discussed in Section 3.3.1.1. Figure 6.1 (identical to Figure 3.10) shows one of the prediction modes for spatial scalability supported in MPEG-4 where prediction of the enhancement layer frame is done in one of three ways: from the previous layer via an upsampling filter, from the previous frame in the enhancement layer or an interpolated combination of the two. Note that in the cases where prediction is made from the previous layer, the problem of multiple quantization of the input signal exists as it does for SNR scalability (Section 3.7.3.2).

In stream morphing any motion-compensated prediction is always done using reference frames in the same layer, unlike MPEG-4 as described above which can use reference frames from either layer. This must be the case given that a single-layer bitstream can always be recovered at each enhancement layer in a stream morphing system. Such bitstreams are a complete description of the sequence in their own right and as such cannot make reference to any previous "layer". The morphing process uses the values of the quantized coefficients in the previous layer to predict the locations and values of the DCT coefficients in
Figure 6.2: Prediction for stream morphing spatial scalability
the current layer which has been processed with a finer quantizer.

The spatial scalable case, as shown in Figure 6.2 for the IPP GOP structure, differs from SNR scalability in that each macroblock in the base layer maps to four in the enhancement layer which requires changes to how the motion vector and macroblock header information is processed and that to use the previous layer's quantized DCT coefficients for prediction, they must first be upsampled, which is analogous to the pixel-domain upsampling that must occur for any such "inter-layer" prediction with B frames in the enhancement layer of Figure 6.1 for MPEG-4 spatial scalability. As in SNR scalability, where the motion vectors in coincident macroblocks are restricted to having the same value to eliminate overhead required to code minor differences between layers, we must similarly restrict the vectors for spatial scalability such that those in the previous layer can be used to predict the enhancement layer vectors, with only a minimal amount of extra information being added there. Apart from these two differences, which will be discussed in detail below, the operation of a spatial scalable stream morphing system and an SNR scalable one are identical. Indeed, it is possible to mix spatial and SNR scalable layers (along with temporal scalability to be discussed later) in multi-layer stream morphing encoders and decoders.

Note that a diagram similar to Figure 6.2 could be constructed with B-VOPs present, although unlike Figure 6.1 all B-VOP motion-compensated prediction would use reference frames in the same layer.

6.2.1 Previous Layer Upsampling

Stream morphing requires a one-to-one correspondence between quantized DCT coefficients in adjacent layers, which is clearly not immediately available for spatial scalability where consecutive layers have different resolutions. To apply the method here, the coefficients in the previous layer must be upsampled so as to have the same overall size as the layer being coded. While upsampling is
most often done in the pixel domain, a number of methods e.g. [72, 73] exist for performing this operation entirely in the DCT domain. We will apply the method described in [72], since it is more computationally-efficient than [73] and has been observed to give otherwise identical results, for this application.

For upsampling by a factor of two, which is the only case we will consider, each 8x8 block $B$ in the previous layer maps to a 16x16 block $\hat{B}_u$ in the current layer:

$$
\hat{B}_u = \begin{bmatrix}
\hat{B}_1 & O_4 & \hat{B}_2 & O_4 \\
O_4 & O_4 & O_4 & O_4 \\
\hat{B}_3 & O_4 & \hat{B}_4 & O_4 \\
O_4 & O_4 & O_4 & O_4
\end{bmatrix}
$$  \hspace{1cm} (6.1)

where $O_4$ is the 4x4 zero matrix and $\hat{B}_1 \cdots \hat{B}_4$ are calculated as follows:

$$
E = T_LT_4^T \\
F = T_R^T T_4^T \\
P = E^T B E \\
Q = E^T B F \\
R = F^T B E \\
S = F^T B F \\
\hat{B}_1 = (P + Q) + (R + S) \\
\hat{B}_2 = (P - Q) + (R - S) \\
\hat{B}_3 = (P + Q) - (R + S) \\
\hat{B}_4 = (P - Q) - (R - S)
$$  \hspace{1cm} (6.2-6.7)

where $T_n$ is the $n$-point DCT operator matrix (i.e. $DCT_n(x) = T_n x T_n^T$), $T_L$ and $T_R$ are each 8x4 matrices resulting from the partition of the 8x8 DCT operator matrix $T_8 = [T_L T_R]$. [72] shows that this method requires an average of $1\frac{1}{4}$ multiplications and $1\frac{1}{4}$ additions per coefficient of the upsampled version. Note that this is a far greater computational load than the whole of the
stream morphing algorithm for SNR scalability described in Chapter 4, specifically Tables 4.10 and 4.11 show that the number of multiplies required for five layers in a typical system (200k-400k/s) is much less than would be required to upsample from QCIF to CIF in a single layer which at 10fps requires $352 \times 288 \times 1.5 \times 10 \times 1.25 \approx 2.28M/s$ which is about a factor of ten greater. An optimized implementation of this method could, however, reduce significantly the computation required by exploiting the sparseness of the non-zero quantized DCT coefficients in the previous layer.

If macroblocks in each layer are to be processed in the standard row-wise scanning pattern then the coefficients for a complete row of macroblocks in the previous layer need to be stored due to the fact that instead of a one-to-one correspondence between macroblocks, as existed in the SNR scalable case, each macroblock in the previous layer maps to four in the higher layer. For each scan through a row of macroblocks in the previous layer, two are made in the higher layer; the second of these will need to access macroblocks in the previous layer which have already been used and as such need to be stored between passes.

### 6.2.2 Processing of Motion Vectors

As previously described for the SNR scalable case, the values of the motion vectors in consecutive layers in a stream morphing system need to be managed carefully to minimize the overhead associated with morphing any changes that exist between layers. Unlike SNR scalability, where we can insist that the motion vectors used in coincident macroblocks must be equal, spatial scalability generally requires larger motion vectors be used in the higher-resolution layer (when their length is measured relative to the pixel pitch) and as such some form of motion vector morphing needs to be developed for use here. This morphing is done in two stages: firstly a predicted motion vector is formed from the previous layer then a difference signal is coded to reflect any change between this prediction and the actual value of the motion vector in the enhancement layer.
6.2 Spatial Scalability

For spatial scalability, the predicted motion vector is found by doubling the length of the coincident vector in the previous layer. Note that since the resolution of each frame has doubled, the length of the vector is still the same as the previous layer if measured in terms of the overall frame size or any features in the image itself. If the macroblock in the previous layer has four motion vectors then each of these vectors maps to individual macroblocks in the upper layer. Otherwise, the same vector is used in calculating the prediction in each of the four macroblocks that lie above it. For each motion vector component, an arithmetic coded symbol is generated describing the difference between the prediction and the final value for that component. For large differences, this single symbol may be insufficient to completely describe the change and may be followed by an escape sequence; details of this are contained in Appendix A.

If four motion vectors are required in any upper layer macroblock, then the same prediction vector is used for all four and eight differences must be coded (two components each for four vectors) rather than two. For MPEG-4 version 1 systems that support half-pel prediction only, the predicted vector in the enhancement layer will be full-pel. For half-pel prediction to be done in the upper layer the additional coded component must provide any fractional part of each motion vector. There is a trade-off here between motion vector precision and the number of bits required to code the additional component: MPEG-4 version 2 systems that support quarter-pel prediction may choose not to use such precise vectors in spatial scalable enhancement layers due to the additional cost that is not recovered by improved prediction quality.

For the experiments conducted here, motion estimation was performed as normal in the upper layer, then the vectors for the lower layer were formed by taking the motion vectors from each group of four macroblocks that are above a single macroblock in the previous layer and dividing the magnitude of each by two. If the four motion vector components in each of the $x$ and $y$ directions differ by one pixel or less then the motion field is approximately uniform and
6.3 Temporal Scalability

A single motion vector is used in the lower macroblock rather than coding four discrete vectors. Divergent fields that do not meet this criteria map different motion vectors to the individual 8x8 blocks in the coincident macroblock in the lower layer. The cost associated with the use of four motion vectors per macroblock in the enhancement layer was found to be prohibitive and was not used for our experiments. A more complex mode decision algorithm might decide to use four vectors in cases where the motion field is non-uniform across the macroblock and the savings associated with the use of a single vector will be lost to extra DCT coefficients that would be required to code the texture information. The encoder description in this paragraph is an illustration of one possible strategy; as in previous cases the actual method the encoder uses to generate motion vectors is non-normative.

6.2.3 Syntax

Apart from the motion vector morphing symbols described above, the only other change required to the encoder is for different symbol statistics to be used for the spatial scalable case, rather than re-using those for SNR scalability. As the results given later will show, the amount of inter-layer correlation that can be exploited by morphing upsampled quantized non-zero coefficients in inter-coded pictures is significantly lower than for the SNR scalable case due to the fact that the coefficients in each layer often do not "line up". This significantly alters the probability distribution of the symbols used for describing what happens to previous layer non-zero coefficients in the current layer (Section 4.3.6.2). The definitions of $D_{qp}$ and $D_{rate}$ from Section 4.3.7 are retained.

6.3 Temporal Scalability

Temporal scalability in MPEG-4 was first discussed in Section 3.3.1.2. Unlike spatial and SNR scalability, it is possible to use the existing MPEG-4 approach to
6.3 Temporal Scalability

Figure 6.3: Prediction for MPEG-4 temporal scalability (reproduction of Figure 3.11)

Figure 6.4: Interleaving temporal scalable layers to form a single layer
temporal scalability while retaining the ability to create a single-layer bitstream from merged consecutive layers as has been shown to be possible for stream morphing. Consider again Figure 3.11 (repeated here as Figure 6.3): the first and third B frames in the enhancement layer are interleaved between the existing frames in the previous layer. If those frames in the enhancement layer that are temporally-coincident with previous layer frames (in the figure these are the P frame and the second and fourth B frames) are skipped then a single-layer bitstream can be created by similarly interleaving the corresponding bitstream segments as depicted in Figure 6.4. While this in itself may be an attractive method, the remainder of this section develops an alternate formulation for temporal scalability that is closer to stream morphing as previously described. The new method has a number of advantages: the complexity associated with bi-directional prediction is not required and there is more control over the quality of the enhanced sequence as the temporally-coincident frames may be refined in the upper layer. There are also issues with the subjective quality of the approach described in Figure 6.3 that are not present when stream morphing is used. Consider that the even-numbered frames in Figure 6.3 have been quantized twice, once in each layer whereas the remaining frames are only processed once in the upper layer. For low quality/bit rates interleaving the two sets of frames results in "flickering" since there is often a significant quality difference between the two. The sequence that has been quantized twice generally has slightly higher quality, even if the same quantizer step size is used in both layers (due to the action of the motion-compensated prediction in the upper layer).

As there is no longer a one-to-one correspondence between frames in consecutive layers in temporal scalability, there are a number of choices for the reference frame (or frames) to be used with stream morphing. Figure 6.5 shows the arrangement that is investigated in this thesis, which we will call the "non-causal" configuration for reasons that will be explained shortly. The solid arrows in the figure denote motion-compensated prediction whose reference pic-
6.3 Temporal Scalability

Figure 6.5: Prediction references for temporal scalability (IPP GOP structure, “causal” configuration)

tures are, as in previous cases with stream morphing, confined to frames in the same layer which is a requirement if a single-layer stream is to be recovered from any enhancement layer. The dashed arrows represent reference frames from which DCT coefficients are morphed to generate the coded texture in the given frame (Figure 6.2 used the same notation when describing spatial scalability). Those P frames in the enhancement layer that are not coincident with a frame in the previous layer are morphed from the texture of the following frame in the previous layer, which in fact corresponds to one enhancement layer frame period into the future, hence the name “non-causal”. The quantized DCT coefficients in the remaining coincident P frames are morphed using two references; the previous frame in the current layer and the coincident frame in the previous layer. For real-time applications where delaying the enhancement layer for the receipt of the future previous layer frame required here is unacceptable, a “causal” configuration is also possible where the intermediate upper layer frames (the second and fourth frames in Figure 6.5) are coded without morphing. This arrangement requires a higher bit rate in the absence of this prediction and will not be considered further here.

Temporal scalability differs from SNR scalability in two main ways: firstly
the way that the enhancement layer motion vectors are calculated and coded and secondly the way that the bi-directional morphing in coincident frames is performed. These two changes are described in the following sections. The coding of the first frame is the same as in the SNR scalable case and the coefficient coding for the intermediate frame that is morphed from the future previous layer frame is performed in a manner identical to SNR scalability.

6.3.1 Processing of Motion Vectors

Like the previously described SNR and spatial scalable cases, restrictions are applied to the motion vectors to ensure that the overhead associated with coding them is kept to a minimum. For temporal scalability, we require that each motion vector in a lower layer predicted frame must be equal to the sum of the spatially-coincident vectors in the corresponding enhancement layer frames, e.g. in Figure 6.5 a motion vector in macroblock \((x, y)\) of the second frame in the previous layer (the first P frame) must be equal to the sum of the motion vectors in macroblocks \((x, y)\) of the second and third enhancement layer frames. For motion vector fields that do not change abruptly over the lower layer frame period, this is a good approximation of what naturally occurs and enforcing this restriction does not significantly affect the quality of the prediction in either layer. The motion vector for the first enhancement layer frame of each pair \(^1\) is coded in the bitstream while the corresponding vector in the other frame can be determined by subtracting the vector in the first frame from the vector in the previous layer, given that the sum of the two enhancement layer vectors must equal the previous layer vector. Furthermore, given the assumption that the motion vector field is approximately uniform over the period of the previous layer frame, the vectors in the first frame of each pair can be predicted using half the value of the corresponding vectors in the previous layer frame.

\(^1\)An enhancement layer frame pair is the set of two frames that correspond to a single previous layer frame, e.g. the second and third enhancement layer frames in Figure 6.5.
The difference between the actual value of the vector and this prediction is then coded in a similar way to the motion vector morphing that has been described for spatial scalability.

Special care needs to be taken when coincident macroblocks in an enhancement layer frame pair have different numbers of motion vectors i.e. one vector in one macroblock and four in the other. In this case the corresponding macroblock in the previous layer will always have four vectors. In order to apply the restriction that the enhancement layer vectors sum to give the previous layer vector, the macroblock with the single vector will be considered (temporarily) to have four vectors, all of equal value. Where the macroblock in the first frame of the pair has four vectors, each of these would normally be morphed with respect to the prediction formed from halving the magnitudes of the previous layer vectors. This is, however, inefficient since we know that each vector differs from the corresponding previous layer vector by the same amount (the single vector in the corresponding macroblock in the next frame). In this case it is more efficient to code that single vector in the first frame rather than the second, to avoid having to code four differential values. This single vector is again predicted from half the magnitude of a previous layer vector. In this case, the encoder selects which of the four previous layer vectors best matches the vector to be coded and signals that in the bitstream followed by a differential value.

For example, if a macroblock in the first frame of an enhancement layer pair has four vectors \{(1, 2), (1, 3), (1, 1), (2, 2)\} and one vector (1, 1) in the second frame then the previous layer has the four vectors \{(2, 3), (2, 4), (2, 2), (3, 3)\}. The encoder signals in the first frame of the pair that the third vector in the previous layer is to be used for prediction and that the differential value is \(0,0\). Once the decoder knows the previous layer vectors and that there are four vectors in the first enhancement layer frame and one in the second, it decodes that the prediction for the vector in the second frame is the third vector in the previous layer, (1, 1), adds the differential signal \(0, 0\) and can then calculate the first
6.3 Temporal Scalability

frame vectors as: \{(2, 3), (2, 4), (2, 2), (3, 3)\} \cdot \{(1, 1), (1, 1), (1, 1), (1, 1)\} = \{(1, 2), (1, 3), (1, 1), (2, 2)\}.

6.3.2 Bi-Directional Morphing

For those predicted frames in the enhancement layer that temporally coincide with a previous layer frame, two reference frames are used by the morphing process (Figure 6.5). Where corresponding syntax elements in the two reference frames are equal, we should expect that the element in the frame being coded will also have this value whereas if they are not then the statistics of coding the symbol will be substantially different, with a more uniform spread of values. For example, if a macroblock is skipped in both of the reference frames then we should expect that it will also be skipped in the frame that is being coded whereas if one or other is coded then the probability that the macroblock is coded in the frame in question is much higher. To reflect this, the syntax for temporal scalability has many cases where two symbol types are defined for syntax elements that are represented by a single type in the SNR scalable case; one type for use where corresponding values in the reference frames are equal and another where they differ. Details of symbol types are contained in Appendix A.

For other types of scalability, if inter-frame prediction is used in a given macroblock in one layer but not the other then the contents of the macroblock are passed through in their original syntax (Section 4.3.2). For bi-directional morphing the macroblock is passed through if either of the two reference macroblocks uses a different prediction type, i.e. one macroblock is coded intra and the other in inter mode. As discussed in the previous section, different numbers of motion vectors can be used in enhancement layer frame pairs where corresponding macroblocks are both coded in inter mode.
6.3.3 Implementation Issues

This formulation of temporal scalability requires that frames in the previous layer (apart from the initial I frame) and first frame of each enhancement layer pair be processed twice: the first time to encode or decode the frame itself and a second time as a reference for bi-directional morphing which occurs in the second frame of each enhancement layer pair. As such, it is necessary to store those frames after encoding or decoding such that they can be used again during the morphing process. In some cases, such as an encoder that writes each bitstream to a file on disk, this storage already exists. At the decoder, however, or for a streaming encoder that sends data directly over the network, extra buffering will be needed.

As for the spatial scalable case, probabilities for the arithmetic coder are kept separately for temporal scalability rather than re-using those defined for SNR scalability. Appendix A describes the temporal scalability syntax in detail.

6.4 Experimental Procedure

For the experiments in this chapter only two layer systems are tested, in all cases using constant quantizer step size values that are equal in both layers. If higher quality is desired then additional SNR scalable layers can be added in addition to temporal or spatial scalability. Another motivation for not increasing the enhancement layer quality in the MPEG-4 temporal scalable case is that the frames from the lower layer are interleaved with those decoded in the enhancement layer to provide the high-quality version of the sequence. In this case it is necessary to equalize the qualities in both layers to avoid any flickering effects that might be visible in the interleaved sequence. The base layer coding in all tests has a single I frame at the start of the sequence followed by all P frames. The two sequences "Mother & Daughter" and "Foreman", which have been used in previous chapters as examples of low- and high-motion sequences, are again
tested here.

For the spatial scalable case comparison is made with MPEG-4 utilizing bi-directional prediction in the enhancement layer as in Figure 6.1 (originally Figure 3.10). MPEG-4 defines another mode with P frame prediction only in the enhancement layer. The performance of this prediction scheme was found to be considerably worse than the other methods shown here and is not included. Tests were conducted at 10fps and 30fps on both sequences with the base layer size being QCIF and the enhancement layer being CIF.

Two relevant temporal scalability modes exist in the MPEG-4 reference software that is used for comparison with stream morphing: "type 0" uses P frame prediction only with the preceding base layer frame as the reference whereas "type 2" uses the bi-directional prediction shown in Figure 3.11 and Figure 6.3 except that the base layer reference for the B-VOP prediction is the preceding frame rather than the next frame. Tests were conducted for QCIF and CIF-sized sequences with the base layer at 15fps and the enhancement layer at 30fps in all cases.

6.5 Results

Figures 6.6 and 6.7 show the results of the spatial scalability tests for "Foreman" and "Mother & Daughter" respectively.

Figures 6.8 and 6.9 show the results of the temporal scalability tests. For the type 2 MPEG-4 temporal scalability tests, it was noted that despite the use of identical quantizer step sizes in both layers, the reconstructed sequence at the top layer (with the base layer frames interleaved between those of the enhancement layer) was subject to noticeable flickering and the PSNR of those frames that make up the enhancement layer had PSNR values up to 1dB higher than the surrounding base layer frames. This was especially evident for low rate/quality sequences using high values for QP. The type 2 results for "Mother & Daughter"
6.5 Results

Spatial Scalability results for Foreman (QCIF->CIF 10fps)

(a) 10fps

Spatial Scalability results for Foreman (QCIF->CIF 30fps)

(b) 30fps

Figure 6.6: Spatial scalability results for "Foreman"
Figure 6.7: Spatial scalability results for "Mother & Daughter"
Figure 6.8: Temporal scalability results for “Foreman”
Figure 6.9: Temporal scalability results for "Mother & Daughter"
which perform far worse than simulcast generate extremely large numbers of non-zero motion vectors, many with large magnitudes. For the 30fps test results shown here with QP=31 only 1.5% of the bitstream is used for coding DCT coefficients, more than half the bitstream is used for coding motion vectors and the remainder is used to code macroblock header information.

Note that the single-layer rate-distortion behaviour in Figure 6.9(b) is not monotonic. It has been observed that with the MPEG reference software (unmodified) that for low-motion sequences at low quality levels a reduction in the quantizer step size can result in fewer bits being generated for the coding of an entire sequence. This is likely due to the behaviour of the motion estimation process in flat areas which can often produce large motion vectors that require many bits to code. More accurate coding (from the use of the finer quantizer) early in a sequence can improve the performance of the motion estimation to such an extent that the reduction in the number of bits required to code motion vectors more than compensates for the additional texture.

6.6 Discussion

The spatial scalability results show that the performance of stream morphing is very close (within 0.5dB) to that of the existing MPEG-4 technique, the only difference being at high quality levels for the high-motion sequence “Foreman”. A significant factor affecting performance that has been mentioned earlier is the requirement that non-zero DCT coefficients must be located in exactly the same position in successive layers for stream morphing to work efficiently. For high-motion sequences at low frame rates (e.g. the 10fps results for “Foreman” in Figure 6.6(a)) the motion-compensated prediction difference to be coded has a relatively large magnitude in all layers and is more likely to share common features after the DCT and quantization are performed and so good performance is achieved. For low-motion and/or high frame rates the spectrum of
the prediction difference in the DCT-domain is flatter and the number of coincident coefficients that will be above the quantization threshold in both layers is reduced. Even without considering any effects due to upsampling, the reference frames for motion-compensated prediction will be different in both layers which perturbs the difference signal such that after the DCT is taken the status of those coefficients near the boundary of the central (zero) quantization bin may be different. Poor coefficient alignment between layers at high quality levels is the cause of the deterioration in performance at high bit rates that can be clearly seen in Figures 6.7 and 6.8.

The results for temporal scalability indicate excellent performance in all cases at low bit rates. For the high-motion “Foreman” sequence, the type 2 prediction for standard MPEG-4 gives objective quality that exceeds stream morphing at high bit rates. It has been observed, however, that flickering is evident in the high quality service, especially at low bit rates. The reason for this is that two frame stores are maintained, one in each layer, and there can be substantial differences between these that can be contained in the central quantization bin and as such will not be corrected. This is especially evident for the DC and low frequency components in relatively static areas when the frames from the two layers are interleaved. This is another example of how stream morphing, which always quantizes a given input signal only once, gives superior subjective performance to other types of scalable coding. For the low-motion sequence “Mother & Daughter” the type 0 prediction performed better than stream morphing at relatively high bit rates for each frame size tested. Again, the most likely explanation for this is that the coefficients in the reference frames do not align themselves as well when the magnitude of the prediction difference is small.

For spatial and temporal scalability, as discussed above, and to a lesser extent for SNR scalability, there is scope for performance gains to be made through the use of non-normative techniques at the encoder for maximizing the amount
of alignment between coefficients in successive layers. One approach may be to consider coefficients that are coincident with non-zero coefficients in the adjacent layer (above or below) and are close to the boundaries of the zero quantization bin but which do not create a non-zero value and to artificially raise their absolute value so that a non-zero quantized coefficient is created. “Promotion” of some coefficients may allow for other coefficients that are not correctly aligned to be “demoted” so that no non-zero coefficients are created without any significant increase in the overall level of distortion. This is complicated in the spatial scalable case by the one-to-many relationship between a given coefficient in the lower layer and those in the upper layer blocks.

For both the spatial and temporal cases it should be noted that stream morphing did not require the complexity associated with bi-directional prediction which was used for the MPEG-4 comparison techniques. The other advantages of stream morphing identified in previous chapters still hold for spatial and temporal scalability, notably that ability to recover single-layer descriptions upstream of the decoder and the subjective quality advantages that follow from quantizing the signal only once regardless of the number of layers present.

6.7 Conclusion

This chapter has completed the description of stream morphing by adapting the algorithm of Chapter 4 to spatial and temporal scalability. Spatial scalability with stream morphing was shown to give results that are very similar to MPEG-4 while temporal scalability performs substantially better for high-motion sequences while avoiding subjective problems related to the interleaving of base and enhancement layer frames in MPEG-4.
Chapter 7
Conclusions and Future Research Directions

This thesis has introduced stream morphing, a new approach to the scalable video problem, which has been developed primarily for the case of SNR scalability but has been shown to also work for temporal and spatial scalability. The motivation for a new approach to scalable video was outlined in Chapter 3 with an examination of two existing SNR scalability techniques, one from MPEG-2 and Fine Granularity Scalability from MPEG-4. These techniques were found to both have significant problems. FGS has poor performance for many sequences, specifically those with low to medium amounts of motion, due to the lack of motion-compensated prediction outside of the base layer. The MPEG-2-compliant SNR scalability, on the other hand, had better performance due to the presence of motion-compensated prediction loops in all layers. However, its method of coding residual DCT coefficients results in significant errors remaining in higher layers due to the fact that coarsely-quantized coefficients coded in the lower layers are often not refined in higher layers. The reason for this is that for a coefficient to change its value another non-zero DCT coefficient must be generated in a higher layer which implies that the residue in that layer is especially large. This is unlikely given that for typical enhancement layer bit rates the quantizer step size used is not significantly smaller than in the base layer due to the approximately quadratic relationship between bit rate and quantizer...
step size.

Chapter 4 began with the observation that rather than updating the texture in each layer as the MPEG-2 SNR scalability does by adding new quantized texture, we can instead work at the level of the quantized coefficients and update their values to reflect the use of a finer quantizer in the higher layer. This approach has several significant advantages. The first is that this solves the quality problems associated with MPEG-2 SNR scalability (while retaining its very low decoder complexity) and in the process further reduces the number of bits required to code a given sequence. This is due to the fact that improved coding accuracy for coefficients results in better prediction in successive frames which was shown to reduce the number of non-zero coefficients and coded blocks required. The second advantage relates to the fact that the morphing process works with a series of single-layer descriptions of increasing quality that can be generated and recovered at any time at very low computational cost. All operations are performed at the quantized coefficient level without the need to perform any forward or inverse DCTs nor motion-compensated prediction. This is in contrast to traditional scalable video methods which can be converted into single-layer form only by full decoding and re-encoding. This ability to move between scalable and single-layer forms with ease allows for on-the-fly generation of scalable content from a set of single-layer bitstreams stored on a video server in any combination. At the decoder side, this allows for single-layer representations to be recovered upstream of the final decoder which can then be used for single-layer switching. Alternatively, the highest quality single-layer bitstream can be forwarded directly to an unmodified single-layer decoder over a narrow-bandwidth channel such as a modem or DSL connection. The use of scalable coding in this case is completely transparent to the decoder whilst retaining the advantages associated with the use of scalable video over heavily congested portions of the network which may exist upstream. Stream morphing requires the use of arithmetic or range coding which increases computational
complexity slightly compared to the MPEG-2 approach.

Chapter 5 developed a rate control framework for stream morphing in the SNR scalable case. As stream morphing works at the level of the quantized DCT coefficients, it was not possible to use pixel-domain measures to estimate the number of bits to be generated by the coding of a particular frame and hence the level of quantization required to achieve the target bit rate. Some pre-analysis of the unquantized coefficients is required which does not in itself require extra computation. This does, however, introduce processing delay in the encoder which may be unacceptable for latency-sensitive real-time applications such as videoconferencing which will require a more approximate approach to rate control. Chapter 5 also provided subjective test results that showed for a range of different video sequences stream morphing consistently outperformed other techniques at identical bit rates. The sequences tested varied from those with low motion to high motion sequences where single-loop techniques such as FGS work well.

Finally, Chapter 6 extended the basic concept of stream morphing to cover spatial and temporal scalability in addition to SNR scalability as previously outlined. Performance for these two new cases is comparable to that achieved by MPEG-4 at low bit rates however at high rates performance is close to or slightly worse than MPEG-4. For temporal scalability it was noted, however, that the MPEG-4 approach suffers from issues with subjective quality that are not present in the stream morphing system. The efficiency of stream morphing, in the SNR scalable case as well as for temporal and spatial scalability, depends greatly on the proportion of non-zero coefficients that are coincident with non-zero coefficients in previous layers. We believe that further research into quantization at the encoder will improve performance by causing more coefficients to be correctly aligned, especially for temporal and spatial scalability. Such optimizations will not require changes to the stream morphing process itself, only to the single-layer inputs to that process.
To summarize: stream morphing provides a complete replacement for the scalable video techniques in the present MPEG standards. For the SNR scalable case superior performance has been demonstrated here and in the future we anticipate improved performance for temporal and spatial scalability also. In addition, stream morphing provides a new level of flexibility that cannot be matched by existing techniques.

7.1 Future Research

The following list outlines some possible areas for future work. They are listed in approximate order of importance.

- The stream morphing algorithm needs to be extended to work with B frames. This should be relatively straightforward for the constant quantizer case since we can work with pre-computed bitstreams (e.g. from the VM software) although we would need to ensure the same prediction modes are used in all layers which would require some coupling of the encoders. To do rate control as well would require significant modification of the experimental software. There will need to be extra syntax for describing elements that only appear in B-VOPs.

At this point there are two obvious approaches for implementing the new functionality: the existing dataflow software could be modified to support B-VOPs or an existing MPEG-4 implementation such as the VM software could be used, the latter is likely to be faster (even considering the time required to familiarize with the other software).

- Stream morphing, as it is current described, only attempts to deal with the problem of extra coefficients being present in the enhancement layers of scalable coders due to quantization effects (Section 3.7.3.2) and not the coefficient sparseness problem (Section 3.7.3.1).
One possible way to deal with the scanning problem would be to use a single incremental scan which is advanced in each layer and to identify in a lower layer any coefficients that will appear in higher layers. Figure 7.1 shows scans across the coefficients of a block in a three layer coder, the filled circles are the non-zero coefficients that are present in that layer while the empty circles denote coefficients that are zero in that layer but which become non-zero in a higher layer. The arrows in the figure denote the run-level symbols that make up the scan in each layer. Enhancement layer scans are started from the point in the base layer where the final coefficient was located (filled circles framed in a square). Special symbols would be required to code those “placeholder” coefficients that are zero in the current layer but are non-zero in one or more of the higher layers. Ideally this would be done in the base layer also although this would mean modification of the base layer syntax, something which has been unnecessary so far. These coefficients would then be processed as part of the scan of “existing” coefficients (Section 4.3.6) even though their values in all previous layers may be zero.

It is not clear how well this method would work in that we are moving much of the information down into the lower layers. The idea of scalable video is to split the total bandwidth fairly evenly between layers. Similarly, we would expect there to be minimal quality gain in the lower layers since symbols are coded there which do not affect the quality of the layer they...
are in but are present to assist coding of higher layers. Rate control would be a significant challenge since the number of bits required for a given layer would depend on what is present in layers higher up (in stream morphing it is only dependent on the current and lower layers).

Something similar to this idea was attempted early on during the research for this thesis however this was done with VLCs only in the enhancement layer and did not use the incremental scan (the locations of all coefficients in all layers was coded in the base layer). Another attempt should be made at this using arithmetic coding and the incremental scan.

- The performance of stream morphing is critically dependent on the correct alignment of coefficients in successive layers. Section 4.3.6.4 describes a simple method used only in the SNR scalable case for promoting coefficients close to the quantization boundary which are non-zero in surrounding layers to prevent a coefficient from being deleted in one layer having to be restored in a higher layer. Better alignment of coefficients could significantly improve the performance of spatial scalability where upwards of 50% of previous layer coefficients are zero in the next layer in many cases. Some form of pre-processing of unquantized DCT coefficients in the encoder may be able to achieve this. A further idea for spatial scalability might be to look at each group of four blocks that lie above a base layer block and to exploit any inter-block redundancy that might exist there.

- For the spatial scalable tests in Chapter 6 the use of four motion vectors per macroblock in the enhancement layer has been disabled due to the extremely large volume of data that is required to code those vectors differentially from the corresponding single base layer vector. For highly divergent motion vector fields, however, the use of four vectors is desirable. Some research should be directed towards modification of the one/four vector mode decision done during the motion estimation process in spa-
tial scalable enhancement layers to allow four vectors to be used in cases where it is likely that the extra bits used to morph the enhancement layer vectors will be less than the number of bits saved in the coding of the macroblock texture.

- Use an adaptive arithmetic or range coder for at least a subset of the symbols in stream morphing and measure how much improvement this provides and estimate the increase in complexity that results.

- Implement causal temporal scalability and compare this to the non-causal case described in Section 6.3.

- Experiment with setting QP values for rate control DCT thresholds on a frame-by-frame basis (Section 5.2.5) rather than using the same fixed values at all times. If these values are chosen well then the errors introduced by interpolation between counts will be reduced at the risk of increasing the error when sudden changes in motion occur and the quantizer step size that achieves the desired rate is located outside the area in which the thresholding has been done, which is likely to be relatively rare.

- Stream morphing should be “ported” from MPEG-4 to H26L which ought to be a relatively straightforward process, although the number of different types of symbols to be morphed is substantially higher in H26L.

- Rate control needs to be implemented for spatial and temporal scalability.

- Experiments need to be conducted to compare stream morphing and MPEG-2-compliant SNR scalability when MPEG quantization is used rather than the H.263 quantization. The thresholding operation for the rate control algorithm in Chapter 5 would need to be substantially modified and the computational complexity increased since the quantization matrix used would change the width of the central (zero) quantization bin across each 8x8 block.
References


REFERENCES


REFERENCES


[58] Wei Ding and Bede Liu, "Rate Control of MPEG Video Coding and Recording by Rate-Quantization Modeling," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 6, no. 1, pp. 12-20, February 1996.


Appendix A

Stream Morphing Bitstream Syntax for MPEG-4

A.1 Introduction

This appendix provides the complete syntax to the stream morphing algorithm as discussed in Chapter 4 for SNR scalability and Chapter 6 for spatial and temporal scalability. The format used here is modeled on that used in the MPEG-4 Verification Model document [54] except that the items marked in bold type are arithmetic coded symbols rather than VLCs.

This syntax has been designed to be used in morphing between MPEG-4 Version 1 bitstreams. While the concept of stream morphing can be applied to other DCT-based video coding standard such as H.263, modifications to this syntax will need to be made to accomodate any different syntactic elements present in the bitstreams used by those standards.

Section A.2 describes the syntax of the overall bitstream, section A.3 describes the semantics of those elements of the syntax that are not arithmetic coded symbols, section A.4 lists the types of arithmetic coded symbols that are used and finally section A.5 shows the mapping between the symbol types and the individual probability models that are used and how alternate models are chosen.
A.2 Bitstream Syntax

Figure A.1 shows the complete syntax for a stream morphing enhancement layer bitstream. Table A.1 provides a cross-reference between the syntax elements here and the contents of Chapters 4 and 6.

<table>
<thead>
<tr>
<th>Line Numbers</th>
<th>Coding of</th>
<th>Ref.</th>
</tr>
</thead>
<tbody>
<tr>
<td>21-24</td>
<td>Quantization step size</td>
<td>4.3.1</td>
</tr>
<tr>
<td>25-36</td>
<td>Macroblock mode</td>
<td>4.3.2</td>
</tr>
<tr>
<td>41-50</td>
<td>Intra AC prediction enable/disable</td>
<td>4.3.4</td>
</tr>
<tr>
<td>51-54</td>
<td>Macroblock skipped</td>
<td>4.3.3</td>
</tr>
<tr>
<td>57-70,95-99</td>
<td>Motion vector morphing (temporal scalability)</td>
<td>6.3.1</td>
</tr>
<tr>
<td>71-73,152-175</td>
<td>Motion vector morphing (spatial scalability)</td>
<td>6.2.2</td>
</tr>
<tr>
<td>178-189</td>
<td>Block coded</td>
<td>4.3.5</td>
</tr>
<tr>
<td>191-197,205-222</td>
<td>Existing intra coefficients</td>
<td>4.3.6.1</td>
</tr>
<tr>
<td>223-224,86-93</td>
<td>Existing inter coefficients</td>
<td>4.3.6.2</td>
</tr>
<tr>
<td>225-227,80-84</td>
<td>Inter coefficient restoration</td>
<td>4.3.6.3</td>
</tr>
<tr>
<td>232-241,123-150</td>
<td>Number of new coefficients in block</td>
<td>4.3.6</td>
</tr>
<tr>
<td>244-257</td>
<td>New coefficients not present in previous layer</td>
<td>4.3.6</td>
</tr>
</tbody>
</table>

Table A.1: Cross-reference between syntax and content in Chapters 4, 6

A.2.1 Notes

1. For spatial scalability and the first frame of an enhancement layer frame pair in temporal scalability that is not temporally-coincident with the previous layer frame, the quantization bins of intra coefficients in successive layers do not necessarily overlap as they do for SNR scalability as described in section 4.3.6.1. In these cases, it is necessary to morph DC coefficients which is the purpose of lines 191-197 of the syntax. The DC coefficient case has been brought forward here out of the loop that processes the
morph_video_object_layer() {
  morph_seq_header()
  do {
    morph_video_object_plane()
  } while (next_bits() == VOP_start_code)
}

morph_video_object_plane() {
  do {
    morph_vop_header()
    start_range_coder()
    foreach_macroblock_in_frame {
      morph_macroblock()
    }
    stop_range_coder()
  } while (nextbits_bytealigned() != '0000 0000 0000 0000 0000 0000 0000 0000')
}

morph_macroblock() {
  if ((prev_1ayer_qp >= 30) && !isvbr)
    dquant30
  else
    dquant
  if (istemporal && !prevtemporal) {
    if (prev_frame_mb_is_intra == prev_layer_mb_is_intra) {
      if (prev_layer_mb_is_intra)
        temporal_mode_same_intra
      else
        temporal_mode_same_inter
    } else
      temporal_mode_diffene
t
  } else if (prev_layer_mb_is_intra)
    change_mode_intra
  else
    change_mode_inter

Figure A.1: Stream morphing syntax
if (passthru) {
    mpeg4_passthru_macroblock()
    return
}

if (mb_is_intra) {
    if (istemporal && !prevtemporal)
        temporal_intra_ac_pred
    else {
        if (prev_layer_is_intra_ac_pred)
            intra_ac_pred_prev
        else
            intra_ac_pred_no_prev
    }
}

if (!mb_is_intra && istemporal && !prevtemporal)
    temporal_mb_coded
else if (VOP_coding_type != "I" && !mb_is_intra && prev_layer_mb_skipped)
    mb_coded_prev_skipped

if (!tnb_is_intra) {
    if (istemporal) {
        if (prevtemporal) {
            if ((mv_count == 4) && (prev_layer_mv_count == 4)) {
                temporal_nextframe8x8
                for (i = 0; i < (next_frame_mv_count*2); ++i)
                    encode_temporal_vector_comp()
            } else if ((mv_count == 1) && (prev_layer_mv_count == 1)) {
                code_temporal_vector_comp()
            } else {
                temporal_nextframe16x16
                code_temporal_vector_comp()
            }
        }
    }
}

Figure A.1 (continued)
} else if (isspatial)
    for (i = 0; i < mv_count; ++i)
        morph_spatial_mv(i)
}

if (mb_coded)
    for (i = 0; i < block_count; ++i)
        morph_block(i)

}

code_inter_restored() {
    inter_restored
    while (escape)
        refine_contract_escape
}

code_inter_existing_coeff() {
    if (abs(prev_layer_coeff_value) == 1)
        inter_existing_coeff_l
    else
        inter_existing_coeff
    while (escape)
        refine_contract_escape
}

code_temporal_vector_comp {
    temporal_mvdiff_sign
    while (escape)
        temporal_mvdiff_mag
}

code_temporal_existing_coeff() {
    if (prev_layer_coeff_value == prev_frame_coeff_value) {
        if (mb_is_intra) {
            temporal_existing_intra_same
        } else {
            temporal_existing_inter_same
        }
    } else {
        while (escape)
            refine_contract_escape
    } else {

Figure A.1 (continued)
if (mb_is_intra) {
  temporal_existing_intra_diff
  while (escape)
  
  temporal_existing_intra_mag
  }
else {
  temporal_existing_inter_diff
  while (escape)
  
  temporal_existing_inter_mag
  }
}

code_num_new_coeffs_intra() {
  do {
    num_new_coeffs_intra
  } while (escape)
}

code_num_new_coeffs_inter0() {
  do {
    num_new_coeffs_inter_base0
  } while (escape)
}

code_num_new_coeffs_inter1() {
  do {
    num_new_coeffs_inter_base1
  } while (escape)
}

code_num_new_coeffs_inter() {
  do {
    num_new_coeffs_inter
  } while (escape)
}

Figure A.1 (continued)
A.2 Bitstream Syntax

code_inter_newcoeff_type() {
  inter_newcoeff_type
  while (escape)
    refine_contract_escape
}

morph_spatial_mv(i) {
  if (mv_count == 1) {
    if (prev_layer_mv_nonzero(i)) {
      do {
        spatial16_mvdiff_nonzero
      } while (!escape);
    } else {
      do {
        spatial16_mvdiff_zero
      } while (!escape);
    }
  } else {
    spatial_mvdiff_sign
    if (prev_layer_mv_nonzero(i)) {
      do {
        spatial8_mvdiff_nonzero
      } while (!escape);
    } else {
      do {
        spatial8_mvdiff_zero
      } while (!escape);
    }
  }
}

morph_block(i) {
  if (istemporal && !prevtemporal)
    temporal_block_coded
  else if ((!mb_is_intra || !prev_layer_block_coded(i)) ||
    isspatial || istemporal || (qp > prev_layer_qp)) &&
  (!prev_layer_block_coded(i)) || (prev_layer_mb_is_intra &&
    (qp > prev_layer_qp))) {

Figure A.1 (continued)
if (block_is_luminance)
    block_coded_prev_skipped_lum
else
    block_coded_prev_skipped_chrom
} else
    block_coded = 1

if (mb_is_intra && (isspatial || istemporal)) {
    if (istemporal && ! prevtemporal)
        code_temporal-existing_coeff()
    } else {
        code_inter-existing_coeff()
    }
}

for (j = 0; j < 64; ++j) {
    if ((prev_layer_quant_texture(i, j) != 0) ||
        (istemporal && ! prevtemporal && prev_frame_quant_texture(i, j) != 0)) {
        if (istemporal && ! prevtemporal)
            code_temporal-existing_coeff()
        else if (mb_is_intra && !isspatial && ! istemporal) {
            if (range_num_quant_bins(prev_layer_intra_range(i, j), qp) == 2) {
                if (j == 0)
                    intra_dc-existing_coeff2
                else
                    intra_ac-existing_coeff2
            } else if (range_num_quant_bins(prev_layer_intra_range(i, j), qp) == 3) {
                if (j == 0)
                    intra_dc-existing_coeff3
                else
                    intra_ac-existing_coeff3
            } else if (range_num_quant_bins(prev_layer_intra_range(i, j), qp) == 4) {
                if (j == 0)
                    intra_dc-existing_coeff4
                else
                    intra_ac-existing_coeff4
            }
        }
    }
}

Figure A.1 (continued)
if (block_coded & (! mb_is_intra || (qp < prev_layer_qp))) {
    if (mb_is_intra)
        code_num_new_coeffs_intra()
    else {
        if (prev_layer_num_coeffs(i) == 0)
            code_num_new_coeffs_inter0()
        else if (prev_layer_num_coeffs(i) == 1)
            code_num_new_coeffs_inter1()
        else
            code_num_new_coeffs_inter()
    }
    for (j = 0; j < 64; ++j) {
        if (coeff_is_new(j))
            if (mb_is_intra) {
                intra_newcoeff_run
                if (range_num_quant_bins(prev_layer_intra_range(i,j),qp)
                    == 2)
                    intra_newcoeff_mag2
                else if (range_num_quant_bins(prev_layer_intra_range(i,j),qp)
                    == 3)
                    intra_newcoeff_mag3
                sign
            } else {
                inter_newcoeff_run
                code_inter_newcoeff_type()
                sign
            }
    }
}

Figure A.1 (continued)
other coefficients due to the fact that in MPEG-4 the value of the DC coefficient relative to the surrounding blocks must be known before the scan order for the remaining coefficients can be determined.

2. The morphing of motion vectors in spatial scalability (lines 152-175) uses different symbol types for the one and four vector cases: for the first of these a single symbol type encodes both the sign and magnitude of any change, while for four vectors the sign is coded separately followed by the absolute value of each vector component. Recall from section 6.2.2 that those macroblocks with a single vector can be predicted directly from the corresponding vector in the base layer (where there are four vectors unless they are all so close together that they can be merged into one without producing significant error); the error between the prediction from the base layer and the actual vector in this case is small, this being the error introduced by halving the vector component for use in the base layer. For the four vector case the errors can be significantly larger: each of these vectors is predicted from a single base layer value and for macroblocks with divergent motion the residual after prediction are large. For this reason the single vector case can be coded using a single symbol with a small number of possible values (in most cases the value being coded is either zero or one) whereas the four vector case needs to be able to code large vector components. To do this efficiently without requiring symbols with a very large number of possible values, the coding of this residue is split into two parts.

A.3 Bitstream Semantics

`next_bits()` This function gives access to the bits that are to be read next without removing them from the bitstream (this is identical to the function of the same name described in the MPEG-4 VM).
nextbits_bytealigned() This function gives access to the bits after the next byte boundary without removing them from the bitstream (identical to MPEG-4 VM function).

morph_seq_header() The sequence header for a stream morphing enhancement layer consists of a single (1 bit) flag isvbr which if true (value 1) disables the use of the alternate dquant30 model for coding change in differential quantizer step size as discussed in section 4.3.1. MPEG-4 defines a large number of other parameters that are encoded in the sequence header however almost all of these must be identical in all layers of a stream morphing system and as such do not need to be coded again for stream morphing enhancement layers but can be taken from the base layer (which uses the standard MPEG-4 syntax). For spatial and temporal scalability the only parameters that will change are the frame size (spatial) and timing information (temporal). The following table lists all the sequence header fields:

<table>
<thead>
<tr>
<th>Field</th>
<th>Num. of bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>isvbr</td>
<td>1</td>
</tr>
<tr>
<td>video_object_layer_width</td>
<td>13</td>
</tr>
<tr>
<td>video_object_layer_height</td>
<td>13</td>
</tr>
<tr>
<td>vop_time_increment_resolution</td>
<td>16</td>
</tr>
</tbody>
</table>

Definitions of the last three fields can be found in [54]. At this point the arithmetic coder has not been started and these VLCs are written to the bitstream in the normal way.

morph_vop_header() In a similar way to the sequence header as described above, many of the parameters in each MPEG-4 VOP header can be taken from the base layer. The following table shows those fields that can be different and are coded in the stream morphing VOP header:
### A.3 Bitstream Semantics

<table>
<thead>
<tr>
<th>Field</th>
<th>Num. of bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>vop_coding_type</td>
<td>2</td>
</tr>
<tr>
<td>modulo_time_base</td>
<td>≥ 1</td>
</tr>
<tr>
<td>vop_time_increment</td>
<td>1-16</td>
</tr>
<tr>
<td>vop_rounding_type</td>
<td>1</td>
</tr>
<tr>
<td>vop_quant</td>
<td>5</td>
</tr>
<tr>
<td>vop_fcode_forward</td>
<td>3</td>
</tr>
</tbody>
</table>

All of the fields here have the same coding and semantics of those with the same name in [54]. These VLCs are written directly to the bitstream as the arithmetic coder has not been started at this point.

**start_range_coder()** At this point the range coder is started and symbols can be written into the bitstream. Usually this operation alone does not require any bits to be actually written.

**stop_range_coder()** The state of the range coder is flushed to the bitstream, which in the current implementation [74] writes 32 bits to the stream. Standard VLCs can now be written to the stream; **start_range_coder** must be called before any range-coded symbols are added.

**prev_layer_qp** The quantizer step size in the coincident macroblock in the previous layer.

**isvbr** True if the **isvbr** flag was set in the sequence header (see definition of **morph_seq_header** above).

**istemporal** True if this layer is a temporal scalable layer.

**prevtemporal** True if this frame is **not** temporally-coincident with the previous layer frame. This corresponds to the first frame of the enhancement layer frame pairs shown in Figure 6.5.
prev_frame_mb_is_intra True if the spatially-coincident macroblock in the previous frame was coded in intra mode.

prev_layer_mb_is_intra True if the coincident macroblock in the previous layer layer was coded in intra mode.

passthru True if the current macroblock was coded in intra mode and the coincident previous layer layer macroblock was coded in inter mode or vice versa. This macroblock cannot be morphed and will be passed through the arithmetic coder verbatim via a uniform binary model (section 4.3.2).

mpeg4_passthru_macroblock() The complete contents of the macroblock, in its original MPEG-4 syntax (VLCs) are encoded bit-by-bit using a uniform binary model (section 4.3.2).

mb_is_intra True if the current macroblock mode is intra.

prev_layer_is_intra_ac_pred True if intra AC coefficient prediction was enabled in the coincident macroblock in the previous layer.

prev_layer_mb_skipped True if the coincident macroblock in the previous layer was skipped. Note that the definition of “skipped” here is only that there were no non-zero DCT coefficients; it does not consider motion vectors or quantizer changes as the macroblock skipped bit in standard MPEG-4 does (section 4.3.3).

isspatial True if this is a spatial scalability enhancement layer.

mv_count The number of motion vectors in the macroblock: either zero, one or four.

prev_layer_mv_count The number of motion vectors in the coincident macroblock in the previous layer: either zero, one or four.
next_frame_mv_count The number of motion vectors in the coincident macroblock in the next frame; either zero, one or four. We can determine this without the need to process the next frame by looking at the difference between the motion vectors in the current macroblock and those in the previous layer. If they all differ by the same amount then the next frame must have only one motion vector.

mb_coded True if mb_coded_prev_skipped was present and was true or if the previous layer layer contained non-zero DCT coefficients (such enhancement layer macroblocks are always scanned).

block_count The number of 8x8 blocks in a macroblock: 6 for 4:2:0 format, 8 for 4:2:2 etc.

escape True if the previous coded symbol was an escape code (marked with "(Escape)" in Tables A.4, A.5, A.8, A.9, A.10 A.11, A.17, A.18, A.20 or A.21).

prev_layer_coef_value The value of the coincident (quantized) coefficient in the previous layer.

prev_layer_block_coded True if the $i$th block in the coincident macroblock in the previous layer was coded, i.e. had at least one non-zero DCT coefficient.

qp The quantization step size for the current macroblock

block_is_luminance True if the current block is a luminance block.

prev_layer_quant_texture The quantized DCT coefficient in the previous layer coincident macroblock in block $i$ and coefficient $j$.

prev_frame_quant_texture The quantized DCT coefficient in the previous frame coincident macroblock in block $i$ and coefficient $j$. 
range_num_quant_bins For intra coefficients, the number of quantization bins that the specified range spans for the given quantization step size (section 4.3.6.1).

prev_layer_intra_range For intra coefficients, the range of possible values that the unquantized coefficient \( j \) in block \( i \) has after the coding of the previous layer (section 4.3.6.1).

quant_texture The quantized DCT coefficient in the current layer in block \( i \) and zig-zag position \( j \).

coded_in_some_prev_layer True if a coincident (block \( i \), zig-zag position \( j \)) non-zero coefficient existed in any previous layer. The search for this coefficient does not go below any macroblocks that were passed through due to a mode change (section 4.3.2) or any frame size/rate change associated with spatial or temporal scalability.

prev_layer_num_coeffs The number of non-zero DCT coefficients in block \( i \) of the coincident macroblock in the previous layer.

A.4 Arithmetic Coding Symbols Types

Each arithmetic coded symbol described in the above syntax is associated with a symbol type, the complete list of these types and the semantics of their values are given in Tables A.2-A.25. Some symbol types are used for more than one model. The mapping between symbols themselves and the symbol types listed here is given in the final section.
### Table A.2: Model type `mb_coded_prev_skipped`

<table>
<thead>
<tr>
<th>Index</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Block skipped (not coded)</td>
</tr>
<tr>
<td>1</td>
<td>Block coded</td>
</tr>
</tbody>
</table>

### Table A.3: Model type `block_coded_prev_skipped`

<table>
<thead>
<tr>
<th>Index</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0 new coefficients</td>
</tr>
<tr>
<td>1</td>
<td>1 new coefficient</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>14 new coefficients</td>
</tr>
<tr>
<td>15</td>
<td>15 new coefficients (Escape)</td>
</tr>
</tbody>
</table>

### Table A.4: Model type `num_new_coeffs_intra`

<table>
<thead>
<tr>
<th>Index</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0 new coefficients</td>
</tr>
<tr>
<td>1</td>
<td>1 new coefficient</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>6 new coefficients</td>
</tr>
<tr>
<td>7</td>
<td>7 new coefficients (Escape)</td>
</tr>
</tbody>
</table>

### Table A.5: Model type `num_new_coeffs_inter`

<table>
<thead>
<tr>
<th>Index</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Intra AC prediction disabled</td>
</tr>
<tr>
<td>1</td>
<td>Intra AC prediction enabled</td>
</tr>
</tbody>
</table>

### Table A.6: Model type `intra_ac_pred`
### A.4 Arithmetic Coding Symbols Types

<table>
<thead>
<tr>
<th>Index</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Same mode as previous layer MB</td>
</tr>
<tr>
<td>1</td>
<td>Change to INTER mode</td>
</tr>
<tr>
<td>2</td>
<td>Change to INTER (8x8 motion vectors) mode</td>
</tr>
<tr>
<td>3</td>
<td>Change to INTRA mode</td>
</tr>
</tbody>
</table>

Table A.7: Model type change_mode

<table>
<thead>
<tr>
<th>Index</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Same value as previous layer</td>
</tr>
<tr>
<td>1</td>
<td>Value refined by 1 (e.g. $1 \rightarrow 2$ or $-2 \rightarrow -3$)</td>
</tr>
<tr>
<td>2</td>
<td>Value contracted by 1 (e.g. $2 \rightarrow 1$ or $-3 \rightarrow -2$)</td>
</tr>
<tr>
<td>3</td>
<td>Value is zero</td>
</tr>
<tr>
<td>4</td>
<td>Value refined by more than 1 (Escape)</td>
</tr>
<tr>
<td>5</td>
<td>Value contracted by more than 1 (Escape)</td>
</tr>
</tbody>
</table>

Table A.8: Model type inter_existent_coeff

<table>
<thead>
<tr>
<th>Index</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Coefficient is (still) zero</td>
</tr>
<tr>
<td>1</td>
<td>Coefficient restored with same value</td>
</tr>
<tr>
<td>2</td>
<td>Coefficient restored and refined (Escape)</td>
</tr>
<tr>
<td>3</td>
<td>Coefficient restored and contracted (Escape)</td>
</tr>
</tbody>
</table>

Table A.9: Model type inter_restored

<table>
<thead>
<tr>
<th>Index</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Escape sequence complete</td>
</tr>
<tr>
<td>1</td>
<td>Refine/contract by 1 (Escape)</td>
</tr>
</tbody>
</table>

Table A.10: Model type refine_contract_escape
### Index Semantics

<table>
<thead>
<tr>
<th>Index</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>New coefficient has value $\pm 1$</td>
</tr>
<tr>
<td>1</td>
<td>New coefficient has value $\pm 2$</td>
</tr>
<tr>
<td>2</td>
<td>New coefficient has value $&gt;2$ or $&lt;-2$ (Escape)</td>
</tr>
</tbody>
</table>

Table A.11: Model type `inter_newcoeff_type`

<table>
<thead>
<tr>
<th>Index</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>DQUANT is -2</td>
</tr>
<tr>
<td>1</td>
<td>DQUANT is -1</td>
</tr>
<tr>
<td>2</td>
<td>DQUANT is 0</td>
</tr>
<tr>
<td>3</td>
<td>DQUANT is 1</td>
</tr>
<tr>
<td>4</td>
<td>DQUANT is 2</td>
</tr>
</tbody>
</table>

Table A.12: Model type `dquant`

<table>
<thead>
<tr>
<th>Index</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Next coefficient is non-zero</td>
</tr>
<tr>
<td>1</td>
<td>One zero before non-zero coefficient</td>
</tr>
<tr>
<td>62</td>
<td>62 zero coefficients before non-zero coefficient</td>
</tr>
<tr>
<td>63</td>
<td>63 zero coefficients before non-zero coefficient</td>
</tr>
</tbody>
</table>

Table A.13: Model type `newcoeff_run`

<table>
<thead>
<tr>
<th>Index</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Quantization interval closest to zero</td>
</tr>
<tr>
<td>1</td>
<td>Quantization interval away from zero</td>
</tr>
</tbody>
</table>

Table A.14: Model type `intra_coeff_mag2`
### A.4 Arithmetic Coding Symbols Types

<table>
<thead>
<tr>
<th>Index</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Quantization interval closest to zero</td>
</tr>
<tr>
<td>1</td>
<td>Middle quantization interval</td>
</tr>
<tr>
<td>2</td>
<td>Quantization interval furthest away from zero</td>
</tr>
</tbody>
</table>

Table A.15: Model type `intra_coeff_mag3`

<table>
<thead>
<tr>
<th>Index</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Quantization interval closest to zero</td>
</tr>
<tr>
<td>1</td>
<td>Second quantization interval</td>
</tr>
<tr>
<td>2</td>
<td>Third quantization interval</td>
</tr>
<tr>
<td>3</td>
<td>Quantization interval furthest away from zero</td>
</tr>
</tbody>
</table>

Table A.16: Model type `intra_coeff_mag4`

<table>
<thead>
<tr>
<th>Index</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Motion vector component difference is -2.0 (Escape)</td>
</tr>
<tr>
<td>1</td>
<td>Motion vector component difference is -1.5</td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
<tr>
<td>4</td>
<td>Motion vector component difference is 0</td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
<tr>
<td>7</td>
<td>Motion vector component difference is 1.5</td>
</tr>
<tr>
<td>8</td>
<td>Motion vector component difference is 2.0 (Escape)</td>
</tr>
</tbody>
</table>

Table A.17: Model type `spatial16_mvdiff`

<table>
<thead>
<tr>
<th>Index</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Motion vector component difference is 0</td>
</tr>
<tr>
<td>1</td>
<td>Motion vector component difference is 0.5</td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
<tr>
<td>7</td>
<td>Motion vector component difference is 3.5</td>
</tr>
<tr>
<td>8</td>
<td>Motion vector component difference is 4.0 (Escape)</td>
</tr>
</tbody>
</table>

Table A.18: Model type `mv_diff_abs`
A.4 Arithmetic Coding Symbols Types

### Table A.19: Model type `temporal_mvdiff_sign`

<table>
<thead>
<tr>
<th>Index</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Value is less than same value in previous frame (Escape)</td>
</tr>
<tr>
<td>1</td>
<td>Value is equal to same value in previous frame</td>
</tr>
<tr>
<td>2</td>
<td>Value is greater than same value in previous frame (Escape)</td>
</tr>
<tr>
<td>3</td>
<td>Value is less than same value in previous layer (Escape)</td>
</tr>
<tr>
<td>4</td>
<td>Value is equal to same value in previous layer</td>
</tr>
<tr>
<td>5</td>
<td>Value is greater than same value in previous layer (Escape)</td>
</tr>
</tbody>
</table>

### Table A.20: Model type `temporal_existing_diff`

<table>
<thead>
<tr>
<th>Index</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Differential value is 1</td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Differential value is 4</td>
</tr>
<tr>
<td>4</td>
<td>Differential value is $&gt;4$ (Escape)</td>
</tr>
</tbody>
</table>

### Table A.21: Model type `temporal_existing_mag`

<table>
<thead>
<tr>
<th>Index</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>There are four motion vectors in this macroblock in the next frame</td>
</tr>
<tr>
<td>1</td>
<td>MB in next frame has one vector: use base vector 1 for prediction</td>
</tr>
<tr>
<td>2</td>
<td>MB in next frame has one vector: use base vector 2 for prediction</td>
</tr>
<tr>
<td>3</td>
<td>MB in next frame has one vector: use base vector 3 for prediction</td>
</tr>
<tr>
<td>4</td>
<td>MB in next frame has one vector: use base vector 4 for prediction</td>
</tr>
</tbody>
</table>

### Table A.22: Model type `temporal_nextframe8x8`
### Table A.23: Model type `temporal_nextframe16x16`

<table>
<thead>
<tr>
<th>Index</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Use motion vector from base layer vector 1 for prediction</td>
</tr>
<tr>
<td>1</td>
<td>Use motion vector from base layer vector 2 for prediction</td>
</tr>
<tr>
<td>2</td>
<td>Use motion vector from base layer vector 3 for prediction</td>
</tr>
<tr>
<td>3</td>
<td>Use motion vector from base layer vector 4 for prediction</td>
</tr>
</tbody>
</table>

### Table A.24: Model type `sign`

<table>
<thead>
<tr>
<th>Index</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Sign is positive</td>
</tr>
<tr>
<td>1</td>
<td>Sign is negative</td>
</tr>
</tbody>
</table>

### Table A.25: Model type `raw_bits`

<table>
<thead>
<tr>
<th>Index</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Bit value is 0</td>
</tr>
<tr>
<td>1</td>
<td>Bit value is 1</td>
</tr>
</tbody>
</table>
Table A.26: Complete model list

<table>
<thead>
<tr>
<th>Name</th>
<th>Model Type</th>
<th>Table Ref</th>
<th># of models, factors affecting model choice</th>
</tr>
</thead>
<tbody>
<tr>
<td>mb_coded_prev_skipped</td>
<td>mb_coded_prev_skipped</td>
<td>A.2</td>
<td>(20 = 5 \times (D_{rate})^3)</td>
</tr>
<tr>
<td>block_coded_prev_skipped</td>
<td>block_coded_prev_skipped</td>
<td>A.3</td>
<td>(20 = 5 \times (D_{rate})^3)</td>
</tr>
<tr>
<td>block_coded_prev_skipped_chrom</td>
<td>block_coded_prev_skipped</td>
<td>A.3</td>
<td>(20 = 5 \times (D_{rate})^3)</td>
</tr>
<tr>
<td>dquant30</td>
<td>dquant</td>
<td>A.12</td>
<td>1</td>
</tr>
<tr>
<td>dquant</td>
<td>dquant</td>
<td>A.12</td>
<td>(5 \times \text{dquant value in previous layer: from -2 to 2 inclusive})</td>
</tr>
<tr>
<td>change_mode</td>
<td>change_mode</td>
<td>A.7</td>
<td>((D_{rate})^3)</td>
</tr>
<tr>
<td>change_mode_inter</td>
<td>change_mode</td>
<td>A.7</td>
<td>((D_{rate})^3)</td>
</tr>
<tr>
<td>intra_ac_pred_prev</td>
<td>intra_ac_pred</td>
<td>A.6</td>
<td>((D_{rate})^3)</td>
</tr>
<tr>
<td>intra_ac_pred_no_prev</td>
<td>intra_ac_pred</td>
<td>A.6</td>
<td>((D_{rate})^3)</td>
</tr>
<tr>
<td>inter_restored</td>
<td>inter_restored</td>
<td>A.9</td>
<td>((D_{qp})^3)</td>
</tr>
<tr>
<td>refine_contract_escape</td>
<td>refine_contract_escape</td>
<td>A.10</td>
<td>1</td>
</tr>
<tr>
<td>inter_existing_coeff</td>
<td>inter_existing_coeff</td>
<td>A.8</td>
<td>((D_{qp})^3)</td>
</tr>
<tr>
<td>inter_existing_coeff2</td>
<td>inter_existing_coeff</td>
<td>A.8</td>
<td>((D_{qp})^3)</td>
</tr>
<tr>
<td>num_new_coeffs_inter</td>
<td>num_new_coeffs_inter</td>
<td>A.4</td>
<td>((D_{rate})^3)</td>
</tr>
<tr>
<td>num_new_coeffs_inter_base_0</td>
<td>num_new_coeffs_inter</td>
<td>A.5</td>
<td>((D_{rate})^3)</td>
</tr>
<tr>
<td>num_new_coeffs_inter_base_1</td>
<td>num_new_coeffs_inter</td>
<td>A.5</td>
<td>((D_{rate})^3)</td>
</tr>
<tr>
<td>inter_newcoeff_type</td>
<td>inter_newcoeff_type</td>
<td>A.11</td>
<td>1</td>
</tr>
<tr>
<td>intra_dc_existing_coeff2</td>
<td>intra_coeff_mag2</td>
<td>A.14</td>
<td>6 (see section 4.3.6.1)</td>
</tr>
<tr>
<td>intra_dc_existing_coeff3</td>
<td>intra_coeff_mag3</td>
<td>A.15</td>
<td>4 (see section 4.3.6.1)</td>
</tr>
<tr>
<td>intra_dc_existing_coeff4</td>
<td>intra_coeff_mag4</td>
<td>A.16</td>
<td>2 (see section 4.3.6.1)</td>
</tr>
<tr>
<td>intra_ac_existing_coeff2</td>
<td>intra_coeff_mag2</td>
<td>A.14</td>
<td>6 (see section 4.3.6.1)</td>
</tr>
<tr>
<td>intra_ac_existing_coeff3</td>
<td>intra_coeff_mag3</td>
<td>A.15</td>
<td>4 (see section 4.3.6.1)</td>
</tr>
<tr>
<td>intra_ac_existing_coeff4</td>
<td>intra_coeff_mag4</td>
<td>A.16</td>
<td>2 (see section 4.3.6.1)</td>
</tr>
<tr>
<td>inter_newcoeff_run</td>
<td>newcoeff_run</td>
<td>A.13</td>
<td>1</td>
</tr>
<tr>
<td>inter_newcoeff_run</td>
<td>newcoeff_run</td>
<td>A.13</td>
<td>1</td>
</tr>
<tr>
<td>intra_newcoeff_mag2</td>
<td>intra_coeff_mag2</td>
<td>A.14</td>
<td>2 (two quantization regions)</td>
</tr>
<tr>
<td>intra_newcoeff_mag3</td>
<td>intra_coeff_mag3</td>
<td>A.15</td>
<td>3 (three quantization regions)</td>
</tr>
</tbody>
</table>

A.5 Arithmetic Coding Symbol Reference

Table A.26 lists all the probability models described in the syntax shown in Figure A.1 and matches them to the symbol types described in the previous section. Many symbols are coded using one of a number of different possible models which exploit the different statistics of the symbol under different conditions as was discussed in Chapter 4. Table A.26 lists the total number of models for each symbol along with a description of the factors that affect the selection of the model used. Often there is more than one factor in which case the number of total models is the product of the number of alternatives due to

\footnote{Symbols that depend on \(D_{qp}\) or \(D_{rate}\) use separate models for SNR, spatial and temporal scalability, see sections 6.2.3 and 6.3.3.}
<table>
<thead>
<tr>
<th>Name</th>
<th>Model Type</th>
<th>Table Ref</th>
<th># of models, factors affecting model choice</th>
</tr>
</thead>
<tbody>
<tr>
<td>spatial_mvdiff16_nonzero</td>
<td>spatial16_mvdiff</td>
<td>A.17</td>
<td>1</td>
</tr>
<tr>
<td>spatial_mvdiff16_zero</td>
<td>spatial16_mvdiff</td>
<td>A.17</td>
<td>1</td>
</tr>
<tr>
<td>spatial_mvdiff8_nonzero</td>
<td>mv_diff_abs</td>
<td>A.18</td>
<td>1</td>
</tr>
<tr>
<td>spatial_mvdiff8_zero</td>
<td>mv_diff_abs</td>
<td>A.18</td>
<td>1</td>
</tr>
<tr>
<td>spatial_mvdiff_sign</td>
<td>sign</td>
<td>A.24</td>
<td>1</td>
</tr>
<tr>
<td>temporal_mode_same_intra</td>
<td>change_mode</td>
<td>A.7</td>
<td>1</td>
</tr>
<tr>
<td>temporal_mode_same_inter</td>
<td>change_mode</td>
<td>A.7</td>
<td>1</td>
</tr>
<tr>
<td>temporal_mode_different</td>
<td>change_mode</td>
<td>A.7</td>
<td>1</td>
</tr>
<tr>
<td>temporal_mvdiff_mag</td>
<td>mv_diff_abs</td>
<td>A.18</td>
<td>5 (magnitude of base layer vector: ≤ 2, ≤ 6, ≤ 12, ≤ 24 or &gt; 24)</td>
</tr>
<tr>
<td>temporal_mb_coded</td>
<td>mb_coded_prev_skipped</td>
<td>A.2</td>
<td>4 (all combinations of skipped/not skipped in the two reference macroblocks)</td>
</tr>
<tr>
<td>temporal_block_coded</td>
<td>block_coded_prev_skipped</td>
<td>A.3</td>
<td>4 (all combinations of coded/not coded in the two reference blocks)</td>
</tr>
<tr>
<td>temporal_intra_ac_pred</td>
<td>intra_ac_pred</td>
<td>A.6</td>
<td>4 (all combinations of enabled/disabled in the two reference macroblocks)</td>
</tr>
<tr>
<td>temporal_existing_intra_same</td>
<td>inter_existing_coeff</td>
<td>A.8</td>
<td>1</td>
</tr>
<tr>
<td>temporal_existing_intra_diff</td>
<td>temporal_existing_diff</td>
<td>A.20</td>
<td>1</td>
</tr>
<tr>
<td>temporal_existing_intra_mag</td>
<td>temporal_existing_mag</td>
<td>A.21</td>
<td>1</td>
</tr>
<tr>
<td>temporal_existing_inter_same</td>
<td>inter_existing_coeff</td>
<td>A.8</td>
<td>1</td>
</tr>
<tr>
<td>temporal_existing_inter_diff</td>
<td>temporal_existing_diff</td>
<td>A.20</td>
<td>1</td>
</tr>
<tr>
<td>temporal_nextframe_8x8</td>
<td>temporal_nextframe8x8</td>
<td>A.22</td>
<td>1</td>
</tr>
<tr>
<td>temporal_nextframe_16x16</td>
<td>temporal_nextframe16x16</td>
<td>A.23</td>
<td>1</td>
</tr>
<tr>
<td>sign</td>
<td>sign</td>
<td>A.24</td>
<td>1</td>
</tr>
</tbody>
</table>

Figure A.26 (continued)

each factor alone, for example whether a macroblock is skipped or not skipped is defined to be dependent upon the number of surrounding macroblocks coded (for causal macroblocks this is a number from 0 to 4 or five models in total) and the value of $D_{rate}$ as discussed in section 4.3.7 which has four possible values giving $5 \times 4 = 20$ models in total.
Appendix B

Test Sequences

The video sequences used in this thesis are mostly drawn from the set of test sequences used in the MPEG-4 standardization process [54]. Each sequence is 10 seconds long and are available in a variety of sizes and frame rates. Figure B.1 shows two sample frames from each of the five sequences which are described in more detail below:

**Akiyo** is a very-low-motion sequence depicting a news reader. Apart from head movement the rest of each frame is static although there is a small amount of movement of the upper body throughout the sequence.

**Carphone** shows a man talking in the back seat of a car. The early part of the sequence has a relative small amount of motion with the background visible out of the car window being distant and not moving very quickly. The later part of the sequence has the car passing near some trees which creates a lot of very fast moving texture out of the window. Some vibration can also be seen which moves the interior of the car relative to the camera.

**Coastguard** is a high-motion sequence that is dominated by a large amount of continuously-moving water in the bottom two-thirds of the picture. The initial phase of the sequence involves the camera panning to the left in following a small boat. About one quarter of the way through the sequence the camera switches to follow another larger boat that is moving in the opposite direction.
Figure B.1: Example frames from sequences (CIF 10fps)
Figure B.1 (continued)
Foreman is another high-motion sequence which initially starts with a close-up view of a man talking. Just after the half-way point in the sequence the camera pans quickly to show a building site and some distant trees. The camera is held unsteadily throughout the sequence.

Mother & Daughter is a low- to medium-motion sequence. While the girl shown in the left-hand side of the frame is relatively static over the entire sequence, her mother on the right-hand side is moving her head constantly and there is some movement of her arm in the sequence also.

Apart from the initial stream morphing tests in Chapter 4, the other experiments conducted are done using only the “Mother & Daughter” and “Foreman” sequences, to limit the volume of data that must be displayed. The performance of scalable video systems depends significantly on the presence or otherwise of prediction in the enhancement layer(s). The coding of low-motion sequences benefits from enhancement layer prediction which is not so important for high-motion content, where much of the texture is changing and cannot be predicted. “Mother & Daughter” and “Foreman” are used as examples of sequences at opposite ends of this spectrum of possible inputs.
Appendix C

Subjective Tests

C.1 Test Setup

The sequences to be used were first upsampled by a factor of two from CIF to Rec. 601 size then loaded onto an Abekas A66 digital video disk recorder. As the A66 can only store short (90 seconds maximum) segments of video, the results were then recorded onto D1 tape in pieces. For display, the output of the D1 was transmitted via Serial Digital Interface (SDI) connection to a Sony BVM-2016P monitor. As such, the path from the generated sequences to final display is completely digital and without any compression that might add further artifacts to the material. For a display of this size (48cm diagonal, 29cm height) Recommendation 500 specifies that viewers should be positioned at a distance of seven times the height of the screen (just over 2m). A maximum of three viewers was tested at any one time. The level of reflected light from the white background surrounding the display was adjusted to be approximately 0.15 times the maximum luminance of the display (all white) as specified by the Recommendation.
C.2 Tape Contents

Table C.1 shows the set of 30 source sequences (each 10s long, CIF 10fps) that were used to create the subjective tests, copies of which can be found on the accompanying CD. The "Sequence" column in the table is one of the following:

- **SM** Stream morphing (5 layers total)
- **SNR** MPEG-2-compliant SNR scalability (5 layers total)
- **FGS** MPEG-4 FGS
- **SNRn** Same as SNR with enhancement layer rates adjusted (normalized) to approximately match the top layer PSNR to that of stream morphing. This is the "same PSNR" test condition described in Chapter 5.
- **FGSn** "Same PSNR" test for MPEG-4 FGS.

Rec. 500 suggests that test sessions last no longer than 30 minutes and given each presentation (A, B then A and B again with appropriate spaces in between plus time at the end to mark the scoresheet) takes around 1 minute it is not possible to test all combinations of these test conditions for the four sequences. The following six combinations (out of the possible 10) were used: SM vs SNR, SM vs FGS, SNR vs FGS, SM vs SNRn and SM vs FGSn. The third of these is used for the results in Chapter 3, the remainder are shown in Chapter 5. Since we are primarily interested in testing stream morphing, the remaining combinations are not especially useful in any case. The order of these 24 tests is shown in Table C.2. These have been randomized so that the same source sequence is never used in successive presentations to prevent any "memory" effects from the previous presentation. To give new users time to adjust to the subjective test process, Rec. 500 suggests an initial training period for which any results collected are not used in the subsequent analysis. Here we use the first four presentations for training and present them again at the end after the
main body of 24 tests is complete (these results are then used in place of the first four discarded results).

C.3 Analysis of Results

Results were obtained with 19 non-expert observers, the first stage of the analysis was to measure the locations of the marks on their scoresheets and to enter these into the computer (included on the CD), normalize the distances measured into a range of 0-100 and to compute the difference score for each presentation (and each viewer).

Each viewer's notion of "Excellent", "Good" etc. video quality as shown on the scoresheet (Figure 3.18) have not been "calibrated" to be in any way consistent with each other. We now adjust the difference scores to ensure those observers whose difference scores are generally small are equally represented in the final mean value for each test. Table C.3 shows the standard deviation $S_i$ of all difference scores for each of the 19 observers. These values range over more than a factor of 2 so if the mean of the subjective difference values were to be taken on the original data then observer 9 would have a much greater impact on the final result than observer 18. We adjust the difference scores $D_{ij}$ for observer $i$ and presentation $j$ by equalizing the standard deviations across all observers to form a corrected score $D_{ij}^*$:

$$D_{ij}^* = D_{ij} \times \frac{\bar{S}}{S_i}$$  \hspace{1cm} (C.1)

where $\bar{S} = \frac{1}{N} \sum_{i=1}^{N} S_i$ is the average standard deviation across all $N$ observers. Note that this adjustment procedure is not part of Rec. 500 however we feel this is necessary otherwise the final results will be disproportionately influenced by those observers whose gradings are spread over a larger range e.g. observer 9 in Table C.3.

Rec. 500 then specifies a screening procedure which counts the number of times any particular observer's result for a given test is located more than two
## C.3 Analysis of Results

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Technique</th>
<th>Base Layer Rate (kbps)</th>
<th>Total Enh. Rate (kbps)</th>
<th>Top Layer PSNR (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Akiyo</td>
<td>SM</td>
<td>32</td>
<td>64</td>
<td>36.74</td>
</tr>
<tr>
<td></td>
<td>SNR</td>
<td>32</td>
<td>64</td>
<td>35.38</td>
</tr>
<tr>
<td></td>
<td>FGS</td>
<td>32</td>
<td>64</td>
<td>32.74</td>
</tr>
<tr>
<td></td>
<td>SNRn</td>
<td>32</td>
<td>101</td>
<td>36.76</td>
</tr>
<tr>
<td></td>
<td>FGSn</td>
<td>32</td>
<td>266</td>
<td>36.74</td>
</tr>
<tr>
<td>Mother &amp; Daughter</td>
<td>SM</td>
<td>56</td>
<td>112</td>
<td>37.57</td>
</tr>
<tr>
<td></td>
<td>SNR</td>
<td>56</td>
<td>112</td>
<td>36.96</td>
</tr>
<tr>
<td></td>
<td>FGS</td>
<td>56</td>
<td>112</td>
<td>35.71</td>
</tr>
<tr>
<td></td>
<td>SNRn</td>
<td>56</td>
<td>146</td>
<td>37.56</td>
</tr>
<tr>
<td></td>
<td>FGSn</td>
<td>56</td>
<td>216</td>
<td>37.56</td>
</tr>
<tr>
<td>Foreman</td>
<td>SM</td>
<td>130</td>
<td>260</td>
<td>33.08</td>
</tr>
<tr>
<td></td>
<td>SNR</td>
<td>130</td>
<td>260</td>
<td>32.85</td>
</tr>
<tr>
<td></td>
<td>FGS</td>
<td>130</td>
<td>260</td>
<td>32.93</td>
</tr>
<tr>
<td></td>
<td>SNRn</td>
<td>130</td>
<td>285</td>
<td>33.10</td>
</tr>
<tr>
<td></td>
<td>FGSn</td>
<td>130</td>
<td>276</td>
<td>33.07</td>
</tr>
<tr>
<td>Carphone</td>
<td>SM</td>
<td>100</td>
<td>200</td>
<td>34.04</td>
</tr>
<tr>
<td></td>
<td>SNR</td>
<td>100</td>
<td>200</td>
<td>33.68</td>
</tr>
<tr>
<td></td>
<td>FGS</td>
<td>100</td>
<td>200</td>
<td>33.99</td>
</tr>
<tr>
<td></td>
<td>SNRn</td>
<td>100</td>
<td>227</td>
<td>34.05</td>
</tr>
<tr>
<td></td>
<td>FGSn</td>
<td>100</td>
<td>205</td>
<td>34.03</td>
</tr>
</tbody>
</table>

Table C.1: Source sequences for subjective tests
## C.3 Analysis of Results

<table>
<thead>
<tr>
<th>Number</th>
<th>Sequence</th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Foreman</td>
<td>SNRn</td>
<td>FGSn</td>
</tr>
<tr>
<td>2</td>
<td>Carphone</td>
<td>SNRn</td>
<td>FGSn</td>
</tr>
<tr>
<td>3</td>
<td>Akiyo</td>
<td>SM</td>
<td>FGS</td>
</tr>
<tr>
<td>4</td>
<td>Mother &amp; Daughter</td>
<td>FGS</td>
<td>SM</td>
</tr>
<tr>
<td>5</td>
<td>Foreman</td>
<td>SM</td>
<td>FGSn</td>
</tr>
<tr>
<td>6</td>
<td>Akiyo</td>
<td>FGSn</td>
<td>SM</td>
</tr>
<tr>
<td>7</td>
<td>Foreman</td>
<td>SNR</td>
<td>FGS</td>
</tr>
<tr>
<td>8</td>
<td>Carphone</td>
<td>SM</td>
<td>FGS</td>
</tr>
<tr>
<td>9</td>
<td>Akiyo</td>
<td>SNRn</td>
<td>SM</td>
</tr>
<tr>
<td>10</td>
<td>Carphone</td>
<td>SM</td>
<td>SNRn</td>
</tr>
<tr>
<td>11</td>
<td>Mother &amp; Daughter</td>
<td>SM</td>
<td>SNR</td>
</tr>
<tr>
<td>12</td>
<td>Carphone</td>
<td>SNR</td>
<td>SM</td>
</tr>
<tr>
<td>13</td>
<td>Foreman</td>
<td>SNRn</td>
<td>SM</td>
</tr>
<tr>
<td>14</td>
<td>Akiyo</td>
<td>SNR</td>
<td>FGS</td>
</tr>
<tr>
<td>15</td>
<td>Foreman</td>
<td>SM</td>
<td>FGS</td>
</tr>
<tr>
<td>16</td>
<td>Carphone</td>
<td>SM</td>
<td>FGSn</td>
</tr>
<tr>
<td>17</td>
<td>Akiyo</td>
<td>FGSn</td>
<td>SNRn</td>
</tr>
<tr>
<td>18</td>
<td>Mother &amp; Daughter</td>
<td>FGSn</td>
<td>SM</td>
</tr>
<tr>
<td>19</td>
<td>Akiyo</td>
<td>SM</td>
<td>SNR</td>
</tr>
<tr>
<td>20</td>
<td>Mother &amp; Daughter</td>
<td>SNRn</td>
<td>FGSn</td>
</tr>
<tr>
<td>21</td>
<td>Foreman</td>
<td>SNR</td>
<td>SM</td>
</tr>
<tr>
<td>22</td>
<td>Mother &amp; Daughter</td>
<td>FGS</td>
<td>SNR</td>
</tr>
<tr>
<td>23</td>
<td>Carphone</td>
<td>SNR</td>
<td>FGS</td>
</tr>
<tr>
<td>24</td>
<td>Mother &amp; Daughter</td>
<td>SM</td>
<td>SNRn</td>
</tr>
</tbody>
</table>

Table C.2: Subjective test presentations
Table C.3: Standard deviation of subjective difference scores for each observer

<table>
<thead>
<tr>
<th>Observer</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Observer</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
</tr>
<tr>
<td>Std. Deviation</td>
<td>10.58</td>
<td>20.34</td>
<td>10.59</td>
<td>15.81</td>
<td>10.65</td>
<td>11.76</td>
<td>15.98</td>
</tr>
<tr>
<td>Observer</td>
<td>15</td>
<td>16</td>
<td>17</td>
<td>18</td>
<td>19</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Std. Deviation</td>
<td>21.49</td>
<td>12.11</td>
<td>8.39</td>
<td>8.13</td>
<td>9.15</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

standard deviations away from the mean score. If this occurs more than a set
number of times the results for that observer should not be included in the final
analysis. Complete details of this procedure can be found in [32], a script to
perform this analysis is included on the CD. All of the 19 observers passed this
test.

Once the data has been adjusted using equation C.1 the mean value for each
presentation can be taken:

\[ \overline{D_j^*} = \frac{1}{N} \sum_{i=1}^{N} D_{ij}^* \]  \hspace{1cm} (C.2)

This represents the difference value that would be registered by a hypothet-
ical "average" observer. The confidence interval for each of these mean values
\( \delta_j \) is:

\[ \delta_j = 1.96 \frac{S_j}{\sqrt{N}} \]  \hspace{1cm} (C.3)

where \( S_j \) is the standard deviation of the results for each presentation (previ-
ously \( S_i \) was the standard deviation across the results for each viewer):

\[ S_j = \sqrt{\frac{\sum_{i=1}^{N} (D_j^* - D_{ij}^*)^2}{N - 1}} \]  \hspace{1cm} (C.4)

The 95% confidence intervals shown in the subjective test results in Chap-
ters 3 and 5 are the range:

\[ [\overline{D_j^*} - \delta_j, \overline{D_j^*} + \delta_j] \]  \hspace{1cm} (C.5)
C.4 Alternative Analysis

To show that the results shown so far have not been significantly affected by lack of consistency between scores from different observers (even after correction) we will look at the results again without considering the “magnitude” of the difference scores registered. Each result will be classified into one of three categories as was done in Chapters 3 and 5: sequence A was better quality than sequence B, the two sequences were indistinguishable (within 2mm on the scoresheet) or B was better than A. We can perform the same analysis as before to generate a result for the “average” observer using equal scores of -1, 0 and +1 for all results in each of the three categories. The central limit theorem states that for a large enough sample a normal distribution will approximate the distribution of the mean score.

Table C.4 shows the confidence with which we can say that the MPEG-2-compliant SNR scalability is subjectively superior to MPEG-4 FGS. This is calculated in the same way as Table 3.7 except that the three categories with scores of -1,0,+1 were used. Similarly, for the subjective test results shown in Table 5.3 the probability that the average user will find the listed technique to be worse
than stream morphing (the reference condition) have been re-calculated and displayed in Table C.5.

This alternative analysis strongly confirms the original conclusion that stream morphing is subjectively superior to MPEG-2-compliant SNR scalability and MPEG-4 FGS for all the sequences tested at identical bit rates. The only significant difference is for the comparison between stream morphing and "same PSNR" MPEG-4 FGS in the case of "Akiyo" (recall that the enhancement bit layer for the FGS case here is more than three times greater than for stream morphing) where Table 5.2 did indicate that more users felt FGS was the better technique in this case. Those viewers who felt stream morphing was better contributed more to the mean score which lead to the original result that this was likely to be the judgement of the "average" observer. As the results here are so close we should therefore conclude that in this case there is no significant difference between the two sequences.
Scalable video coding by stream morphing / thesis 2002 Macnicol BARCODE: 361928
SCALABLE VIDEO CODING
BY STREAM MORPHING

JAMES MACNICOL (ver 2)