Efficient Block Quantisation for Image and Speech Coding

Stephen So, BEng (Hons)

School of Microelectronic Engineering

Faculty of Engineering and Information Technology

Griffith University, Brisbane, Australia

March 2005

Dissertation submitted in fulfillment

of the requirements of the degree of

Doctor of Philosophy

Efficient Block Quantisation for Image and Speech Coding

GUID:	777386
Date of Submission:	1 March 2005
Principal Supervisor:	Professor Kuldip K. Paliwal
Associate Supervisor:	Professor Yoshinori Sagisaka

Abstract

Signal coding or compression has played a significant role in the success of digital communications and multimedia. The use of signal coding pervades many aspects of our digital lifestyle—a lifestyle that has seen widespread demand for applications like third generation mobile telephony, portable music players, Internet-based video conferencing, digital television, etc. The issues that arise, when dealing with the transmission and storage of digital media, are the limited bandwidth of communication channels, the limited capacity of storage devices, and the limited processing ability of the encoding and decoding devices. The aim of signal coding is therefore to represent digital media, such as speech, music, images, and video, as efficiently as possible. Coding efficiency encompasses rate-distortion (for lossy coding), computational complexity, and static memory requirements.

The fundamental operation in lossy signal coding is quantisation. Its rate-distortion efficiency is influenced by the properties of the signal source, such as statistical dependencies and its probability density function. Vector quantisers are known to theoretically achieve the lowest distortion, at a given rate and dimension, of any quantisation scheme, though their computational complexity and memory requirements grow exponentially with rate and dimension. Structurally constrained vector quantisers, such as product code vector quantisers, alleviate these complexity issues, though this is achieved at the cost of degraded rate-distortion performance.

Block quantisers or transform coders, which are a special case of product code vector quantisation, possess both low computational and memory requirements, as well as the ability to scale to any bitrate, which is termed as bitrate scalability. However, the prerequisite for optimal block quantisation, namely a purely Gaussian data source with uniform correlation, is rarely ever met with real-world signals. The Gaussian mixture model-based block quantiser, which was originally developed for line spectral frequency (LSF) quantisation for speech coding, overcomes these problems of source mismatch and non-stationarity by estimating the source using a GMM.

The split vector quantiser, which was also successfully applied to LSF quantisation in the speech coding literature, is a product code vector quantiser that overcomes the complexity problem of unconstrained vector quantisers, by partitioning vectors into subvectors and quantising each one independently. The complexity can be significant reduced via more vector splitting, though this inevitably leads to an accompanying degradation in the rate-distortion efficiency. This is because the structural constraint of vector splitting causes losses in several properties of vector quantisers, which are termed as 'advantages'.

This dissertation makes several contributions to the area of block and vector quantisation, more specifically to the GMM-based block quantiser and split vector quantiser, which aim to improve their rate-distortion and computational efficiency. These new quantisation schemes are evaluated and compared with existing and popular schemes in the areas of lossy image coding, LSF quantisation in narrowband speech coding, LSF and immittance spectral pair (ISP) quantisation in wideband speech coding, and Mel frequency-warped cepstral coefficient (MFCC) quantisation in distributed speech recognition. These contributions are summarised below.

A novel technique for encoding fractional bits in a fixed-rate GMM-based block quantiser scheme is presented. In the GMM-based block quantiser, fractional bitrates are often assigned to each of the cluster block quantisers. This new encoding technique leads to better utilisation of the bit budget by allowing the use of, and providing for the encoding of, quantiser levels in a fixed-rate framework. The algorithm is based on a generalised positional number system and has a low complexity.

A lower complexity GMM-based block quantiser, that replaces the KLT with the discrete cosine transform (DCT), is proposed for image coding. Due to its source independent nature and amenability to efficient implementation, the DCT allows a fast GMM-based block quantiser to be realised that achieves comparable rate-distortion performance as the KLT-based scheme in the block quantisation of images.

Transform image coding often suffers from block artifacts at relatively low bitrates. We propose a scheme that minimises the block artifacts of block quantisation by pre-processing the image using the discrete wavelet transform, extracting vectors via a tree structure that exploits spatial self-similarity, and quantising these vectors using the GMM-based block quantiser. Visual examination shows that block artifacts are considerably reduced by the wavelet pre-processing step.

The multi-frame GMM-based block quantiser is a modified scheme that exploits memory across successive frames or vectors. Its main advantages over the memoryless scheme in the application of LSF and ISP quantisation, are better rate-distortion and computational efficiency, through the exploitation of correlation across multiple frames and mean squared error selection criterion, respectively. The multi-frame GMM-based block quantiser is also evaluated for the quantisation of MFCC feature vectors for distributed speech recognition and is shown to be superior to all quantisation schemes considered.

A new product code vector quantiser, called the switched split vector quantiser (SSVQ), is proposed for speech LSF and ISP quantisation. SSVQ is a hybrid scheme, combining a switch vector quantiser with several split vector quantisers. It aims to overcome the losses of rate-distortion efficiency in split vector quantisers, by exploiting full vector dependencies before the vector splitting. It is shown that the SSVQ alleviates the losses in two of the three vector quantiser 'advantages'. The SSVQ also has a remarkably low computational complexity, though this is achieved at the cost of an increase in memory requirements.

Statement of Originality

This work has not previously been submitted for a degree or diploma in any university. To the best of my knowledge and belief, the thesis contains no material previously published or written by another person except where due reference is made in the thesis itself.

Stephen So

March 2005

Contents

Abstract **v**

Statement of Originality **vii**

List of Figures **xvii**

List of Tables **xxiii**

Notation **xxix**

Acronyms **xxxiii**

Acknowledgements **xxxv**

1 Introduction **1**

1.1 The Role of Signal Coding 1

1.2 Quantisation of Scalars and Vectors 3

1.3 Dissertation Organisation 4

1.3.1 Chapter Summary 4

1.3.2 Composite Literature Review 7

1.4 Contributions and Resulting Publications 8

1.4.1 Contributions Made in this Dissertation 8

1.4.2 Journal Articles 10

1.4.3 Conference Articles 10

2 Efficient Block Quantisation **13**

2.1 Abstract 13

2.2 Introduction to Block Quantisation 15

2.2.1 The Karhunen-Loève Transform 17

2.2.2 Non-Uniform Scalar Quantisation 29

2.2.3 Bit Allocation 30

2.2.4 Problems with the KLT in Transform Coding 32

2.3 The Discrete Cosine Transform 33

2.4 Adaptive Transform Coding 37

2.4.1 Introduction 37

2.4.2 Literature Review 37

2.4.3 Using K-Means Clustering 40

2.5 Gaussian Mixture Model-Based Block Quantisation 42

2.5.1 Source Modelling for Quantiser Design 42

2.5.2 PDF Estimation Using Gaussian Mixture Models and the EM Algorithm 46

2.5.3 Bit Allocation 53

2.5.4 Minimum Distortion Block Quantisation 55

2.5.5 Quantiser Index Encoding 56

2.5.6 Computational Complexity and Memory Requirements 57

2.6 GMM-based Block Quantisation using the Discrete Cosine Transform . . . 58

2.6.1 PDF Estimation 59

2.6.2 Minimum Distortion Block Quantisation 60

2.7 GMM-Based Block Quantisation with Memory 60

2.7.1 Introduction 60

2.7.2 Predictive GMM-Based Block Quantisation 61

2.7.3 Multi-Frame GMM-Based Block Quantisation 63

2.8 Non-Integer Bit Allocation and Encoding at a Fixed-Rate 64

2.8.1 Introduction 64

2.8.2 Positional Value Number Systems 66

2.8.3 Fractional Bit Encoding 68

2.8.4 Heuristic Algorithms for Compensating Quantiser Levels 69

2.9 Chapter Summary 72

3 Efficient Vector Quantisation **75**

3.1 Abstract 75

3.2 Introduction 76

3.3 Vector Quantiser Design Using the Linde-Buzo-Gray Algorithm 77
3.4 Advantages of Vector Quantisation 79
3.4.1 Linear and Non-Linear Dependency 79
3.4.2 Dimensionality 80
3.4.3 Probability Density Function Shape 81
3.4.4 Space-Filling Advantage 81
3.4.5 Shape Advantage 82
3.4.6 Memory Advantage 82
3.5 Practical Limitations of Unconstrained Vector Quantisation 84
3.6 Product Code Vector Quantisers 84
3.6.1 Introduction 84
3.6.2 Split Vector Quantisers 86
3.6.3 Multistage Vector Quantisers 87
3.7 Tree Structured and Classified Vector Quantisers 90
3.8 Switched Split Vector Quantisation 91
3.8.1 Hybrid Vector Quantisation Schemes 91
3.8.2 Suboptimality of Split Vector Quantisers 92
3.8.3 Switched Split Vector Quantisers 94
3.8.4 Advantages of the Switched Split Vector Quantiser 98
3.8.5 Computational Complexity and Memory Requirements 102
3.9 Chapter Summary 105

4 Lossy Image Coding **107**
4.1 Abstract 107
4.2 Introduction 108
4.3 Distortion Measures for Image Coding 110
4.4 Overview of Image Coding Techniques 111
4.4.1 Vector Quantisation 111
4.4.2 Transform Coding 112
4.4.3 Subband Coding 115
4.4.4 Wavelet-Based Image Coding 121
4.5 Non-Expansive Filtering for the Subband Decomposition of Images 134
4.5.1 Non-Causal Filtering with Minimal Delay 135

4.5.2 Signal Extension Methods 138

4.5.3 An Example of Symmetric Extension for Subband Decomposition of Finite Length Signals 144

4.5.4 An Example of Symmetric Extension for the Discrete Wavelet Transform of Finite Length Signals 147

4.6 Experimental Setup for Image Coding Simulations 152

4.7 Application of the Block Quantiser 155

4.7.1 The KLT-Based Block Quantiser 155

4.7.2 The DCT-Based Block Quantiser 157

4.7.3 The K-Means-Based Multiple Transform Block Quantiser 162

4.8 Application of the GMM-Based Block Quantiser 169

4.8.1 The GMM-Based Block Quantiser 169

4.8.2 Comparison Between Integer and Fractional Bit-Based Cluster Block Quantisers in the GMM-Based Block Quantiser 178

4.8.3 The GMM-DCT-Based Block Quantiser 179

4.9 Reducing Block Artifacts in the GMM-Based Block Quantiser Using Wavelet Pre-Processing 186

4.9.1 Experimental Setup 189

4.9.2 Results and Discussion 190

4.10 Chapter Summary 190

5 LPC Parameter Quantisation in Narrowband Speech Coding 197

5.1 Abstract 197

5.2 Preliminaries of Speech Coding 198

5.2.1 Narrowband versus Wideband Speech 198

5.2.2 Speech Production 198

5.2.3 Autoregressive Modelling and Linear Prediction Analysis of the Speech Signal 200

5.3 Linear Prediction-Based Speech Coding 207

5.3.1 The LPC Vocoder 209

5.3.2 The RPE-LTP Speech Coder 210

5.3.3 Differential Pulse Coded Modulation with Perceptual Weighting . . 211

5.3.4 Code Excited Linear Predictive Speech Coders 215

5.3.5 Pre-Processing of LPC Coefficients: Bandwidth Expansion and High Frequency Compensation 223

5.4 LPC Parameter Representations 226

5.4.1 Reflection Coefficients, Log Area Ratios and Arcsine Reflection Coefficients 226

5.4.2 Line Spectral Frequencies 229

5.4.3 Immittance Spectral Pairs 233

5.5 Quantisation of LPC Parameters 234

5.5.1 LPC Parameter Distortion Measures 235

5.5.2 Experimental Setup for LSF Quantisation Experiments 238

5.5.3 PDF-Optimised Scalar Quantisers with Non-Uniform Bit Allocation 238

5.5.4 Product Code Vector Quantisation 240

5.5.5 GMM-Based Block Quantisation 245

5.6 LSF Quantisation Experiments 248

5.6.1 The KLT-Based Block Quantiser 248

5.6.2 The Multi-Frame GMM-Based Block Quantiser 249

5.6.3 The Switched Split Vector Quantiser 256

5.7 Chapter Summary 263

6 LPC Parameter Quantisation in Wideband Speech Coding 265

6.1 Abstract 265

6.2 Introduction 266

6.2.1 The Improved Quality of Wideband Speech 266

6.2.2 LPC Analysis of Wideband Speech 267

6.2.3 Coding of Wideband Speech 268

6.3 Wideband Speech Coders 269

6.3.1 The ITU-T G.722 Wideband Speech Coder 269

6.3.2 The Transform Coded Excitation (TCX) Wideband Speech Coder . 270

6.3.3 The Adaptive Multirate Wideband (AMR-WB) Speech Coder 271

6.4 Quantisation of Wideband LPC Parameters 273

6.4.1 Introduction 273

6.4.2 Literature Review 274

6.4.3 Experimental Setup 275

6.4.4 PDF-Optimised Scalar Quantisers with Non-Uniform Bit Allocation 275

6.4.5 Unconstrained Vector Quantisers and an Informal Lower Bound for Transparent Coding 278

6.4.6 Split-Multistage Vector Quantisers with MA Predictor 282
6.4.7 GMM-Based Block Quantisers 283
6.4.8 Switched Split Vector Quantisers 286
6.4.9 Multi-Frame GMM-based Block Quantisers 290
6.5 Chapter Summary 292

7 MFCC Quantisation in Distributed Speech Recognition 297

7.1 Abstract 297
7.2 Preliminaries of Speech Recognition 298
7.2.1 Speech Production 298
7.2.2 Feature Extraction 299
7.2.3 Pattern Recogniser 303
7.3 Client/Server-Based Speech Recognition 304
7.3.1 Network Speech Recognition 305
7.3.2 Distributed Speech Recognition 308
7.4 The ETSI Aurora-2 Experimental Framework 312
7.5 Setup of Experiments for Evaluating Different Quantisation Schemes in a DSR Framework 313
7.6 Recognition Performance of the Memoryless GMM-Based Block Quantiser . 315
7.7 Recognition Performance of the Multi-Frame GMM-Based Block Quantiser 316
7.8 Comparison with the Recognition Performance of the Non-Uniform Scalar Quantiser 318
7.9 Comparison with the Recognition Performance of the Unconstrained Vector Quantiser 319
7.10 Effect of Additive Noise on Recognition Performance 321
7.11 Chapter Summary 327

8 Conclusions and Future Research 329

8.1 Chapter Summary and Conclusions 329
8.1.1 Chapter 2: Efficient Block Quantisation 329
8.1.2 Chapter 3: Efficient Vector Quantisation 330
8.1.3 Chapter 4: Lossy Image Coding 331
8.1.4 Chapter 5: LPC Parameter Quantisation in Narrowband Speech Coding 331

8.1.5 Chapter 6: LPC Parameter Quantisation in Wideband Speech Coding 332
8.1.6 Chapter 7: MFCC Quantisation in Distributed Speech Recognition . 334
8.2 Suggestions for Future Research 334

Bibliography **337**

List of Figures

2.1 Schematic of a typical block quantiser 16

2.2 Scatter diagram of joint Gaussian random source 25

2.3 Estimated PDF of joint Gaussian random source 25

2.4 Block quantisation using 3 bits/sample with no KLT applied 26

2.5 Scalar quantisation of KLT coefficients k_1 and k_2 using 3 bits/sample 27

2.6 Block quantisation using 3 bits/sample with KLT applied 28

2.7 The function $K(b)$ of a Gaussian Lloyd-Max scalar quantiser 31

2.8 Scalar quantisation of DCT coefficients d_1 and d_2 using 3 bits/sample . . . 35

2.9 Block quantisation using 3 bits/sample with DCT applied 36

2.10 The image 'goldhill' along with its 2D PDF and first eigenvector 38

2.11 Voronoi regions from K-means algorithm 40

2.12 The image 'barbara' and its PDF in two dimensions 47

2.13 Original PDF and a single cluster GMM of the image 'barbara' 48

2.14 Original PDF and a 4 cluster GMM of the image 'barbara' 49

2.15 Original PDF and a 16 cluster GMM of the image 'barbara' 50

2.16 Original PDF and a 32 cluster GMM of the image 'barbara' 51

2.17 Minimum distortion block quantisation (Q – cluster block quantiser) 55

2.18 Example of quantiser level encoding and cluster number partitioning 56

2.19 PDF estimation and bit allocation procedure for the GMM-DCT-based block quantiser 59

2.20 Schematic of the modified GMM-based block quantiser based on DCT . . . 60

2.21 Schematic of the 'modified case' predictive GMM-based block quantiser . . 61

2.22 Schematic of the 'trained case' predictive GMM-based block quantiser . . . 61

3.1 Voronoi regions of a nearest neighbour (MSE) vector quantiser 77

3.2 Voronoi regions of a 4 bit vector quantiser designed using the LBG algorithm 78

3.3 Vector quantisation using 3 bits/sample of random Gaussian vectors 83
3.4 Two part split vector quantiser 86
3.5 Three stage multistage vector quantiser 88
3.6 Codebook searches of an M-L searched multistage vector quantiser 89
3.7 Tree structured vector quantisation 90
3.8 Illustrating the memory suboptimality of two-part split vector quantiser in two dimensions 93
3.9 Illustrating the space-filling and shape suboptimality of two-part split vector quantiser in two dimensions 94
3.10 Switched split vector quantiser (training) 95
3.11 Switched split vector quantiser (coding) 96
3.12 Quantiser index partitioning for 3 bit SSVQ with two-way switch 97
3.13 Illustrating the memory advantage of the switched split vector quantiser . . 98
3.14 Comparing product code-vectors of 8 bits/vector split vector quantiser with 8 bits/vector switched split vector quantiser on vectors with memory 100
3.15 Comparing product code-vectors of 8 bits/vector split vector quantiser with 8 bits/vector switched split vector quantiser on vectors without memory . . 101
3.16 Computational complexity of two-part and three-part SSVQ as a function of number of bits for switch vector quantiser 102
3.17 Memory requirements of two-part and three-part SSVQ as a function of number of bits for switch vector quantiser 104

4.1 A typical 8-bit greyscale image ‘bird’ 109
4.2 The image ‘lena’ coded using a vector quantiser at 0.5625 bits/pixel 112
4.3 Block diagram of the lossy JPEG image coding standard 113
4.4 The block artifacts of JPEG on the image ‘bird’ 114
4.5 Uniform splitting of frequency spectrum into four subbands 115
4.6 Block diagram of a two-band subband coder 116
4.7 Brick-wall filter with ideal filter response 116
4.8 Frequency response of quadrature mirror filters 117
4.9 Frequency response of the Johnston QMF (8-taps) 118
4.10 Frequency response of the Johnston QMF (16-taps) 119
4.11 Two dimensional subband coding using separable filters 119
4.12 Subband decomposition configurations 120
4.13 Laplacian pyramidal image coder 123

4.14 Discrete wavelet transform of greyscale test images: 'lena', 'bird', 'cameraman' (using 9/7-tap biorthogonal wavelet filters) 127

4.15 Image subbands and their respective directions 128

4.16 Zerotrees used in the embedded zerotree wavelet coder 129

4.17 Spatial orientation trees used in SPIHT coding 132

4.18 Subband coding with non-causal filtering and signal extension 135

4.19 Impulse responses of causal and non-causal linear phase filters 136

4.20 The circular extension method 138

4.21 The symmetric extension method 139

4.22 Impulse response of an even and odd-tapped symmetric filter 141

4.23 Symmetric extension for filters with an even number of taps 142

4.24 Symmetric extension for filters with an odd number of taps 143

4.25 Ramp signal, $x(n) = n + 1$ where $n = 0, 1, \ldots, N$ 146

4.26 Symmetrically extended ramp signal, $\tilde{x}(n)$ 146

4.27 Low-pass and high-pass filtered signals, $v_1(n)$ and $v_2(n)$ 147

4.28 Low-pass and high-pass extended and upsampled signals, $\tilde{w}_1(n)$ and $\tilde{w}_2(n)$. 148

4.29 Filtered extended and upsampled signals, $\hat{x}_1(n)$ and $\hat{x}_2(n)$ 148

4.30 Symmetrically extended ramp signal, $\tilde{x}(n)$ 149

4.31 Low-pass and high-pass filtered signals, $v_1(n)$ and $v_2(n)$ 150

4.32 Filtered extended and upsampled signals, $\hat{x}_1(n)$ and $\hat{x}_2(n)$ 151

4.33 Training set of images 153

4.34 Testing set of images 154

4.35 Results of the 'goldhill' image at various bitrates using the KLT-based block quantiser 158

4.36 Results of the 'boat' image at various bitrates using the KLT-based block quantiser 159

4.37 Results of the 'goldhill' image at various bitrates using the DCT-based block quantiser 163

4.38 Results of the 'boat' image at various bitrates using the DCT-based block quantiser 164

4.39 Results of the 'goldhill' image using the K-means-based multiple transform block quantiser with varying number of clusters at 0.5 bits/pixel 167

4.40 Results of the 'boat' image using the K-means-based multiple transform block quantiser at 0.5 bits/pixel 168

4.41 Plot of PSNR as a function of bitrate of GMM-based block quantisation of 'boat' with varying number of clusters 171

4.42 Results of the 'goldhill' image using the GMM-based block quantiser with varying number of clusters at 0.5 bits/pixel 172

4.43 Results of the 'boat' image using the GMM-based block quantiser at 0.5 bits/pixel 173

4.44 Plot of PSNR as a function of bitrate of GMM-DCT-based block quantisation of 'boat' with varying number of clusters 181

4.45 Results of the 'goldhill' image using the GMM-DCT-based block quantiser with varying number of clusters at 0.5 bits/pixel 182

4.46 Results of the 'boat' image using the GMM-DCT-based block quantiser at 0.5 bits/pixel 183

4.47 Wavelet tree structure for extracting vectors from the same spatial location 187

4.48 Quality comparison of the 'lena' image using the GMM-based block quantiser with and without the wavelet transform pre-processing at 0.5 bits/pixel191

4.49 Quality comparison of the 'boat' image using the GMM-based block quantiser with and without the wavelet transform pre-processing at 0.5 bits/pixel192

4.50 Quality comparison of the 'barbara' image using the GMM-based block quantiser with and without the wavelet transform pre-processing at 0.5 bits/pixel 193

4.51 Comparing the wavelet transform of the images 'man', 'vegas', and 'barbara' with their quantised versions using the GMM-based block quantiser at 0.5 bits/pixel 194

5.1 Waveform and spectrogram of narrowband speech 199

5.2 Anatomy of the human vocal tract 199

5.3 Source-filter synthesis model of speech 201

5.4 Estimation of the spectral envelope and excitation/residual signal of voiced speech, /i/, using a 12th order linear prediction filter 204

5.5 Estimation of the spectral envelope and excitation/residual signal of unvoiced speech, /s/, using a 12th order linear prediction filter 204

5.6 Cubed autocorrelation (unbiased) of the residual for voiced and unvoiced speech 205

5.7 Plot of normalised residual error versus order of linear prediction for voiced and unvoiced speech 206

5.8 Different types of prediction used in speech coding 208

5.9 Block diagram of generalised speech synthesis 208

5.10 Block diagram of the LPC vocoder 209

5.11 Block diagram of the RPE-LTP speech coder used in GSM 06.10 211

5.12 Block diagram of the DPCM coder with and without noise shaping 212

5.13 Block diagram of the generalised DPCM speech coder 214

5.14 The analysis-by-synthesis speech coder 215

5.15 Block diagram of the CELP coder 217

5.16 Relationship between frames, subframes, and analysis windows 217

5.17 Frequency response of a perceptual weighting filter 219

5.18 Modified CELP coder with lower computational complexity 220

5.19 Block diagram of the pitch predictor and adaptive codebook 222

5.20 Illustrating the effect of bandwidth expansion on the spectral envelope estimate 224

5.21 Illustrating the effect of high frequency compensation on the spectral envelope estimate 225

5.22 The acoustic tube model of the vocal tract 227

5.23 The acoustic tube with two artificial boundary conditions for the line spectral frequency representation 231

5.24 Pole-zero plot of $P(z)$ and $Q(z)$ of 12th order LPC analysis 232

5.25 Spectral envelope estimate and line spectral frequency locations 232

5.26 Original and reconstructed spectral envelope of two LSFs that have been shifted 237

5.27 Scatter diagram of the first and second line spectral frequencies 244

5.28 Histograms of 10 line spectral frequencies for narrowband speech 245

5.29 Average spectral distortion (SD) of two-part and three-part SSVQ at 24 bits/frame as a function of the bits required for switching 257

5.30 Computational complexity of two-part and three-part SSVQ at 24 bits/frame as a function of the bits required for switching 258

5.31 Partial-band spectral distortion (SD) histograms for the 24 bits/frame two-part switched split vector quantiser ($m = 16$) using minimum distortion (using weighted distance measure and spectral distortion) and nearest neighbour selection 260

6.1 Waveform and spectrogram of wideband speech 266

6.2 Spectral envelope estimate of wideband speech for different orders of linear prediction analysis 267

6.3 Block diagram of the 64 kbps ITU-T G.722 wideband speech coder 269

6.4 Block diagram of the 16 kbps TCX wideband speech coder 270

6.5 Histograms of immittance spectral frequencies (ISFs) and line spectral frequencies (ISFs) from the TIMIT database 276

6.6 Original and reconstructed power spectral envelope estimates of 16th order LSFs and ISFs with last parameter shifted by 142 Hz 280

6.7 Extrapolating from the operating distortion-rate curve of the unconstrained vector quantisers for wideband LSFs and ISFs to approximate a lower bound 281

6.8 Block diagram of the split-multistage vector quantiser 282

6.9 Original and reconstructed spectral envelope of two LSFs that have been shifted 289

7.1 A typical speech recognition system (after [178]) 298

7.2 Waveform and spectrogram of the sentence, *she had your dark suit in greasy wash-water all year* 299

7.3 Magnitude and phase response of the pre-emphasis filter 300

7.4 The effect of pre-emphasis on the power spectral density of speech 301

7.5 Filterbank of Mel frequency-warped triangular-shaped filters used for MFCC calculation from 8 kHz speech ($M = 20$) 303

7.6 A typical 6 state, left-to-right hidden Markov model (HMM) (after [203]) . 304

7.7 Client-server-based speech recognition modes 309

7.8 Summary of average word recognition accuracies for all quantisation schemes considered 320

7.9 Plot of recognition accuracy versus SNR for all quantisation schemes (excluding SQ) at 1.2 kbps 323

7.10 Plot of recognition accuracy versus SNR for all quantisation schemes (excluding SQ) at 0.6 kbps 326

List of Tables

2.1 Bitrate independent computational complexity of the GMM-based block quantiser (after [183]) . . . 58

2.2 Example of integer bit allocation table . . . 65

2.3 Example of binary coding a block . . . 65

2.4 Example of fractional bit allocation table . . . 65

2.5 System of positional weights . . . 67

2.6 Positional weights for the S number system . . . 67

2.7 Example of converting decimal to the S number system via method 1 . . . 68

2.8 Example of converting decimal to the S number system via method 2 . . . 68

4.1 Coefficients of the spline variant 9/7-tap wavelet filter . . . 127

4.2 Symmetry properties of convolved signals . . . 140

4.3 Coefficients of the 32-tap Johnston symmetric QMFs . . . 145

4.4 Training and testing image set for image coding experiments . . . 154

4.5 PSNR as a function of bitrate for the KLT-based block quantiser . . . 156

4.6 Levels allocation table for KLT-based block quantiser at 0.25 bits/pixel . . 156

4.7 Levels allocation table for KLT-based block quantiser at 0.5 bits/pixel . . . 157

4.8 Levels allocation table for KLT-based block quantiser at 1.0 bits/pixel . . . 157

4.9 PSNR as a function of bitrate for the DCT-based block quantiser . . . 160

4.10 Levels allocation table for DCT-based block quantiser at 0.25 bits/pixel . . 161

4.11 Levels allocation table for DCT-based block quantiser at 0.5 bits/pixel . . . 161

4.12 Levels allocation table for DCT-based block quantiser at 1.0 bits/pixel . . . 162

4.13 PSNR as a function of number of clusters for the K-means-based multiple transform block quantiser at 0.5 bits/pixel . . . 165

4.14 Levels allocation table for cluster 1 of 4 cluster K-means-based multiple transform block quantiser at 0.5 bits/pixel . . . 166

4.15 Levels allocation table for cluster 2 of 4 cluster K-means-based multiple transform block quantiser at 0.5 bits/pixel 167

4.16 Levels allocation table for cluster 3 of 4 cluster K-means-based multiple transform block quantiser at 0.5 bits/pixel 168

4.17 Levels allocation table for cluster 4 of 4 cluster K-means-based multiple transform block quantiser at 0.5 bits/pixel 169

4.18 PSNR as a function of number of clusters for the GMM-based block quantiser at 0.5 bits/pixel 170

4.19 Levels allocation table for cluster 1 of 4 cluster GMM-based block quantiser at 0.5 bits/pixel 175

4.20 Levels allocation table for cluster 2 of 4 cluster GMM-based block quantiser at 0.5 bits/pixel 175

4.21 Levels allocation table for cluster 3 of 4 cluster GMM-based block quantiser at 0.5 bits/pixel 176

4.22 Levels allocation table for cluster 4 of 4 cluster GMM-based block quantiser at 0.5 bits/pixel 176

4.23 Cluster densities for the 4 cluster GMM-based block quantiser at 0.5 bits/pixel176

4.24 PSNR performance of the fixed-rate (2 bits/pixel), 4 cluster GMM-based block quantiser using integer and fractional bit-based cluster block quantisers (underallocation) on images that were part of the training set 176

4.25 PSNR performance of the fixed-rate (2 bits/pixel), 4 cluster GMM-based block quantiser using integer and fractional bit-based cluster block quantisers (underallocation) on images that were not part of the training set 177

4.26 PSNR performance of the fixed-rate (0.15 bits/pixel), 4 cluster GMM-based block quantiser using integer and fractional bit-based cluster block quantisers (overallocation) on images that were part of the training set 177

4.27 PSNR performance of the fixed-rate (0.15 bits/pixel), 4 cluster GMM-based block quantiser using integer and fractional bit-based cluster block quantisers (overallocation) on images that were not part of the training set 177

4.28 Effective bitrates for each cluster at 2 bits/pixel 178

4.29 Effective bitrates for each cluster at 0.15 bits/pixel 179

4.30 PSNR as a function of number of clusters for the GMM-DCT-based block quantiser at 0.5 bits/pixel 180

4.31 Levels allocation table for cluster 1 of 4 cluster GMM-DCT-based block quantiser at 0.5 bits/pixel 184

4.32 Levels allocation table for cluster 2 of 4 cluster GMM-DCT-based block quantiser at 0.5 bits/pixel 184

4.33 Levels allocation table for cluster 3 of 4 cluster GMM-DCT-based block quantiser at 0.5 bits/pixel 185

4.34 Levels allocation table for cluster 4 of 4 cluster GMM-DCT-based block quantiser at 0.5 bits/pixel 185

4.35 Cluster densities for the 4 cluster GMM-DCT-based block quantiser at 0.5 bits/pixel 186

4.36 Comparison of average processing times between the GMM-KLT and GMM-DCT-based block quantiser (in seconds) 186

4.37 PSNR of the 16 cluster GMM-based block quantiser at 0.5 bits/pixel with and without the wavelet transform pre-processing 189

4.38 Summary of PSNRs for all block quantisation schemes considered at 0.5 bits/pixel 196

5.1 Comparing the characteristics of the DPCM with perceptual weighting and CELP speech coders 216

5.2 Partial-band spectral distortion (SD) performance of uniform scalar quantisers operating at 34 bits/frame from the FS-1016 4.8 kbps CELP coder on the TIMIT database 240

5.3 Partial-band spectral distortion (SD) performance of non-uniform scalar quantisers for different bitrates on LSF vectors from the TIMIT database . 240

5.4 Partial-band spectral distortion (SD), computational complexity, and memory requirements (ROM) of the two-part split vector quantiser as a function of bitrate on LSF vectors from the TIMIT database 241

5.5 Partial-band spectral distortion (SD), computational complexity, and memory requirements (ROM) of the three-part split vector quantiser as a function of bitrate on LSF vectors from the TIMIT database 241

5.6 Partial-band spectral distortion (SD), computational complexity, and memory requirements (ROM) of the two-stage multistage vector quantiser as a function of bitrate on LSF vectors from the TIMIT database 242

5.7 Average spectral distortion (SD) performance of the 16 cluster, memoryless, fixed-rate GMM-based block quantiser using spectral distortion criterion at different bitrates on LSF vectors from the TIMIT database 246

5.8 Bitrate independent computational complexity (in kflops/frame) and memory requirements (ROM) of the GMM-based block quantiser using spectral distortion-based quantiser selection as a function of the number of clusters . 246

5.9 Average spectral distortion of the 16 cluster predictive GMM-based block quantiser (trained case) using SD criterion as a function of bitrate on LSF vectors from the TIMIT database 247

5.10 Prediction coefficients for the predictive GMM-based block quantiser (calculated using the covariance method) 247

5.11 Average spectral distortion of the KLT-based block quantiser as a function of bitrate on LSF vectors from the TIMIT database 249

5.12 Average spectral distortion of the 16 cluster multi-frame GMM-based block quantiser using MSE criterion as a function of bitrate and number of concatenated frames, p . . . 250

5.13 Bitrate independent computational complexity (in kflops/frame) and memory requirements (ROM) of the multi-frame GMM-based block quantiser using MSE criterion as a function of number of concatenated vectors, p, and number of clusters, m . . . 251

5.14 Average spectral distortion of the 32 cluster multi-frame GMM-based block quantiser using MSE criterion as a function of bitrate and number of concatenated frames, p . . . 252

5.15 Average spectral distortion of the 16 cluster multi-frame GMM-based block quantiser using SD criterion as a function of bitrate and number of concatenated frames, p . . . 253

5.16 Average spectral distortion of the 32 cluster multi-frame GMM-based block quantiser using SD criterion as a function of bitrate and number of concatenated frames, p . . . 254

5.17 Bitrate independent computational complexity (in kflops/frame) and memory requirements (ROM) of the multi-frame GMM-based block quantiser using SD criterion as a function of number of concatenated vectors, p, and number of clusters, m . . . 254

5.18 Partial-band spectral distortion (SD) and computational performance of the two-part switched split vector quantiser at 24 bits/frame as a function of the number of switch directions using different bit allocations . . . 255

5.19 Partial-band spectral distortion (SD) and computational performance of the three-part switched split vector quantiser at 24 bits/frame as a function of the number of switch directions using different bit allocations . . . 256

5.20 Partial-band spectral distortion (SD), computational complexity, and memory requirements (ROM) of the two-part switched split vector quantiser as a function of bitrate and number of switch directions . . . 259

5.21 Partial-band spectral distortion (SD), computational complexity, and memory requirements (ROM) of the three-part switched split vector quantiser as a function of bitrate and number of switch directions . . . 259

6.1 Average spectral distortion of the PDF-optimised scalar quantisers as a function of bitrate on wideband ISF vectors from the TIMIT database . . . 277

6.2 Average spectral distortion of the PDF-optimised scalar quantisers as a function of bitrate on wideband LSF vectors from the TIMIT database . . . 277

6.3 Average spectral distortion of the unconstrained vector quantiser as a function of bitrate on wideband ISF vectors from the TIMIT database . . . 279

6.4 Average spectral distortion of the unconstrained vector quantiser as a function of bitrate on wideband LSF vectors from the TIMIT database . . . 279

6.5 Average spectral distortion as a function of bitrate of the AMR-WB (ITU-T G.722.2) split-multistage vector quantiser with MA prediction on wideband ISF vectors from the TIMIT database 283

6.6 Average spectral distortion performance of the 16 cluster memoryless, fixed rate GMM-based block quantiser using spectral distortion criterion at different bitrates on wideband ISF vectors from the TIMIT database 284

6.7 Average spectral distortion performance of the 16 cluster memoryless, fixed rate GMM-based block quantiser using spectral distortion criterion at different bitrates on wideband LSF vectors from the TIMIT database 284

6.8 Bitrate independent computational complexity (in kflops/frame) and memory requirements (ROM) of the GMM-based block quantiser using spectral distortion-based quantiser selection as a function of the number of clusters for wideband speech coding 286

6.9 Average spectral distortion (SD), computational complexity, and memory requirements (ROM) of the five-part switched split vector quantiser using unweighted MSE as a function of bitrate and number of switch directions of wideband LSF vectors from the TIMIT database 286

6.10 Average spectral distortion (SD), computational complexity, and memory requirements (ROM) of the five-part switched split vector quantiser using unweighted MSE as a function of bitrate and number of switch directions of wideband ISF vectors from the TIMIT database 288

6.11 Average spectral distortion (SD) of the five-part switched split vector quantiser using weighted MSE as a function of bitrate and number of switch directions of wideband LSF vectors from the TIMIT database 290

6.12 Average spectral distortion as a function of bitrate and number of concatenated frames, p, of the 16 cluster multi-frame GMM-based block quantiser on wideband LSF vectors from the TIMIT database 291

6.13 Average spectral distortion as a function of bitrate and number of concatenated frames, p, of the 16 cluster multi-frame GMM-based block quantiser on wideband ISF vectors from the TIMIT database 293

6.14 Bitrate independent computational complexity (in kflops/frame) and memory requirements (ROM) of multi-frame GMM-based block quantiser as a function of the number of concatenated frames, p and number of clusters, m 293

6.15 Average spectral distortion as a function of bitrate and number of concatenated frames, p, of the 32 cluster multi-frame GMM-based block quantiser on wideband LSF vectors from the TIMIT database 294

7.1 Average word recognition accuracy as a function of bitrate and number of clusters for the memoryless GMM-based block quantiser (baseline accuracy = 98.01%) 315

7.2 Average word recognition accuracy as a function of bitrate and number of frames for 16 cluster multi-frame GMM-based block quantiser (baseline accuracy = 98.01%) 316

7.3 Average word recognition accuracy as a function of bitrate and number of clusters for 5 frame multi-frame GMM-based block quantiser (baseline accuracy = 98.01%) . . . 317

7.4 Average word recognition accuracy as a function of bitrate for non-uniform scalar quantiser (baseline accuracy = 98.01%) . . . 318

7.5 Average word recognition accuracy, computational complexity (in kflops/frame), and memory requirements (ROM) as a function of bitrate for vector quantiser (baseline accuracy = 98.01%) . . . 319

7.6 Bitrate independent computational complexity (in kflops/frame) and memory requirements (ROM) of the multi-frame GMM-based block quantiser as a function of number of concatenated vectors, p, and number of clusters, m 320

7.7 Word recognition accuracy for speech corrupted with subway noise at varying SNRs (in dB) at 1.2 kbps. . . . 322

7.8 Word recognition accuracy for speech corrupted with babble noise at varying SNRs (in dB) at 1.2 kbps . . . 322

7.9 Word recognition accuracy for speech corrupted with car noise at varying SNRs (in dB) at 1.2 kbps . . . 322

7.10 Word recognition accuracy for speech corrupted with exhibition noise at varying SNRs (in dB) at 1.2 kbps . . . 322

7.11 Word recognition accuracy for speech corrupted with subway noise at varying SNRs (in dB) at 0.6 kbps. . . . 324

7.12 Word recognition accuracy for speech corrupted with babble noise at varying SNRs (in dB) at 0.6 kbps . . . 324

7.13 Word recognition accuracy for speech corrupted with car noise at varying SNRs (in dB) at 0.6 kbps . . . 325

7.14 Word recognition accuracy for speech corrupted with exhibition noise at varying SNRs (in dB) at 0.6 kbps . . . 325

Notation

General

x	a random variable (scalar)
$E[x]$	expectation of the random variable x
$\{x_i\}_{i=1}^N$	set of N scalars
$\boldsymbol{x}$	a random vector (column)
$\boldsymbol{x}^T$	vector transpose of $\boldsymbol{x}$
$\Vert\boldsymbol{x}\Vert$	Euclidean norm of vector $\boldsymbol{x}$
$\Vert\boldsymbol{x}-\boldsymbol{y}\Vert$	Euclidean distance between vectors $\boldsymbol{x}$ and $\boldsymbol{y}$
$\boldsymbol{A}$	a square matrix
$\boldsymbol{I}$	a square identity matrix
$\boldsymbol{A}(i,j)$	element on the ith row and jth column of matrix $\boldsymbol{A}$
$\boldsymbol{A}^{-1}$	inverse of matrix $\boldsymbol{A}$
$\boldsymbol{A}^T$	matrix transpose of $\boldsymbol{A}$
$\vert\boldsymbol{A}\vert$	determinant of matrix $\boldsymbol{A}$
n	dimension/size of a vector/block
N	number of vectors
$\mathcal{R}^k$	real vector space of dimension k
$\mathcal{I}$	integer space

Gaussian Mixture Models

$\boldsymbol{\mu}$	mean vector
$\boldsymbol{\Sigma}$	covariance matrix
$\mathcal{N}(\boldsymbol{x};\boldsymbol{\mu},\boldsymbol{\Sigma})$	multivariate Gaussian of random vector, $\boldsymbol{x}$, with mean, $\boldsymbol{\mu}$, and covariance matrix, $\boldsymbol{\Sigma}$

Eigenvalue Decomposition and Karhunen-Loève Transform

$\boldsymbol{v}_i$	ith eigenvector
λ_i	ith eigenvalue
$\boldsymbol{\Sigma}$	covariance matrix
$\boldsymbol{U}$	$N \times N$ matrix whose columns contain the eigenvectors, $\{\boldsymbol{v}_i\}_{i=1}^{N}$
$\boldsymbol{P}$	$N \times N$ transformation matrix
$\boldsymbol{R_x}$	autocorrelation matrix of vectors, $\boldsymbol{x}$

Wavelet Transform

$L^2(R)$	space of square integrable functions
$\psi_{a,b}(t)$	wavelet function of scale, a, and translation, b
$\phi_{a,b}(t)$	scaling function of scale, a, and translation, b

Autoregressive Modelling and Linear Prediction Analysis

$H(z)$	synthesis filter
$A(z)$	analysis filter
$a_{p,k}$	kth AR model parameter or linear prediction coefficient of order p
G_p	filter gain
P_{min}	minimum prediction error or residual error
$R(i)$	ith autocorrelation coefficient
$\phi(i,j)$	covariance coefficient (i,j)

LPC Parameter Representations

a_i	ith linear prediction coefficient
k_i	ith reflection coefficient
g_i	ith log-area-ratio
j_i	ith arcsine reflection coefficient
$P(z)$	transfer function of acoustic tube with closed glottal end
$Q(z)$	transfer function of acoustic tube with open glottal end
$\omega_i^{(p)}$	ith zero of $P(z)$ expressed as a frequency in rad/s (LSF)
$\omega_i^{(q)}$	ith zero of $Q(z)$ expressed as a frequency in rad/s (LSF)
f_i	ith frequency in Hz (LSF and ISF)
$\mathcal{I}_p(z)$	pth order immittance function
$\omega_k^{(i)}$	kth pole/zero of immittance function expressed as a frequency (ISF)
$D_{sd}(i)$	full-band spectral distortion of the ith speech frame
$D_{psd}(i)$	partial-band spectral distortion of the ith speech frame
$d_w(\boldsymbol{f}, \hat{\boldsymbol{f}})$	weighted Euclidean distance measure between original LPC vector, $\boldsymbol{f}$, and approximate LPC vector, $\hat{\boldsymbol{f}}$

Acronyms

3GPP	Third Generation Partnership Project
AAC	Advanced Audio Coding
ACELP	Algebraic Code Excited Linear Predictive
AMDF	Average Magnitude Difference Function
AMR	Adaptive Multirate (Narrowband)
AMR-WB	Adaptive Multirate Wideband
AR	Autoregressive
ASR	Automatic Speech Recognition
ASRC	ArcSine Reflection Coefficients
ATCELP	Adaptive Transform Code Excited Linear Predictive
BER	Bit Error Rate
BFCC	Bark frequency-warped cepstral coefficients
BQ	Block Quantiser/Quantisation
CELP	Code Excited Linear Predictive
COT	Coding Optimal Transform
CS-ACELP	Conjugate Structure Algebraic Code Excited Linear Predictive
CWT	Continuous Wavelet Transform
CVQ	Classified Vector Quantiser/Quantisation
DARPA	Defense Advanced Research Projects Agency
DCT	Discrete Cosine Transform
DWT	Discrete Wavelet Transform
DSR	Distributed Speech Recognition
EM	Expectation Maximisation
ETSI	European Telecommunications Standards Institute
EVD	Eigenvalue Decomposition
EZW	Embedded Zerotree Wavelet
GMM	Gaussian Mixture Model
GSM	Global System for Mobile Communications
ISF	Immittance Spectral Frequency
ISP	Immittance Spectral Pair
ITU-T	International Telecommunications Union (Telecommunications Standardisation Sector)
JPEG	Joint Photographic Experts Group

KLT	Karhunen-Loève Transform
LAR	Log-Area-Ratio
LDA	Linear Discriminant Analysis
LD-CELP	Low Delay Code Excited Linear Predictive
LOT	Lapped Orthogonal Transform
LPC	Linear Preditive Coding
LPCC	Linear Prediction-based Cepstral Coefficients
LSF	Line Spectral Frequency
LSP	Line Spectral Pair
MA	Moving Average
MDCT	Modified Discrete Cosine Transform
MFCC	Mel Frequency-Warped Cepstral Coefficients
MFCCP	Peak-isolated Mel Frequency-Warped Cepstral Coefficients
MP3	MPEG-1 Layer 3
MPEG	Moving Pictures Experts Group
MRA	Multiresolution Analysis
MSE	Mean Squared Error
MSVQ	Multistage Vector Quantiser/Quantisation
NSR	Network Speech Recognition
PARCOR	Partial Correlation
PCA	Principal Component Analysis
PDF	Probability Density Function
PLP	Perceptual Linear Prediction
PSD	Power Spectral Density
PSNR	Peak Signal-to-Noise Ratio
RC	Reflection Coefficients
RPE-LTP	Regular Pulse Excited-Long Term Prediction
SD	Spectral Distortion
SNR	Signal-to-Noise Ratio
SPIHT	Set Partitioning In Hierarchical Trees
SSVQ	Switched Split Vector Quantiser/Quantisation
STFT	Short Time Fourier Transform
SVD	Singular Value Decomposition
SVQ	Split Vector Quantiser/Quantisation
TCX	Transform Coded Excitation
TIMIT	Texas Instruments, Massachussetts Institute of Technology
TSVQ	Tree Structured Vector Quantiser/Quantisation
VFR_MFCCP	Variable Frame-Rate MFCCP
VQ	Vector Quantiser/Quantisation
WSS	Wide-Sense Stationary

Acknowledgements

Firstly, I would like to thank my supervisor, Professor Kuldip K. Paliwal, for his invaluable guidance, advice, and support during the course of my doctoral degree. His profound knowledge and expertise has not only been of tremendous help in my quest to understand, implement, and explore, but has also inspired my ever-growing interest in signal processing. I would also like to offer my thanks to Brett Wildermoth for his assistance in technical matters and computing support. I also want to acknowledge and thank my fellow colleagues from the Signal Processing Laboratory (in no particular order): Nanda Koestoer, Conrad Sanderson, Ben Shannon, Leigh Alsteris, Calvin Breakenridge, and anyone else who have happened to slip my mind during the time of writing, for offering their support and making the environment lively and memorable. It has been a great privilege to work with them. My thanks also go to the visiting researchers from around the world who have offered valuable thoughts and insights in their respective areas of signal processing research. I would also like to thank the School of Microelectronic Engineering and the Head of School, Professor David V. Thiel, for the advice and support I have received over the years. I would like to acknowledge the several examiners and reviewers of my journal and conference publications. It is because of their expert comments and invaluable feedback that has improved the standard of the research documented in this dissertation. Last but not least, I want to offer my sincerest gratitude to my parents for their ever-continuing support throughout the degree. It was this support, through thick and thin, that has successfully steered me through an often at times, turbulent and arduous journey.

Efficient Block Quantisation for Image and Speech Coding

Chapter 1

Introduction

1.1 The Role of Signal Coding

Signal coding or compression has played a significant role in the success of digital communications and multimedia. The use of signal coding pervades many aspects of our digital lifestyle—a lifestyle that has seen widespread demand for applications like third generation mobile telephony, portable music players, Internet-based video conferencing, digital television, etc. The issues that arise, when dealing with the transmission and storage of digital media, are the limited bandwidth of communication channels, the limited capacity of storage devices, and the limited processing ability of the encoding and decoding devices. The aim of signal coding is therefore to represent digital media, such as speech, music, images, and video, as efficiently as possible. Coding efficiency encompasses rate-distortion (for lossy coding), compression ratio (for lossless coding), computational complexity, and static and dynamic memory requirements.

The coding may be *lossless*, where no information is lost during the coding process. That is, the decoded data are exactly the same as the original. In *lossy* coding, some information is lost during the process so the decoded data will not be exactly the same as the original, thus they will suffer from quality degradation. While lossy coding techniques generally achieve better compression ratios than lossless ones, there will be a compromise between the amount of compression achievable and the degree of quality degradation or distortion incurred. This compromise is known as the *rate-distortion* trade-off [59]. The decision to use lossy or lossless coding depends on the requirements of the application. For example, medical imagery requires the preservation of critical detail and thus demands coding of the lossless type. For the vast majority of applications, which will be described below, lossy coding schemes are preferred since they generally achieve significantly higher compression ratios than lossless schemes.

A popular application involves the storage of high quality digitised music that is recorded from audio compact discs. Through the use of portable digital music players, consumers can store songs of their preference and also be able to play their music while they are roaming about. Such players have limited storage and processing abilities which have an influence on important factors such as the battery life of the device. Portable music players mostly use lossy perceptual coding algorithms such as MPEG-1 Layer 3 (MP3), Windows Media Audio (WMA), and MPEG-4 Advanced Audio Coding (AAC), where information loss is confined to parts of the audio that are less perceptible to the human ear.

When a person talks into his or her mobile phone, their speech is coded into a digital form by a digital signal processor and transmitted wirelessly over the cellular phone network where eventually it arrives at the other person's phone. There are various coding standards, used in second generation (2G) and earlier third generation (3G) mobile communication systems, which operate on narrowband speech[1]. Prominent coders include the GSM series, such as the 13 kbps full-rate (FR), the 5.6 kbps half-rate (HR), 12.2 kbps enhanced full-rate (EFR), and Adaptive Multirate (AMR) speech codecs. For wireless applications which demand higher quality speech reproduction, such as teleconferencing, Voice over IP, and other Internet applications, wideband speech[2] coding algorithms, such as the Adaptive Multirate Wideband codec (AMR-WB), have recently been adopted for use in the latest 3G systems [19]. Like the perceptual audio coders, low bitrate speech coders attempt to restrict the information loss to parts of the speech which are less perceptible to the human ear.

[1]Narrowband speech is sampled at 8 kHz [31] and bandlimited to the frequency range of 300-3400 Hz or 200-3400 Hz, which is the bandwidth of wired telephone speech in Europe and the US, respectively [117].

[2]Wideband speech is sampled at 16 kHz and bandlimited to the frequency range of 50-7000 Hz. The wider audio bandwidth improves the intelligibility [31] and naturalness of speech [19].

1.2 Quantisation of Scalars and Vectors

The fundamental operation in lossy signal coding is quantisation, which is the mapping of input samples to a codebook of a finite number of codewords. Its rate-distortion efficiency is influenced by the properties of the signal source, such as statistical dependencies (otherwise known as memory) and the probability density function (PDF). The simplest quantiser is the scalar quantiser, where input samples are mapped individually to scalar codewords, which are also referred to as code-points or reproduction values [55]. Because the quantisation of the input samples are independent of each other, redundancies between these samples are included as well. If these redundancies can be modelled and reproduced by the decoder, then their inclusion leads to inefficient coding.

Shannon [155] showed that better rate-distortion efficiency can be achieved by quantising vectors (or, blocks) of samples. Block quantisers[3] and transform coders improve the efficiency of scalar quantisation by removing the linear dependencies or correlation between the samples in a block via the Karhunen-Loève transform (KLT), which is a linear decorrelating transform. The decoder can then 'add' the correlation back to the decoded values via the inverse KLT. Also, the block quantiser is *bitrate scalable*, which means that it can scale to any bitrate without any quantiser re-training. However, the prerequisite for optimal block quantisation, namely a purely Gaussian data source with uniform correlation, is rarely ever met with real-world signals. This mismatch degrades the rate-distortion efficiency of the block quantiser [43]. Also, non-linear dependencies remain after decorrelation [108], which suggests that further gains in rate-distortion efficiency are possible.

The vector quantiser, which individually maps input vectors to vector codewords (or, code-vectors), is theoretically the ultimate solution, in terms of achieving the best rate-distortion efficiency. In other words, the vector quantiser can achieve the lowest distortion, at a given rate and dimension, of any quantisation scheme. It can be shown that vector quantisers not only exploit non-linear dependencies, but also outperform scalar quantisation even on memoryless and independent sources [55]. It is because of the freedom to place code-vectors in a higher dimensional space, that gives the vector quantiser ad-

[3]Generally, the term *block quantiser* is also a synonym of *vector quantiser* (such as, in [53]) and this is reflected in the title of this dissertation. However, we now make a distinction between these two terms, by noting in [198], where the fixed-rate transform coding scheme of Huang and Schultheiss [72] is referred to as 'block quantisation'. This contrasts to the vector quantiser, as described in [55, 59].

ditional advantages in matching the source PDF shape and filling space efficiently [103]. However, their computational complexity and memory requirements grow exponentially with rate and dimension and this inherent disadvantage inhibits their use in applications that require high bitrates and dimensionality. Structurally constrained vector quantisers, such as product code vector quantisers [55], alleviate the complexity issues by simplifying the code-vector search, though this results in degraded rate-distortion performance. Compared with the block quantiser, vector quantisers are generally not bitrate scalable.

This dissertation makes several contributions to improving the rate-distortion and/or computational efficiency of block and vector quantisation schemes. Specifically, we examine the block quantisation scheme introduced in [183], which we refer to as the *Gaussian mixture model-based block quantiser*, that derives better rate-distortion performance by accurately estimating the source using a mixture of Gaussian sources and designing multiple KLTs to decorrelate the individual overlapping basis sources, rather than assume a unimodal Gaussian source with uniform correlation. Contributions are made to this quantisation scheme in the context of image coding, by replacing the KLT with the discrete cosine transform (DCT). The resulting *GMM-DCT-based block quantiser* is computationally simpler and achieves comparable performance to the KLT-based one. In addition to this, a new memory-based scheme, which we term the *multi-frame GMM-based block quantiser*, is introduced to further improve the rate-distortion efficiency of the single frame, memoryless scheme by exploiting interframe correlation. We also propose a new product code vector quantiser, called the *switched split vector quantiser* (SSVQ), that achieves better rate-distortion performance and significantly lower computational complexity than the split vector quantiser [123]. We apply and evaluate the performance of the multi-frame GMM-based block quantiser and SSVQ in line spectral frequency (LSF) quantisation in narrowband speech coding, LSF and immittance spectral pair (ISP) quantisation in wideband speech coding, and Mel frequency-warped cepstral coefficient (MFCC) quantisation in distributed speech recognition.

1.3 Dissertation Organisation

1.3.1 Chapter Summary

This dissertation comprises two chapters on the theory and operation of various block and vector quantisation schemes (Chapter 2 and 3, respectively), while the remaining four chapters deal with the application of these schemes in lossy image coding (Chapter 4), narrowband speech coding (Chapter 5), wideband speech coding (Chapter 6), and distributed speech recognition (Chapter 7). The chapters are described in detail below:

- **Chapter 2** provides the theoretical basis of block quantisation, which includes the Karhunen-Loève transform and its decorrelating and energy compaction characteristics, optimum bit allocation, and the shortcomings of block quantisers on data with non-stationary statistics and multimodal probability density functions. This leads to the area of adaptive transform coding, where we provide a review of techniques that have been reported in the image coding literature. Following this, we provide a detailed description of the GMM-based block quantiser, which includes the estimation of PDFs using Gaussian mixture models via the EM algorithm, allocation of bits among and within clusters, and the minimum distortion block quantiser. We then present our two proposed schemes, namely the GMM-DCT-based block quantiser and multi-frame GMM-based block quantiser. Also, a novel technique for encoding fractional (or, non-integer) bits in a fixed-rate block quantiser is presented, accompanied by some algorithms for dealing with non-integer bit allocation.

- **Chapter 3** begins with the theory of vector quantisation, which includes the so-called 'vector quantiser advantages' [103]. This provides the foundation for identifying the source of rate-distortion suboptimality in structurally constrained vector quantisers, such as split and multistage vector quantisers. We analyse the 'advantage' losses in the split vector quantiser specifically, which leads us to the switched split vector quantiser. Through simulations and plots, we show why and by how much the switched split vector quantiser improves upon normal split vector quantisation and relate these to two of the three vector quantiser advantages. The computational complexity and memory requirements of SSVQ are also discussed.

- **Chapter 4** first provides a broad review of quantisation techniques that have been applied to lossy image coding. These include vector quantisation, transform coding, subband coding, and discrete wavelet transform-based coding. The difficulties that are often encountered in subband and wavelet-based image coding involve the expansion of coefficients after the filtering of a finite length signal (such as a row or column). Therefore, a detailed summary of non-causal and non-expansive[4] filtering techniques using the symmetric extension method is given which is complemented with MATLAB simulations that implement subband and wavelet decompositions. The rest of the chapter then presents and discusses the results of our image coding experiments, using the KLT-based and DCT-based block quantiser (as a baseline), the GMM-based block quantiser with integer and non-integer bit allocation, and the GMM-DCT-based block quantiser. Image quality is judged based on peak signal-to-noise ratios (PSNRs) and visual inspection of the decoded images. Finally, we propose a method of reducing the block artifacts that are common in transform coded images, by pre-processing the image using a discrete wavelet transform, extracting vectors using a tree structure that exploits spatial self-similarity, before being quantised by the GMM-based block quantiser.

- **Chapter 5** starts off with the preliminaries of narrowband speech coding, which includes speech production and linear prediction analysis. Following this, we describe the operation of some popular speech coding algorithms that have been adopted in industry and highlight the need for accurate quantisation of the LPC coefficients. Following this, we review the various LPC parameter representations (reflection coefficients, line spectral frequencies, immittance spectral pairs, etc.) and the quantisation schemes that have been investigated in the speech coding literature. The spectral distortion performance of popular schemes, such as scalar quantisers, split vector quantisers and multistage vector quantisers, on LSF vectors from the TIMIT database is also provided. These form a basis of comparison for the multi-frame GMM-based block quantiser and switched split vector quantiser, which are evaluated on LSF vectors from the TIMIT database. Quantisation performance is determined using the average spectral distortion.

- **Chapter 6** begins with the definition of wideband speech and its advantages over toll-quality narrowband speech. We then briefly review the state-of-the-art coding

[4]That is, when filtering n input samples, n output samples are produced.

schemes for wideband speech, such as the Transform Coded Excitation (TCX) coder, and the industry standard coders such as the ITU-T G.722 and ITU-T G.722.2 (also known as AMR-WB). Following this, we report the spectral distortion performance of various quantisation schemes, such as scalar quantisers, the vector quantiser, and the GMM-based block quantiser on the two competing LPC parameter representations in the wideband coding literature: line spectral frequencies (LSFs) and immittance spectral pairs (ISPs). Also, we extrapolate the operating curve of the vector quantiser to derive informal lower bounds on the bitrate required to transparently code LSFs and ISPs. Finally, the rest of the chapter is dedicated to evaluating the multi-frame GMM-based block quantiser and switched split vector quantiser. We compare and contrast their spectral distortion performance on the LSF and ISP representation.

- **Chapter 7** investigates the application of multi-frame GMM-based block quantisers for coding Mel frequency-warped cepstral coefficient (MFCC) vectors for distributed speech recognition (DSR). The focus is on bitrate scalable quantisation schemes. We begin the chapter with some background theory on automatic speech recognition. Following this, we provide a general review of client/server-based speech recognition systems and the various types of modes (NSR and DSR) that have been proposed and reported in the literature. We also briefly describe the Aurora-2 DSR experimental framework, which will be used extensively to evaluate the performance and robustness to noise of the various DSR schemes. As well as examining the recognition performance of DSR using the multi-frame GMM-based block quantiser, we compare it with that using scalar quantisers, the vector quantiser, and the memoryless GMM-based block quantiser.

- **Chapter 8** collates the conclusions made from our evaluation of the block and vector quantisation schemes, in the four application areas that were considered in this dissertation. In addition to this, we discuss some topics for future research.

1.3.2 Composite Literature Review

Since this dissertation covers several distinct topics and application areas, each chapter (apart from the Introduction and Conclusion) has its own literature review. Therefore, the literature review of this dissertation is of a composite nature and comprises:

- Section 2.2, which covers the basics of block quantisation, including the Karhunen-Loève transform and bit allocation; Section 2.3 which covers the discrete cosine transform; and Sections 2.4 and 2.5 which describe adaptive transform coders and the GMM-based block quantiser;

- Section 3.3, which covers the Linde-Buzo-Gray algorithm [100] for the design of a vector quantiser; Section 3.4 which describes the vector quantiser advantages [103]; Section 3.5 which lists the practical limitations of unconstrained vector quantisers; and Sections 3.6 and 3.7 which cover the different types of structurally constrained vector quantisers;

- Section 4.4, which covers the popular schemes used in image coding such as vector quantisation, transform coding, subband coding, and wavelet-based coding and Section 4.5, which provides a comprehensive summary and analysis of non-expansive filtering techniques for the subband decomposition of images;

- Section 5.2, which covers preliminary topics such as speech production and linear prediction analysis of speech; Section 5.3, which reviews the numerous linear prediction speech coding algorithms and Section 5.4, which covers the various LPC parameter representations used for representing short-term correlation information in LPC-based speech coders;

- Section 6.2, which defines wideband speech and discusses its improved quality as well as coders that have been investigated in the literature; Section 6.3, which describes the operation of industry standard and state-of-the-art wideband speech coders; and Section 6.4.2 which reviews the LPC parameter quantisation literature; and

- Section 7.2, which covers the preliminary topics on automatic speech recognition such as feature extraction and pattern recognition and Section 7.3, which covers the different modes of client-server speech recognition systems.

1.4 Contributions and Resulting Publications

1.4.1 Contributions Made in this Dissertation

The work presented in this dissertation makes contributions in several different areas of signal coding. These contributions are summarised as follows:

1. Section 2.6: A modified GMM-based block quantiser, termed the *GMM-DCT-based block quantiser*, that replaces the KLT with the DCT, is proposed for image coding. Because images exhibit a high degree of correlation and possess strong Gauss-Markov properties, the DCT is an excellent alternative to the KLT and achieves similar decorrelation and energy compaction properties to the latter. Furthermore, the DCT is data independent, which permits the realisation of a computationally simpler scheme.

2. Section 2.7.3: A GMM-based block quantiser that utilises interframe memory, termed the *multi-frame GMM-based block quantiser*, is proposed for LPC parameter quantisation in narrowband and wideband speech coding, as well as Mel frequency-warped cepstral coefficients for distributed speech recognition applications.

3. Section 2.8: A novel method of encoding fractional bits in a fixed-rate GMM-based block quantisation framework is described. The proposed method utilises a generalisation of the positional number system and allows the use and encoding of fractional numbers of bits (or, integer number of quantiser levels) at a fixed bitrate, as opposed to variable bitrate. By using an integer number of levels, which has finer granularity than using an integer number of bits, better utilisation of the bit budget is possible.

4. Section 3.8: A new product code vector quantisation scheme, termed the *switched split vector quantiser* (SSVQ), is proposed for the quantisation of LPC parameters for narrowband and wideband speech coding. The SSVQ reduces the loss in rate-distortion performance in split vector quantisers that are due to the vector partitioning. The resulting scheme encompasses improvements in both rate-distortion and computational efficiency, at the expense of increased memory requirements.

5. Section 4.9: A block reduction technique for the GMM-based block quantisation of images is proposed, which uses the discrete wavelet transform as a pre-processing

step. Blocks of wavelet coefficients are extracted using a tree structure, similar to that used in the embedded zerotree wavelet coder [158] and the wavelet tree-based coder of [98], which exploits spatial self-similarity.

6. Section 6.4.5: An informal lower bound on the bitrate required to achieve transparent coding of wideband LSFs and ISPs is derived. The difference between the performance of LSFs and ISPs is later confirmed by all joint block and vector quantisation experiments.
7. Section 6.4.8: A weighted Euclidean distance measure, based on the fixed weighting system of a similar distance measure used in [123] for narrowband LSFs, is introduced to improve the quantisation performance of the switched split vector quantiser on wideband LSF vectors.

1.4.2 Journal Articles

Note: All references in this dissertation follow the IEEE reference and citation guidelines prescribed in [1].

1. K.K. Paliwal and S. So, “A fractional bit encoding technique for the GMM-based block quantisation of images”, *Digital Signal Processing*, vol. 15, pp. 255–275, May 2005.
2. K.K. Paliwal and S. So, “Low complexity GMM-based block quantisation of images using the discrete cosine transform”, *Signal Processing: Image Communication*, vol. 20, pp. 435–446, June 2005.
3. S. So and K.K. Paliwal, “Scalable distributed speech recognition using Gaussian mixture model-based block quantisation”, submitted to *Speech Commun.*, 2005.
4. S. So and K.K. Paliwal, “Multi-frame GMM-based block quantisation of line spectral frequencies”, to appear in *Speech Commun.*, 2005.
5. S. So and K.K. Paliwal, “Efficient product code vector quantisation using the switched split vector quantiser”, submitted to *Digital Signal Processing*, Nov. 2004.
6. S. So and K.K. Paliwal, “A comparative study of LPC parameter representations and quantisation schemes in wideband speech coding”, submitted to *Digital Signal*

Processing, May 2005.

7. S. So and K.K. Paliwal, “A comparison of LPC parameter representations for wideband speech coding”, to be submitted to *IEEE Signal Processing Lett.*, 2005.

1.4.3 Conference Articles

1. S. So and K.K. Paliwal, “Efficient block coding of images using Gaussian mixture models”, in *Proc. Fourth Australasian Workshop on Signal Processing and Applications 2002*, Brisbane, Australia, Sept. 2002, pp. 71–74.

2. K.K. Paliwal and S. So, “Low complexity GMM-based block quantisation of images using the discrete cosine transform”, accepted by ICIP 2003[5].

3. K.K. Paliwal and S. So, “Low complexity Gaussian mixture model-based block quantisation of images”, in *Proc. Microelectronic Engineering Research Conference*, Brisbane, Australia, Nov. 2003.

4. K.K. Paliwal and S. So, “Multiple frame block quantisation of line spectral frequencies using Gaussian mixture models”, in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, Montreal, Canada, 2004, pp. I-149–152.

5. K.K. Paliwal and S. So, “Scalable distributed speech recognition using multi-frame GMM-based block quantization”, in *Proc. Int. Conf. Spoken Language Processing*, Jeju, Korea, Oct. 2004.

6. S. So and K.K. Paliwal, “Efficient vector quantisation of line spectral frequencies using the switched split vector quantiser”, in *Proc. Int. Conf. Spoken Language Processing*, Jeju, Korea, Oct. 2004.

7. S. So and K.K. Paliwal, “Multi-frame GMM-based block quantisation of line spectral frequencies for wideband speech coding”, in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, vol. I, Philadelphia, USA, 2005, pp. 121–124.

8. S. So and K.K. Paliwal, “Switched split vector quantisation of line spectral frequencies for wideband speech coding”, to appear in *Proc. European Conf. Speech Communication and Technology* (Eurospeech), Lisbon, Portugal, Sept 2005.

[5]This paper was accepted by ICIP 2003 but was later withdrawn due to the authors’ inability to attend the conference.

9. S. So and K.K. Paliwal, "A comparison of LSF and ISP representations for wideband LPC parameter coding using the switched split vector quantiser", to appear in *Proc. IEEE Int. Symp. Signal Processing and Applications* (ISSPA), Sydney, Australia, Aug 2005.

Chapter 2

Efficient Block Quantisation

2.1 Abstract

In this chapter, we first review the block quantiser and how it has been applied as a less complex alternative to vector quantisation. The energy compaction and decorrelation characteristics of the Karhunen-Loève transform (KLT) and its role in the block quantiser are also covered, with a mention of its limitations in practice and how the discrete cosine transform (DCT) is used instead as a fast, source-independent alternative transform for image data. The block quantiser makes use of the assumption that the statistical dependencies, such as correlation, are uniform throughout the vector space, hence a single, global KLT is expected to decorrelate all vectors. Also, the optimality of the KLT for quantisation is itself highly dependent on how close the underlying PDF is to a Gaussian. However, after analysing the PDFs of real-life image data, it becomes clear that these assumptions are not valid and the mismatch causes performance degradation in the single transform block quantiser. This leads to the concept of adaptive transform coders and we provide a literature review of these techniques that have appeared in the image coding area. The vector space is partitioned into regions that are relatively stationary and the vectors within the region are quantised using a transform coder. It is desirable for the vector space partitioning, transformation, and quantisation to be designed jointly in order to minimise distortion. However, these adaptive transform coders are not bitrate scalable. That is, the bitrate cannot be changed unless they are re-trained.

Next, we review a new paradigm of efficient transform coding design that has appeared in the literature and is based on source modelling. The GMM-based block quantiser, which we describe in detail, uses a Gaussian mixture model to parametrically estimate the source PDF. The scheme is bitrate scalable and soft decision-based, where each vector is quantised multiple times and the best representation (minimum distortion) is determined which can be computationally complex for large dimensional vectors. We then present our modification of the GMM-based block quantiser for image coding, where the Gauss-Markov nature of image data has made the DCT the preferred transform. Because the DCT is source independent, only one transformation operation needs to be used. Furthermore, because the DCT basis functions are fixed, no inverse DCT needs to be performed and the minimum distortion calculation can be performed within the DCT domain. As a result, the GMM-DCT-based block quantiser is considerably faster than the GMM-based block quantiser. We also present a new GMM-based block quantiser that exploits memory between multiple blocks or frames, resulting in better overall rate-distortion performance than the memoryless scheme.

Traditional bit allocation for block quantisers has been done in a bit-wise fashion, which presents no problem in quantiser index encoding. Fractional bits from the closed-form bit allocation formula are truncated and negative bits are set to zero. The use of quantiser levels, rather than quantiser bits, allows a finer granularity for allocation since the allocated levels in bit-wise allocation are constrained to be powers of two. Also, the cluster block quantisers within the GMM-based block quantiser are often allocated with a fractional bitrate, thus there exists the need for scalar quantisers to operate at an integer number of levels. However, binary encoding quantiser level indices from a block at fixed-rate with non-uniform allocation is not as straightforward. Therefore, we present an elegant and simple fractional bit encoding technique that maps quantiser level indices to a single integer for each block. Also, heuristic compensation techniques for levels-based scalar quantisers are described which deal with the truncation and negative allocation issues.

Publications resulting from this research: [126, 127, 128, 129, 130, 131, 165, 167, 168, 170, 171, 174]

2.2 Introduction to Block Quantisation

One of the most important results from Shannon's rate-distortion theory is that quantising vectors is more efficient than quantising scalars [55]. From a theoretical perspective, vector quantisation is the best coding solution. One advantage of vector quantisers over scalar quantisers is the exploitation of statistical dependency between consecutive samples [55]. Scalar quantisers, on the other hand, operate on individual data samples only and thus do not take into account the dependencies that exist between these samples. Another important result from the rate-distortion theory is that the performance of a vector-based quantiser (such as the block quantiser and vector quantiser), in terms of distortion, improves as the vector dimension is increased [59]. However, unconstrained vector quantisers operating at high bitrates and large vector dimension, require an exorbitant amount of memory and computations which, depending on the application, may make them impractical.

Transform coding has been used as a less complex alternative to unconstrained vector quantisation[1] [10]. First proposed by Kramer and Mathews [89] and mathematically analysed by Huang and Schultheiss [72], it involves grouping n correlated samples into a *vector* or *block*, $\boldsymbol{x} = \{x_i\}_{i=1}^{n}$, linearly transforming each block to a new set of co-ordinates, and then scalar quantising each component independently.

Block quantisation [72] can be considered a special case of transform coding, where quantisation is performed by non-uniform scalar quantisers of fixed-rate [198]. Figure 2.1 shows a schematic of the block quantisation procedure. This contrasts with other transform coders, where uniform scalar quantisers are used in conjunction with variable-rate entropy coders. In fact, it was shown by Wood [199], that in the high resolution sense, the entropy of the output of a uniform scalar quantiser is lower than that of a fixed-rate, optimum non-uniform scalar quantiser, regardless of the source PDF. Therefore, uniform scalar quantisation followed by entropy coding will generally be more efficient than non-uniform scalar quantisers [199, 59], though the variable nature of the bitrate requires increased complexity and causes problems with error propagation and buffer underflows/overflows [55, 56]. The JPEG image compression standard is an example of a variable-rate transform coder, where discrete cosine transform (DCT) coefficients are coded with scalar quantisers

[1]A block quantiser can be considered equivalent to constrained vector quantisation of transform coefficients.

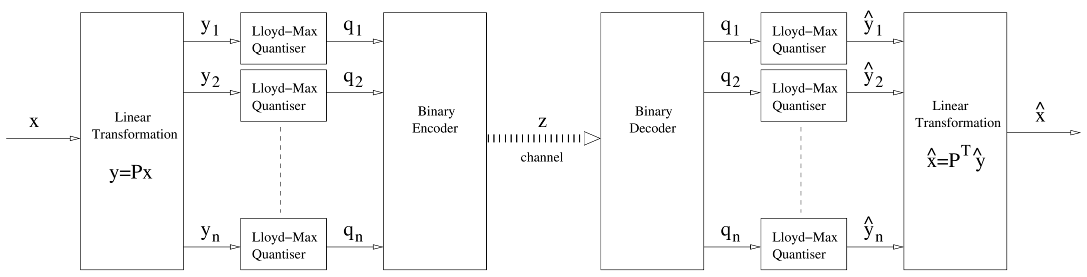


Figure 2.1: Schematic of a typical block quantiser

of uniform step sizes, followed by runlength and Huffman encoding [196].

The role of the transform is to remove linear dependencies between the components of the block before they are quantised, since it is generally observed in practice that the more independent the transformed coefficients, the more efficient scalar quantising becomes [55]. Therefore, block quantisation allows the exploitation of correlation[2] across consecutive samples, similar to what vector quantisers do, and this leads to more efficient coding[3] [34].

One particular advantage of using a linear transform is that it may be more convenient to quantise the data in another co-ordinate system. This allows bits to be allocated to each component, relative to its importance in influencing the subjective quality of the reconstructed signal [55]. For example, consider the coding of audio using a Modified Discrete Cosine Transform (MDCT) based perceptual coder [177]. Since the MDCT is a Fourier-based transform, the cosine bases are frequency-related. It is well-known in human auditory perception studies that the ear: processes sound via critical bandwidth filters; has a non-linear frequency response to loudness; and masks low-level tones in the presence of a stronger tone [117]. Therefore, MDCT-based perceptual audio coders transform the audio into a frequency-related domain where it is more convenient to exploit spectral-based psychoacoustic phenomena and shape the quantisation noise in order to improve the subjective quality of the reconstructed audio [177].

Other advantages of block quantisation, when compared with vector quantisation, include bitrate scalability and relatively low and fixed computational and memory require-

[2]It should be noted that correlation is a linear statistical dependency. Unconstrained vector quantisers, however, can exploit non-linear statistical dependencies as well [55].

[3]Despite the common notion that scalar quantising independent or decorrelated data is better than quantising correlated data, it has been shown recently [44] that decorrelation alone is insufficient for optimal fixed-rate and variable-rate coding.

ments. Bitrate scalability is the ability to alter the operating bitrate 'on-the-fly', without requiring any quantiser re-design or re-training. As will be mentioned later in this chapter, the allocation of bits in block quantisation can be expressed in closed-form[4], hence any change in the bitrate will require only a recalculation of the bit allocation only. The scalar quantisers remain fixed. On the other hand, vector quantisers are designed for a specific bitrate only, as it is related to the size of the codebook. In order to change the bitrate, either the vector quantiser needs to be re-designed for this lower bitrate or it needs to store and use different codebooks for different bitrates. Also, the block quantiser has relatively low and fixed computational and memory requirements since it uses non-uniform scalar quantisers, which can be implemented as a uniform scalar quantiser (rounding function) with appropriate companding and expanding functions [183]. While for the vector quantiser, search complexity and memory requirements increase exponentially with the number of bits [59].

2.2.1 The Karhunen-Loève Transform

An orthogonal linear transform projects a *zero-mean*, n-dimension vector, $\boldsymbol{x}$, onto a series of orthogonal basis vectors that span the transform space and form the rows of the transformation matrix, $\boldsymbol{P}$. The linear transformation is expressed as:

$$\boldsymbol{y} = \boldsymbol{P}\boldsymbol{x} \tag{2.1}$$

where $\boldsymbol{y}$ is the transformed vector containing the transform coefficients, $\{y_i\}_{i=1}^{n}$. The inverse linear transformation, where the transformed vector is converted back to the original co-ordinates, is expressed as:

$$\boldsymbol{x} = \boldsymbol{P}^{-1}\boldsymbol{y} \tag{2.2}$$

Because of the orthogonality condition where $\boldsymbol{P}^T = \boldsymbol{P}^{-1}$ [55], (2.2) can be rewritten as:

$$\boldsymbol{x} = \boldsymbol{P}^T\boldsymbol{y} \tag{2.3}$$

[4]As well as closed form expressions, bit allocation can be performed using fixed-slope and greedy algorithms as well [55, 147].

The covariance matrix[5] of the transformed vectors, $\boldsymbol{\Sigma_y}$, is given as:

$$\begin{aligned}
\boldsymbol{\Sigma_y} &= E[\boldsymbol{y}\boldsymbol{y}^T] \\
&= E[\boldsymbol{Px}(\boldsymbol{Px})^T] \\
&= \boldsymbol{P}E[\boldsymbol{x}\boldsymbol{x}^T]\boldsymbol{P}^T \\
&= \boldsymbol{P}\boldsymbol{\Sigma_x}\boldsymbol{P}^T \qquad (2.4)
\end{aligned}$$

where $\boldsymbol{\Sigma_x}$ is the covariance of the original vector source and $E[\bullet]$ is the expectation operator. As mentioned earlier, one desirable attribute of the linear transform is to be able to decorrelate the data. That is, the covariance matrix of the transformed vectors should be diagonal. By substituting $\boldsymbol{U} = \boldsymbol{P}^T$, (2.4) can be expressed as an eigenvalue decomposition, shown in (2.5) below, where the source covariance matrix, $\boldsymbol{\Sigma}_x$, is diagonalised by the matrix, $\boldsymbol{U}$, whose columns consist of the eigenvectors, $\{\boldsymbol{v}_i\}_{i=1}^n$, of the source [72]:

$$\boldsymbol{\Sigma_y} = \boldsymbol{U}^T\boldsymbol{\Sigma_x}\boldsymbol{U} \qquad (2.5)$$

$$\text{where } \boldsymbol{\Sigma_y} = \begin{bmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda_n \end{bmatrix} \qquad (2.6)$$

The main diagonal of $\boldsymbol{\Sigma_y}$ consists of the eigenvalues of the source, $\{\lambda_i\}_{i=1}^n$, which are also the variances of each transform coefficient. These eigenvalues (and their corresponding eigenvectors) are rearranged in descending order, from largest to smallest. This type of transform is called the Karhunen-Loève transform[6] (KLT). The KLT projects the source vectors onto a new basis set, called the *principal components*, which are effectively the eigenvectors of the source covariance matrix. This means the KLT is a *source-dependent* transform. That is, the KLT needs to be calculated for different sources. For any source, there will always exist at least one unique KLT since covariance matrices are always symmetric and positive semidefinite [56]. That is, their eigenvalues are non-negative [65].

[5]Since we are always assuming zero-mean vectors, the covariance matrix becomes equivalent to the autocorrelation matrix, $\boldsymbol{R_x}$, hence both terms can be used interchangeably here.

[6]Strictly speaking, the transform discovered by Karhunen [82] and Loève [102] operated on continuous time signals. The analysis of discrete variables is attributed to Hotelling [71], who called it *method of principal components* [198]. Therefore, the discrete KLT is also known as the eigenvector transform, Hotelling transform, or principal component analysis (PCA) [198, 56]. From here on, use of the terms 'KLT' or 'Karhunen-Loève Transform' refer to the discrete version.

Eigenvalue Decomposition Using Jacobi's Method

Calculating the transformation matrix for the KLT requires finding n distinct eigenvectors. There are various methods of finding eigenvectors and eigenvalues available in linear algebra. One of them, called Jacobi's method, works on square and real symmetric matrices only. Since covariance matrices are always square and symmetric, Jacobi's method is a simple and well-suited algorithm for matrix diagonalisation, though it is not as robust to ill-conditioned systems as other methods associated with singular value decomposition (SVD).

Essentially, for each iteration, a rotation matrix, $\boldsymbol{R}_i$, is designed to 'zero out' a non-diagonal element, $\boldsymbol{A}(p,q)$ via [88]:

$$\boldsymbol{A}_1 = \boldsymbol{R}_1^T \boldsymbol{A} \boldsymbol{R}_1 \tag{2.7}$$

After k iterations through Jacobi's method, we have [88]:

$$\boldsymbol{A}_k = \boldsymbol{R}_k^T \boldsymbol{R}_{k-1}^T \dots \boldsymbol{R}_1^T \boldsymbol{A} \boldsymbol{R}_1 \dots \boldsymbol{R}_{k-1} \boldsymbol{R}_k \tag{2.8}$$

where $\boldsymbol{A}_k$ becomes a diagonal matrix. Setting $\boldsymbol{U} = \prod_{i=1}^{k} \boldsymbol{R}_i$, the columns will contain the eigenvectors and the corresponding eigenvalues will appear on the diagonal of $\boldsymbol{A}_k$.

Jacobi's method is described below where the matrix, $\boldsymbol{A}$, is square, symmetric, and positive semidefinite [88].

Step 1:

Initialise an identity matrix, $\boldsymbol{U}$, and set $\epsilon = 0.001$.

Step 2:

Find the element with the largest magnitude, $\boldsymbol{A}(p,q)$, that is not on the main diagonal (ie. $p \neq q$).

Step 3:

Setting:

$$f = -\boldsymbol{A}(p,q) \tag{2.9}$$

$$\text{and } g = \frac{\boldsymbol{A}(p,p) - \boldsymbol{A}(q,q)}{2} \tag{2.10}$$

calculate $\sin\theta$ and $\cos\theta$ using the following formulae:

$$\sin\theta = \frac{h}{\sqrt{2(1+\sqrt{1-h^2})}} \tag{2.11}$$

$$\cos\theta = \sqrt{1-\sin^2\theta} \tag{2.12}$$

$$\text{where } h = \begin{cases} \frac{f}{\sqrt{f^2+g^2}} & \text{, if } g \geq 0 \\ -\frac{f}{\sqrt{f^2+g^2}} & \text{, otherwise} \end{cases} \tag{2.13}$$

Step 4:

Form the rotation matrix, $\boldsymbol{R}$, by setting it to an identity matrix and setting the following four elements:

$$\boldsymbol{R}(p,p) = \cos\theta \tag{2.14}$$

$$\boldsymbol{R}(p,q) = \sin\theta \tag{2.15}$$

$$\boldsymbol{R}(q,p) = -\sin\theta \tag{2.16}$$

$$\boldsymbol{R}(q,q) = \cos\theta \tag{2.17}$$

Step 5:

$$\boldsymbol{U} = \boldsymbol{U}\boldsymbol{R} \tag{2.18}$$

$$\boldsymbol{A} = \boldsymbol{R}^T\boldsymbol{A}\boldsymbol{R} \tag{2.19}$$

Step 6:

If the largest magnitude element not on the main diagonal of $\boldsymbol{A}$ is less than ϵ, stop. Else, go to Step 2.

When used in the KLT, the eigenvectors are sorted such that their corresponding eigenvalues are in descending order (ie. $\lambda_1 > \lambda_2 > \ldots > \lambda_n$).

Energy Compaction and Linear Approximation

The KLT tends to rotate the vector space such that the eigenvector with the largest corresponding eigenvalue, is aligned with the first dimension [56]. Since the KLT is a unitary transform where Euclidean distances are preserved, the total variance of all the components remains unchanged after transformation [198, 55]. However, because the eigenvalues, (and their corresponding eigenvectors) are naturally ordered in a descending fashion in the KLT (ie. $\lambda_1 > \lambda_2 > \ldots > \lambda_n$), the distribution of the coefficient variances become skewed with the first eigenvector having the most variance. The compaction of variance or energy of the source into the first few components is another property of KLT-type of transformations that is useful for block quantisation since the skewed variance distribution allows a prioritisation of the quantiser bit budget.

Consider the case of linear approximation (which in the pattern classification literature, is known as *dimensionality reduction*), where $n-m$ consecutive transform coefficients are truncated and the remaining m coefficients are kept. That is:

$$\hat{\boldsymbol{y}} = \boldsymbol{I}_m \boldsymbol{y} \tag{2.20}$$

where $\boldsymbol{I}_m$ keeps the first m transform coefficients and zeros out the rest. Assuming the distortion is MSE, it can be shown that the Karhunen-Loève transform is the optimum transform for minimising MSE after linear approximation [34]. In other words, the mean squared error distortion:

$$D_{mse} = E[\|\boldsymbol{x} - \hat{\boldsymbol{x}}\|^2] \tag{2.21}$$

will be minimised when the transformation matrix, $\boldsymbol{P}$, is that of the KLT. The proof is given below and has been adapted from [143].

Let us assume that we have a set of n basis vectors, $\{\boldsymbol{\phi}_i\}_{i=1}^n$, which are orthonormal. That is:

$$\langle \boldsymbol{\phi}_i, \boldsymbol{\phi}_j \rangle = \delta_{i,j} \tag{2.22}$$

where $\langle \bullet, \bullet \rangle$ is the inner product and $\delta_{i,j}$ is the Kronecker delta. Any vector, $\boldsymbol{x}$, can be expressed as a linear combination of the basis functions:

$$\boldsymbol{x} = \sum_{i=1}^{n} c_i \boldsymbol{\phi}_i \tag{2.23}$$

$$\text{where } c_i = \langle \boldsymbol{x}, \boldsymbol{\phi}_i \rangle$$

Here c_i are the transform coefficients. If we use only the first m transform coefficients to reconstruct our approximated vector:

$$\hat{\boldsymbol{x}} = \sum_{i=1}^{m} c_i \boldsymbol{\phi}_i \tag{2.24}$$

then the error vector between this and the original will be:

$$\begin{aligned}
\boldsymbol{x} - \hat{\boldsymbol{x}} &= \sum_{i=1}^{n} c_i \boldsymbol{\phi}_i - \sum_{i=1}^{m} c_i \boldsymbol{\phi}_i \\
&= \sum_{i=m+1}^{n} c_i \boldsymbol{\phi}_i \\
E[\|\boldsymbol{x} - \hat{\boldsymbol{x}}\|^2] &= E\left[\left\langle \sum_{i=m+1}^{n} c_i \boldsymbol{\phi}_i, \sum_{i=m+1}^{n} c_i \boldsymbol{\phi}_i \right\rangle\right]
\end{aligned}$$

Due to the constraint (2.22):

$$\begin{aligned}
E[\|\boldsymbol{x} - \hat{\boldsymbol{x}}\|^2] &= E\left[\sum_{i=m+1}^{n} |c_i|^2\right] \\
&= E\left[\sum_{i=m+1}^{n} |\langle \boldsymbol{x}, \boldsymbol{\phi}_i \rangle|^2\right] \qquad (2.25)
\end{aligned}$$

Equation (2.25) is a quadratic form, thus it can be reduced down to:

$$\begin{aligned}
E[\|\boldsymbol{x} - \hat{\boldsymbol{x}}\|^2] &= E\left[\sum_{i=m+1}^{n} \boldsymbol{\phi}_i^T \boldsymbol{x}\boldsymbol{x}^T \boldsymbol{\phi}_i\right] \\
&= \sum_{i=m+1}^{n} \boldsymbol{\phi}_i^T E[\boldsymbol{x}\boldsymbol{x}^T] \boldsymbol{\phi}_i \\
&= \sum_{i=m+1}^{n} \boldsymbol{\phi}_i^T \boldsymbol{R_x} \boldsymbol{\phi}_i \qquad (2.26)
\end{aligned}$$

where $\boldsymbol{R_x}$ is the autocorrelation matrix[7] of $\boldsymbol{x}$. The aim is to minimise $E[\|\boldsymbol{x} - \hat{\boldsymbol{x}}\|^2]$ under the constraint (2.22), which can be performed using Lagrangian minimisation:

$$\frac{\partial}{\partial \boldsymbol{\phi}_k}\left[\sum_{i=m+1}^{n} \boldsymbol{\phi}_i^T \boldsymbol{R_x} \boldsymbol{\phi}_i - \lambda_i \langle \boldsymbol{\phi}_i, \boldsymbol{\phi}_i \rangle\right] = 0 \tag{2.27}$$

[7]Or covariance matrix, $\boldsymbol{\Sigma_x}$, of the zero-mean vector, $\boldsymbol{x}$

where λ_i is the Lagrange multiplier. Using the following properties for finding derivatives of vectors and matrices:

$$\begin{aligned}\frac{\partial}{\partial \boldsymbol{\phi}} \boldsymbol{\phi}^T \boldsymbol{R_x} \boldsymbol{\phi} &= 2\boldsymbol{R_x}\boldsymbol{\phi} \\ \frac{\partial}{\partial \boldsymbol{\phi}} \langle \boldsymbol{\phi}, \boldsymbol{\phi} \rangle &= 2\boldsymbol{\phi}\end{aligned}$$

Equation (2.27) becomes:

$$\begin{aligned}2\boldsymbol{R_x}\boldsymbol{\phi}_k - 2\lambda_k \boldsymbol{\phi}_k &= 0 \\ (\boldsymbol{R_x} - \lambda_k \boldsymbol{I})\boldsymbol{\phi}_k &= 0 \qquad (2.28)\end{aligned}$$

where $\boldsymbol{I}$ is the identity matrix. Equation (2.28) is the typical eigenvalue problem which is solved when $\{\boldsymbol{\phi}_k\}_{k=1}^m$ are the eigenvectors of the autocorrelation matrix, $\boldsymbol{R_x}$. The Lagrange multipliers, $\{\lambda_k\}_{k=1}^m$, become the corresponding eigenvalues. Therefore, the KLT minimises the MSE for linear approximation. It is interesting to note that the above proof is not dependent on the distribution of the vectors, $\boldsymbol{x}$ [115].

KLT Optimality of Correlated Gaussian Sources for Quantisation

Even though the KLT is the best linear orthogonal transform in the mean squared sense, when coefficients are truncated in linear approximation and the remaining coefficients are retained with infinite precision, this optimality does not necessarily hold after the transform coefficients have been quantised [56, 34]. This was shown by Effros *et al.* [44], where they proved that decorrelation is not only insufficient, but is also not necessary for optimality, in the high resolution sense[8]. Also, there are sources where the KLT produces independent coefficients, yet the transform coder is suboptimal [44]. Huang and Schultheiss [72] showed that, under high resolution quantiser approximations, it can be shown that the KLT is optimal if the input vectors are jointly Gaussian. From [72], the mean squared error incurred by Lloyd-Max non-uniform scalar quantisers in block quantisation is given

[8]High resolution analysis is one of two asymptotic methods of analysing and obtaining closed-form expressions of the performance of quantisers. The bitrate is assumed to be very high and the quantiser cells and average distortion are small which allows us to assume uniform densities within each cell. Shannon's rate-distortion analysis assumes fixed and finite bitrate and high vector dimension [59].

by:

$$\begin{aligned} D_{mse} &= E[\|\boldsymbol{y} - \hat{\boldsymbol{y}}\|^2] &(2.29)\\ &= K(\bar{b})2^{-2\bar{b}} \left(\prod_{i=1}^{n} \sigma_i^2 \right)^{\frac{1}{n}} &(2.30) \end{aligned}$$

where $\bar{b}$ is the average bitrate, σ_i^2 is the variance of the ith component, and $K(\bar{b})$ is a quantiser-related function (shown in Figure 2.7) which is asymptotically equal to $\frac{\pi\sqrt{3}}{2}$ for Gaussian sources [199, 59]. In order to minimise D_{mse}, the product $\prod_{i=1}^{n} \sigma_i^2$ must be minimised as it is the only variable quantity. Using Hadamard's inequality[9] ($\prod_{i=1}^{n} \sigma_i^2 \geq |\boldsymbol{\Sigma_y}|$) [34]:

$$D_{mse} \geq K(\bar{b})2^{-2\bar{b}} |\boldsymbol{\Sigma_y}|^{\frac{1}{n}} \tag{2.31}$$

When the transform is the KLT, $\boldsymbol{\Sigma_y}$ is diagonal and hence the inequality becomes an equality [34]:

$$D_{mse} = K(\bar{b})2^{-2\bar{b}} |\boldsymbol{\Sigma_x}|^{\frac{1}{n}} \tag{2.32}$$

Therefore, the KLT is the best transform, in the high resolution sense, for minimising the MSE distortion when the input vectors are jointly Gaussian. Huang *et al.* [72] showed that for fixed-rate quantisation, Gaussian-matched Lloyd-Max scalar quantisers were optimal when minimising the MSE for each component separately. Segall [154] later showed that the best scalar quantiser for minimising MSE over all components, on average and assuming high resolution, was also the Lloyd-Max scalar quantiser. And finally, Goyal *et al.* [57] proved the optimality of the KLT for correlated Gaussian sources *without making any high resolution assumptions* [44].

In order to appreciate the importance of the KLT in improving the efficiency of scalar quantising correlated Gaussian sources, a computer simulation was performed where two dimensional random vectors were generated, as shown in Figure 2.2. The vectors are jointly Gaussian with the following covariance matrix:

$$\boldsymbol{\Sigma_x} = \begin{bmatrix} 4.5 & 4.0 \\ 4.0 & 6.0 \end{bmatrix} \tag{2.33}$$

[9]Hadamard's inequality says that for a positive semidefinite matrix, the product of the elements on the main diagonal is greater than the determinant of the matrix, or is equal if the matrix is diagonal [34].

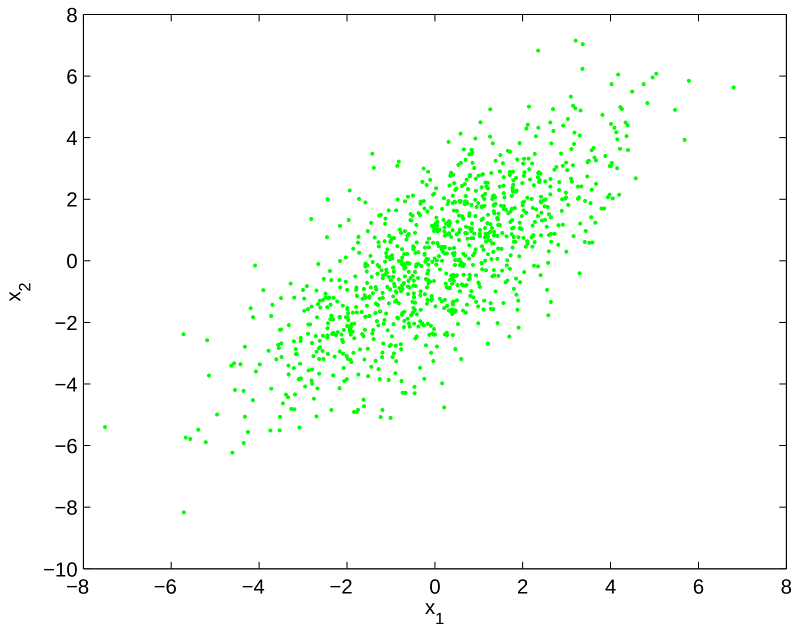


Figure 2.2: Scatter diagram of joint Gaussian random source with covariance matrix given by (2.33)

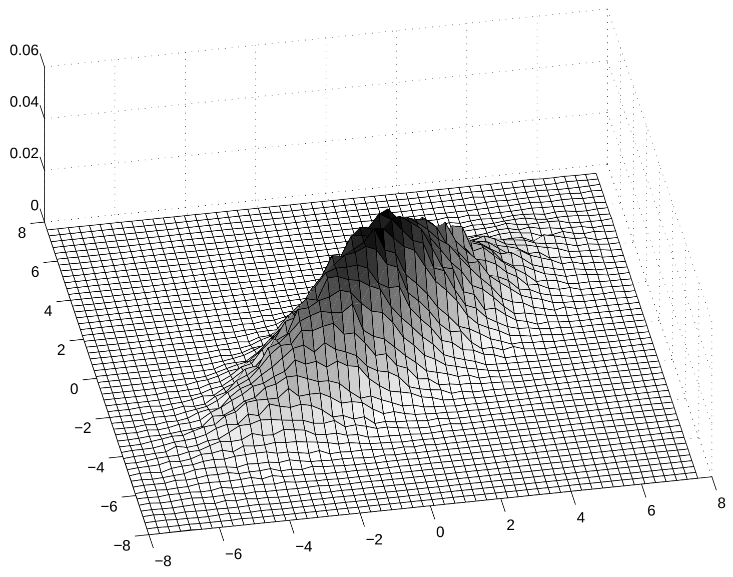


Figure 2.3: Estimated PDF of joint Gaussian random source with covariance matrix given by (2.33)

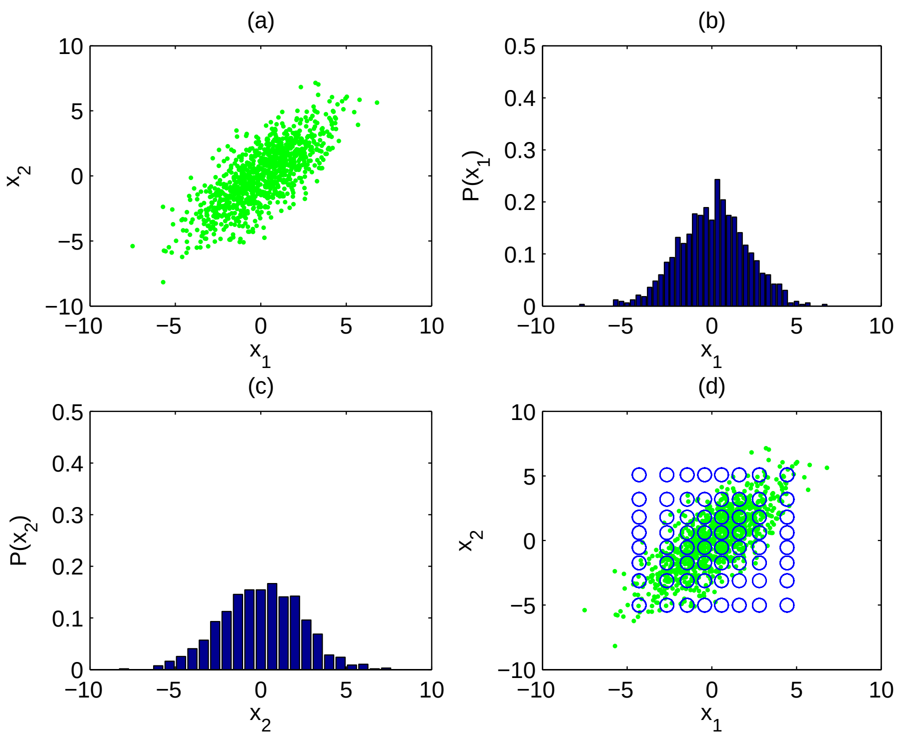


Figure 2.4: Block quantisation using 3 bits/sample with no KLT applied (SNR=14.70 dB); (a) Scatter diagram of joint Gaussian random source with covariance matrix given by (2.33); (b) Estimated marginal PDF of component x_1; (c) Estimated marginal PDF of component x_2; (d) An overlay of the lattice of reproduction points (circles) on original data.

Figure 2.3 shows the estimated PDF[10] of the data. It can be seen in Figure 2.3 that the distribution has a certain 'tilt', indicative of the statistical dependency that exists between the two components, x_1 and x_2.

If no KLT is applied and each component is quantised using a 3 bit Lloyd-Max non-uniform scalar quantiser, the signal-to-noise ratio (SNR) of this coder is 14.70 dB. Figures 2.4(b) and (c) show the estimated marginal PDFs of x_1 and x_2 where most of the mass is contained within the range of $-5 \ldots 5$ and this determines the range of the Lloyd-Max scalar quantisers. In Figure 2.4(d), the lattice of reproduction points of the two scalar quantisers is overlaid on top of the original vectors. It can be observed that there is a significant fraction of reproduction points falling in areas which do not contain any data points. Thus there is a 'wastage' of reproduction points since the correlation, represented

[10]There are, in fact, many methods for estimating the PDF. The method we have adopted in this work uses normalised histograms. In order to simplify the terminology, we refer to these normalised histograms as estimated PDFs throughout the dissertation.

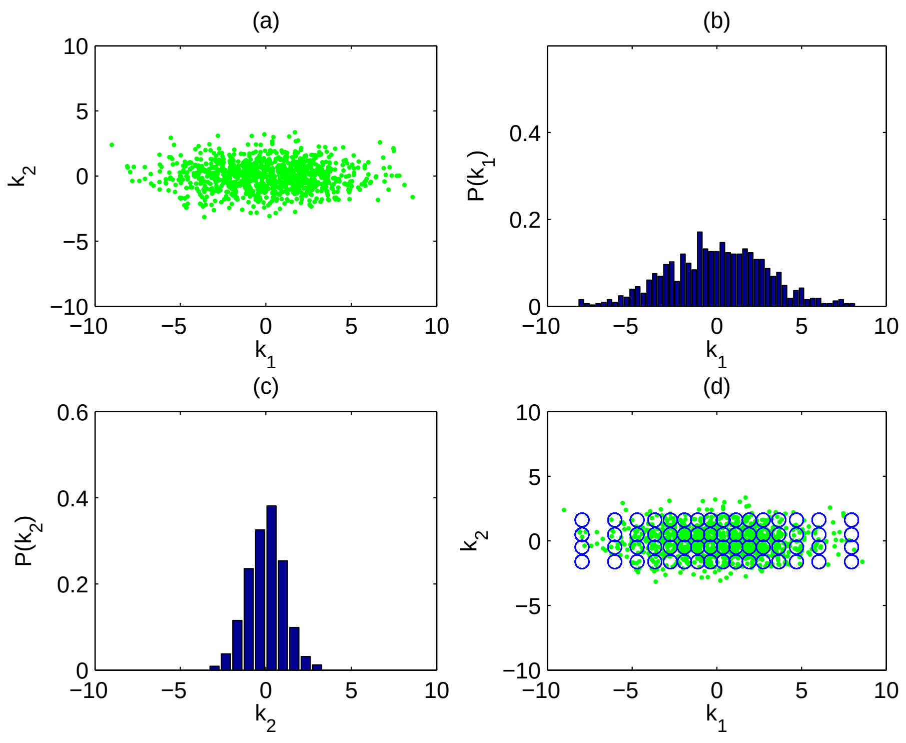


Figure 2.5: Scalar quantisation of KLT coefficients k_1 and k_2 using 3 bits/sample; (a) Scatter diagram of transformed (KLT) joint Gaussian random source; (b) Estimated marginal PDF of k_1; (c) Estimated marginal PDF of component k_2; (d) An overlay of the lattice of reproduction points (circles) on transformed (KLT) data.

by the 'tilt', cannot be exploited by the scalar quantisers from the marginal PDFs.

Figure 2.5(a) shows the transformed data points where it can be seen that the KLT has removed the 'tilt' or correlation from the data. The first component now has a larger variance than the second, as can be observed in Figures 2.5(b) and (c). Therefore, it can be said that more energy has been packed into the first eigenvector and the bit allocation algorithm should assign more bits to the scalar quantiser of this component. In this case, of the 6 bits available in the bit budget for each vector, 4 bits have been assigned to the first eigenvector while 2 bits to the second component. After the inverse KLT is applied and the vectors are transformed back to the original co-ordinate system of x_1 and x_2, it can be seen in Figure 2.6 that the resulting lattice of reproduction points is tilted in a similar way to the original data and there are fewer points falling in empty areas. In other words, the correlation between x_1 and x_2 has been been exploited by the block quantiser through the use of the KLT. The SNR of this system is 16.70 dB, thus the application of the KLT has resulted in a 2 dB improvement over just independent scalar quantising of

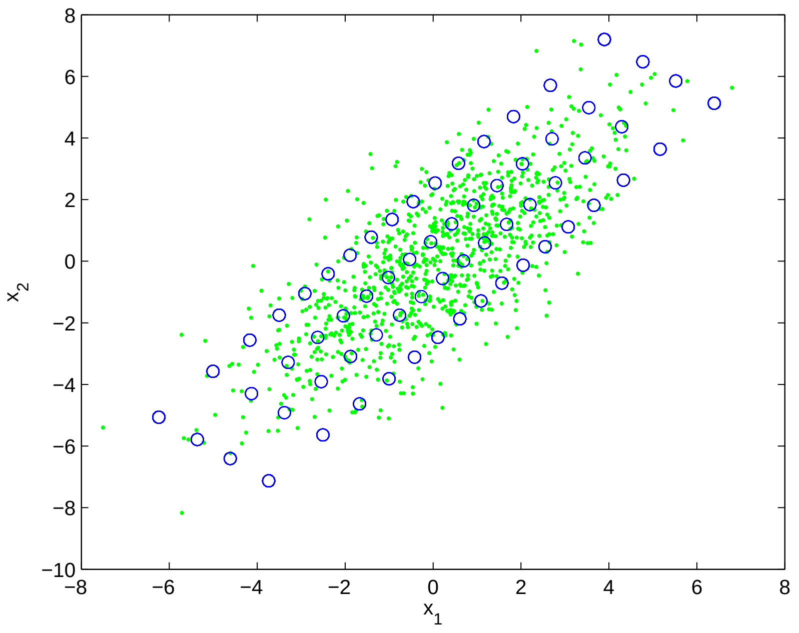


Figure 2.6: Block quantisation using 3 bits/sample with KLT applied (SNR=16.70 dB). Reproduction points (circles) and original data is also shown.

each vector component.

It is useful to quantify the *coding gain* achievable using the KLT in a block quantiser over no KLT performed and scalar quantising each vector sample independently. The samples are assumed to be generated by a weakly stationary Gaussian source, hence each sample has a common variance, $\sigma_{\boldsymbol{x}}^2$ [55]. From [72], the MSE distortion for each vector (of dimension n), of a typical block quantiser operating at $\bar{b}$ bits/sample, using high resolution approximations, is given by:

$$D_{bq} = nK(\bar{b})2^{-2\bar{b}}|\boldsymbol{\Sigma}_{\boldsymbol{x}}|^{\frac{1}{n}} \tag{2.34}$$

The MSE distortion of a vector (of dimension n), when each of its components are coded individually using a Gaussian Lloyd-Max scalar quantiser operating at $\bar{b}$ bits/sample and no transformation applied, is given by (assuming high resolution) [72]:

$$D_{sq} = nK(\bar{b})\sigma_{\boldsymbol{x}}^2 2^{-2\bar{b}} \tag{2.35}$$

Therefore, the coding gain is:

$$\begin{aligned}\frac{D_{sq}}{D_{bq}} &= \frac{nK(\bar{b})\sigma_{\boldsymbol{x}}^2 2^{-2\bar{b}}}{nK(\bar{b})2^{-2\bar{b}}|\boldsymbol{\Sigma}_{\boldsymbol{x}}|^{\frac{1}{n}}} \\ &= \frac{\sigma_{\boldsymbol{x}}^2}{(\prod_{i=1}^{n}\lambda_i)^{\frac{1}{n}}} \end{aligned} \tag{2.36}$$

Because the KLT is a unitary transform, the variance is preserved. That is, the sum of the variances of the transform coefficients (ie. eigenvalues) is the same as that of the original samples, $\sum_{i=1}^{n}\sigma_i^2 = \sum_{i=1}^{n}\lambda_i$ [198]. Therefore, since the individual variances of the samples, $\sigma_{\boldsymbol{x}}^2$, are common, the numerator of (2.36) can be rewritten in terms of the eigenvalues:

$$\frac{D_{sq}}{D_{bq}} = \frac{\frac{1}{n}\sum_{i=1}^{n}\lambda_i}{(\prod_{i=1}^{n}\lambda_i)^{\frac{1}{n}}} \tag{2.37}$$

which is the ratio of the arithmetic mean to the geometric mean of the eigenvalues. This ratio is always greater than or at least equal to unity and is higher when the vector components are more correlated [55]. That is, the more correlated the vectors, the greater is the benefit of using block quantisation over scalar quantisation.

2.2.2 Non-Uniform Scalar Quantisation

After the n dimension input vectors are transformed, producing n transform coefficients, each coefficient is independently scalar quantised. Since the distortion performance of any quantiser depends on the PDF of the input source [59], it is important to match the scalar quantisers to the PDF of the transform coefficients. Lloyd [101] introduced two methods for designing non-uniform scalar quantisers that matched any arbitrary source PDF. The second method was later re-derived by Max [111] and so the two methods are known as the Lloyd-Max algorithm.

As mentioned in the previous section, a special property of the KLT is that for jointly Gaussian sources, the transform coefficients are independent and Gaussian-distributed [55, 56]. Therefore, non-uniform scalar quantisers, specially designed to quantise Gaussian sources with minimum distortion (Lloyd-Max), are used in block quantisation. Even though not all sources are jointly Gaussian, the Gaussian assumption is generally a reasonable one since the transform coefficients are formed from linear combinations of vector components. When the dimension of the vectors is large, the PDF of the transform coefficients tends to approach a Gaussian distribution due to the Central Limit Theorem [28].

2.2.3 Bit Allocation

The role of bit allocation in block quantisation is to determine how to assign the budget of b_{tot} bits to the n scalar quantisers such that the overall distortion, D, is minimised. Since the distribution of the transform coefficient variances is non-uniform, it comes from the linear approximation property of the KLT that the number of bits assigned should be relative to the variance of that component, since components from subspaces spanned by the smallest eigenvalues contribute very little to the final reconstruction [34]. That is, more bits are assigned to components with larger variance (high energy) while less bits to components of small variance (low energy).

The minimisation of the distortion is of a constrained nature, since there are a limited number of bits to allocate. Huang and Schultheiss in [72] derived the optimal bit allocation formula for block quantising correlated and identically distributed Gaussian vectors. The distortion is assumed to be MSE. Their high resolution derivation, which involves Lagrangian minimisation, is presented below.

The distortion of a single Lloyd-Max non-uniform scalar quantiser for a Gaussian source, operating at b bits, is given as [72]:

$$D = K(b)\sigma^2 2^{-2b} \tag{2.38}$$

where σ^2 is the variance of the source. Given the variance and number of bits assigned to the ith component is λ_i and b_i, respectively, the average distortion incurred by a block quantiser on an n-dimension vector is:

$$D = \frac{1}{n}\sum_{i=1}^{n} K(b_i)\lambda_i 2^{-2b_i} \tag{2.39}$$

The goal is to minimise (2.39) under the fixed-rate constraint of:

$$b_{tot} = \sum_{i=1}^{n} b_i \tag{2.40}$$

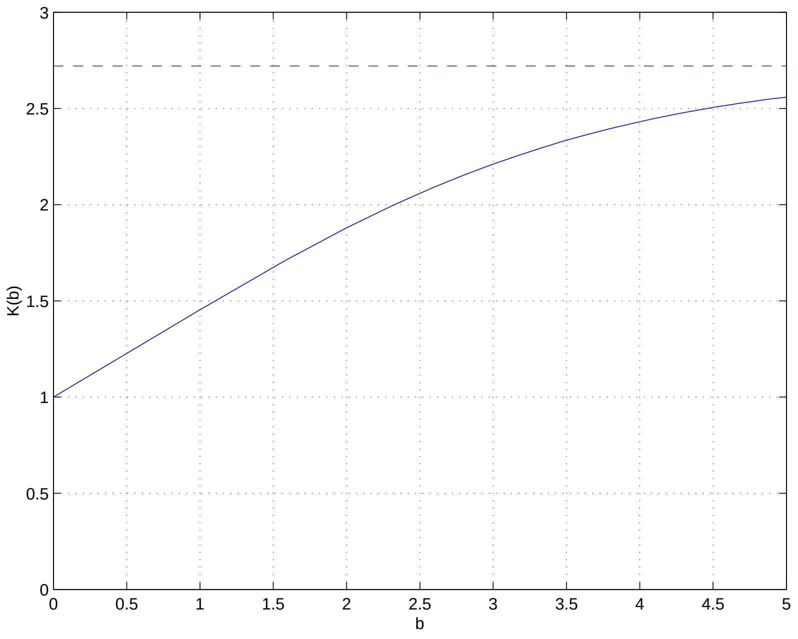


Figure 2.7: Plot of $K(b)$ as a function of the number of bits, b, for the Lloyd-Max scalar quantisation of a unity variance Gaussian source. The dashed line is the high resolution asymptote, $K(\infty) = \frac{\pi\sqrt{3}}{2}$ (after [72]).

Figure 2.7 shows the quantiser function, $K(b)$, as a function of the number of bits, b, for a Gaussian source, based on the error data from [111]. Since $K(b_i)$ asymptotically approaches a constant, $K = \frac{\pi\sqrt{3}}{2}$, at high bitrates, we use this to simplify the derivation. We also replace the 2^{-2b_i} with $e^{-2b_i \ln(2)}$ [72]. Using Lagrangian minimisation, where the multiplier is β, the expression to be minimised becomes:

$$\begin{aligned} \frac{\partial}{\partial b_j}\left[\frac{1}{n}\sum_{i=1}^{n} K\lambda_i e^{-2b_i \ln(2)} - \beta(b_{tot} - \sum_{i=1}^{n} b_i)\right] &= 0 \\ -\frac{1}{n}\lambda_j K 2\ln(2) e^{-2b_j \ln(2)} + \beta &= 0 \end{aligned}$$

Rearranging this equation:

$$\begin{aligned} \lambda_j e^{-2b_j \ln(2)} &= \frac{n\beta}{K 2\ln(2)} \\ &= C \end{aligned}$$

where C is a constant. Solving for b_j:

$$b_j = \frac{\ln \lambda_j - \ln C}{2\ln(2)} \tag{2.41}$$

Substituting this into the fixed bitrate constraint, (2.40):

$$\begin{aligned} b_{tot} &= \sum_{i=1}^{n} \left[\frac{\ln \lambda_i - \ln C}{2\ln(2)} \right] \\ \ln C &= \frac{1}{n} \ln \left(\prod_{i=1}^{n} \lambda_i \right) - 2\ln(2) \frac{b_{tot}}{n} \end{aligned} \tag{2.42}$$

Therefore, substituting (2.42) back into (2.41):

$$\begin{aligned} b_j &= \frac{1}{2\ln(2)} \left[\ln \lambda_j - \frac{1}{n} \ln \left(\prod_{i=1}^{n} \lambda_i \right) + 2\ln(2) \frac{b_{tot}}{n} \right] \\ b_j &= \frac{b_{tot}}{n} + \frac{1}{2} \log_2 \frac{\lambda_j}{\left(\prod_{i=1}^{n} \lambda_i \right)^{\frac{1}{n}}} \end{aligned} \tag{2.43}$$

The bit allocation formula of (2.43) allows us to distribute b_{tot} bits to the n scalar quantisers in such a way as to minimise the overall MSE distortion. It can be seen the number of bits allocated is proportional to the variance of the component, hence more bits will generally be given to the first few transform coefficients, where most of the energy has been packed into.

2.2.4 Problems with the KLT in Transform Coding

Though the KLT has optimal properties that are useful for transform coding, such as decorrelation and energy compaction, it has inherent disadvantages which have prevented it from being used in practical transform coders. Firstly, the transform is calculated from the source covariance matrix, and hence is source dependent [152, 34]. For the decoder to be able to perform the inverse transformation, it needs to know either the transformation matrix or the source covariance matrix, for the specific data to be decoded. These parameters need to be transmitted as side information, thus incurring transmission overhead. Alternatively, a static and global KLT matrix can be derived from training data and used in both encoding and decoding. However, this leads to the problem of source mismatch and suboptimal coding for data that is not part of and is statistically different to the training set [116].

Secondly, there are sources where the covariance matrix is (or is close to being) singular which leads to the KLT not being able to be uniquely defined [116]. It has been shown by Effros *et al.* [44] that for sources where the KLT is not unique, the 'worst' KLT can give distortion performance that is 1.5 dB lower than the 'best' KLT.

Lastly, the KLT can be a computationally complex operation. For a vector dimension of n, the KLT requires $2n^2 - n$ flops[11]. It is not amenable to fast and efficient computation, unlike the fast Fourier transform (FFT).

2.3 The Discrete Cosine Transform

The discrete cosine transform (DCT) transforms a vector, $\boldsymbol{x}$, to another vector, $\boldsymbol{y}$, by multiplying it with a transformation matrix, $\boldsymbol{D}$, which consists of cosine basis functions [143]:

$$\boldsymbol{y} = \boldsymbol{D}\boldsymbol{x} \tag{2.44}$$

where:

$$\boldsymbol{D}(i,j) = c_i \cos\left(\frac{2j+1}{2n} i\pi\right) \tag{2.45}$$

and

$$c_i = \begin{cases} \frac{1}{\sqrt{n}}, & \text{if } i = 0 \\ \sqrt{\frac{2}{n}}, & \text{otherwise} \end{cases} \tag{2.46}$$

For image applications where a two dimensional DCT is required and blocks or matrices, $\boldsymbol{X}$, of an image are operated on, the resulting coefficients are multiplied with the transpose of $\boldsymbol{D}$ in order to transform along the second dimension. In other words, the 2-D DCT is separable and can be expressed as (for a $n \times n$ block):

$$\boldsymbol{Y} = \boldsymbol{D}\boldsymbol{X}\boldsymbol{D}^T \tag{2.47}$$

where $\boldsymbol{X}$ is the $n \times n$ matrix, $\boldsymbol{Y}$ is the transformed matrix and $\boldsymbol{D}(i,j)$ is the (i,j)th element of the transformation matrix.

The DCT is popularly used as an approximation to the KLT. For first order Gauss-Markov processes[12] with exponential autocorrelation, $R(k) = \rho^{|k|}$, and a correlation coefficient, ρ, that is within the range of 0.5 and 1, the DCT has similar decorrelation and

[11]In our work, each addition, multiplication and comparison is considered one floating point operation (flop)

[12]A Gauss-Markov process is defined as $x_i = \rho x_{i-1} + u$, where ρ is the correlation coefficient and u is a Gaussian random variable of unit variance and zero mean [55].

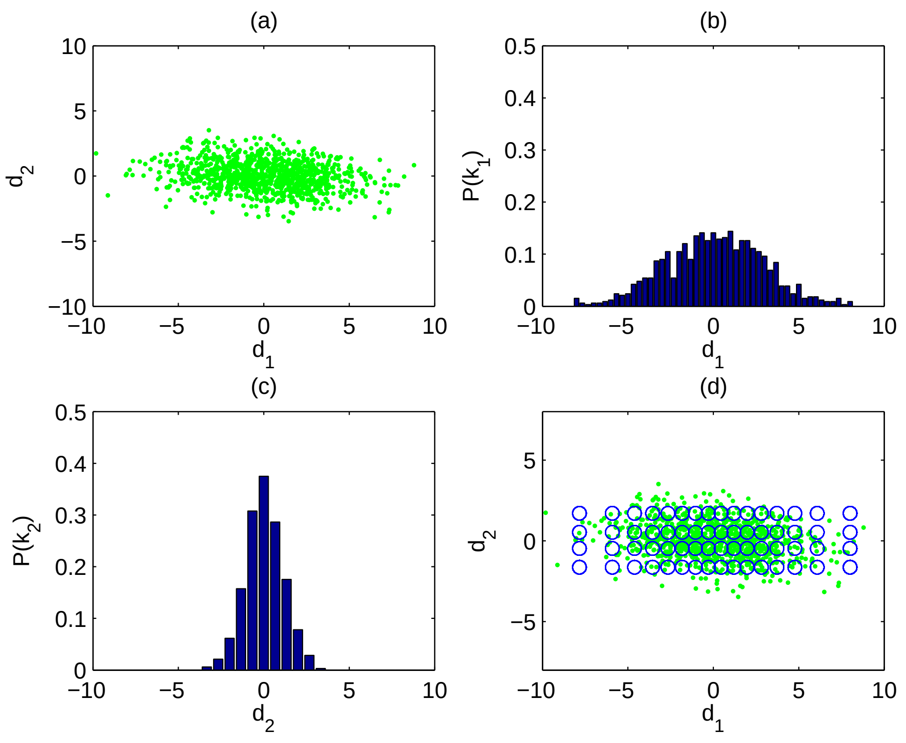


Figure 2.8: Scalar quantisation of DCT coefficients d_1 and d_2 using 3 bits/sample; (a) Scatter diagram of transformed (DCT) joint Gaussian random source; (b) Estimated marginal PDF of component d_1; (c) Estimated marginal PDF of component d_2; (d) An overlay of the lattice of reproduction points (circles) on transformed (DCT) data.

energy packing characteristics as the KLT[13] [78]. The n-point DCT is equivalent to the discrete time Fourier transform of the signal symmetrically extended with a period of $2n$. For real-life signals, which are normally lowpass, there are no artificial discontinuities in the periodically extended blocks, hence the smoothness leads to better energy packing than the FFT [34]. Secondly, the transformation matrix of the DCT contains no terms related to a source statistic such as covariance and hence the same matrix can be applied to all vectors generated from all sources [116]. And finally, because the DCT is related to the discrete Fourier transform [34], fast and efficient implementations are available. The JPEG image coding standard uses the DCT to transform image pixels into an energy-compact form [196].

In order to compare the performance of the DCT with the KLT in a block quantisation framework, the same correlated Gaussian data shown in Figure 2.2 was quantised using

[13]It is worth noting that the autocorrelation matrices of stationary processes have a Toeplitz structure and that as the dimensionality increases, the eigenvectors converge to complex exponentials [34]. Therefore, the discrete Fourier transform and its related transforms asymptotically converge to the KLT.

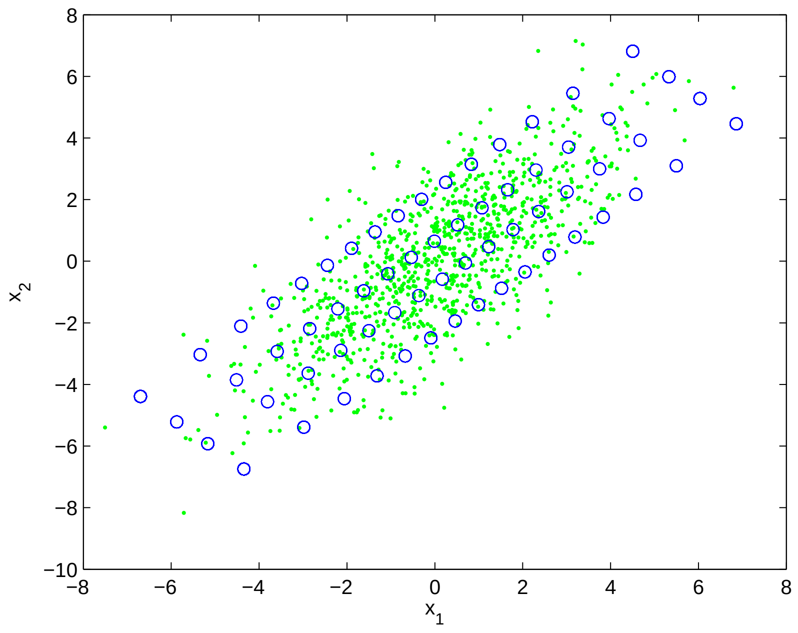


Figure 2.9: Block quantisation using 3 bits/sample with DCT applied (SNR=16.45 dB). Reproduction points (circles) and original data is also shown.

a 3 bits/sample block quantiser that uses the DCT. Figure 2.8(a) shows the transformed data. It can be seen that the decorrelation characteristics of the DCT are similar to that of the KLT, where most of the 'tilt' in the data points has been removed. Figures 2.8(b) and (c) shows the estimated marginal PDFs of the DCT components, where it can be observed that the variance of d_1 is higher than d_2, thus demonstrating the energy packing property of the KLT.

After scalar quantisation in the DCT domain, the resulting lattice of quantiser reproduction points in the original vector space is shown in Figure 2.9 with an SNR of 16.45 dB. As expected, the SNR of the DCT-based block quantiser is slightly lower than the KLT-based one. However, compared with the scalar quantisation of each component without any transform applied, the DCT offers lower distortion for the same bitrate. Therefore the DCT is a fast alternative to the KLT, though it is suboptimal in terms of coding performance. Since most image sources can typically be modelled as Gauss-Markov, the DCT is particularly popular in image coding applications.

2.4 Adaptive Transform Coding

2.4.1 Introduction

Even though the KLT, in combination with the Lloyd-Max scalar quantiser, has been proven to be optimal for correlated and identically distributed Gaussian sources [72, 154, 57], real-life sources rarely have PDFs that are Gaussian or even unimodal. Figure 2.10 shows the image ‘goldhill’ along with its PDF in two dimensions. Consecutive pixels from the image are taken as a vector. It can be observed that the source PDF (Figure 2.10(b)) is multimodal with a global mean of $(112, 112)$. If a block quantiser was applied to these vectors, it will align the Lloyd-Max quantiser reproduction lattice to match the global Gaussian, depicted in Figure 2.10(c). It can be seen that while the quantiser is focused in the region around $(112, 112)$ where most of the mass is centred, it will fail to reproduce the pixel values at the sharp peak located at around $(230, 230)$ accurately. This mismatch will invariably cause a decrease in distortion performance of the block quantiser.

The quantiser mismatch is two-fold. Firstly, the block quantiser, as mentioned above, assumes the PDF to be unimodal and Gaussian. As can be seen from Figure 2.10(b), this is certainly not the case and the optimality of the KLT for correlated Gaussian sources is not valid. And secondly, the block quantiser assumes the correlation is uniform throughout the vector space [13]. Figure 2.10(d) shows the first eigenvector of the global covariance matrix. However, at a finer localised level, the correlation may not necessarily be uniform and the local eigenvectors may point in different directions[14]. In other words, the assumption of global stationarity is not valid for most data sources, such as images [40]. Performance gains are expected to be achieved if the vector space is partitioned into disjoint regions where the statistics are relatively stationary. Designing a KLT for each region allows them to exploit the local correlation, allowing a better match with the block quantisers. This has led to the idea of adaptive transform coding.

2.4.2 Literature Review

Dony and Haykin [40] considered a scheme where the vector space is partitioned into classes, using an unsupervised, linear subspace classifying algorithm called *optimally in-*

[14]Note that because of the scale and relatively low dimensionality, finer level eigenvectors may be difficult to visualise in Figure 2.10(b).

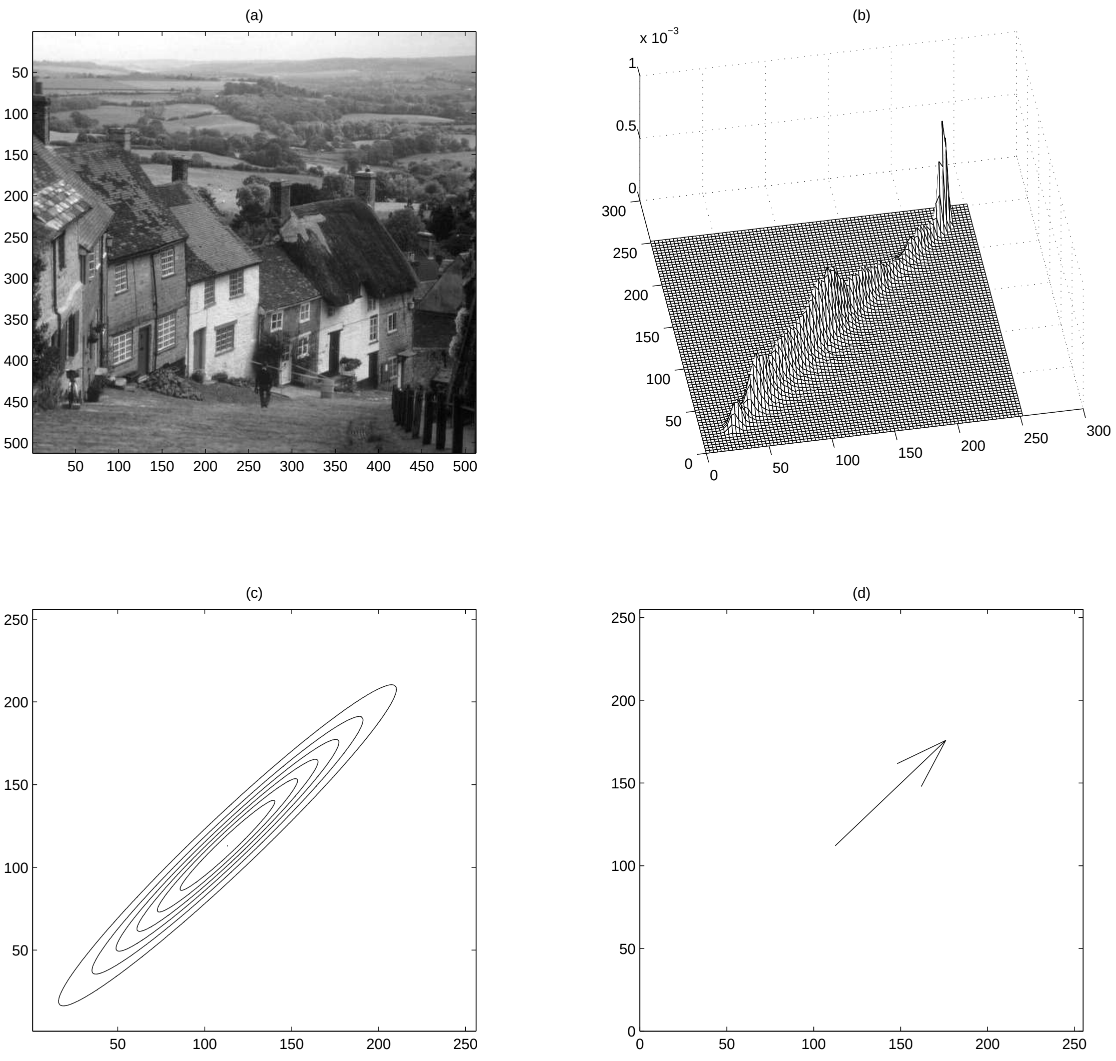


Figure 2.10: (a) The image ‘goldhill’; (b) PDF in two dimensions; (c) Contour plot of global ‘optimal’ Gaussian PDF; (d) Direction of first eigenvector

tegrated adaptive learning (OIAL). Their reasoning is that Euclidean distance-based classifiers, such as K-means [104] and LBG [100], are unsuitable for linear transform coding because they do not incorporate the vector norm. That is, two vectors which differ by a scalar multiple will be classified into different classes by K-means, when in fact, they would be appropriately represented by the same transformation bases [40]. The linear subspace classifier groups vectors into classes, based on the maximisation of the expected norm of the projected vectors, or alternatively speaking, minimising the MSE between the original vector and linearly approximated vector reconstructed from a reduced dimensional subspace [40]. As shown previously, the orthogonal bases which will minimise MSE in linear approximation are those of the KLT.

Archer and Leen [9] describe the adaptive transform coding problem as a local PCA and consider it an alternative to non-linear PCA, which partitions vector space using smooth and curved boundaries or manifolds, as opposed to the former, which uses straight, local hyperplanes. Their technique classifies vectors into subspaces of varying dimension, rather than a fixed global dimension, as was done by Dony *et al.* [40]. The 'dimension allocation' is a Lagrangian optimisation algorithm that minimises the expected distortion, subject to a fixed average dimension [9].

Later, Archer and Leen [11] noted that all previous adaptive transform techniques used clustering methods that minimised MSE due to dimensionality reduction rather than distortion induced by quantisation. Their modified algorithm consisted of a constrained version of the Linde-Buzo-Gray algorithm [100] (CLBG), which clusters the vectors based on minimum quantiser distortion [11]. The CLBG transform coder achieved an SNR of 2.5 to 3 dB higher than the global transform coder. A further increase of 1 dB was achieved by replacing the KLT with the *coding optimal transform* (COT), which is a special transform designed to minimise quantiser distortion rather than the MSE that is due to dimensionality reduction [12, 13]. This stems from the fact that the KLT is only optimal after quantisation, in the MSE sense, if the source is Gaussian. There is no guarantee that the vectors in each disjoint region obtained from the clustering process are a realisation from a Gaussian source, thus the optimality of the KLT reduces to that of dimensionality reduction only, which we have shown to be always valid, regardless of the source density function. In fact, it was shown that the COT reduces to the KLT when the data is Gaussian [12].

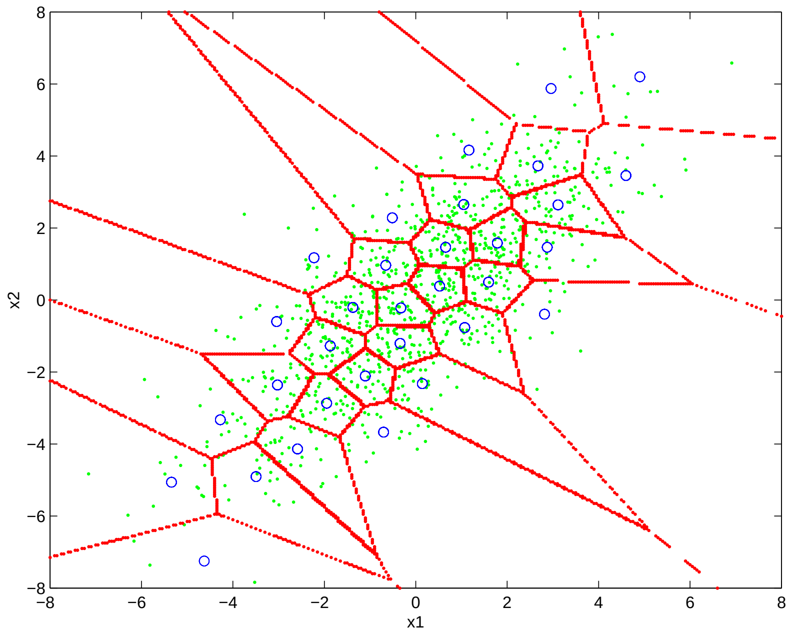


Figure 2.11: Voronoi regions from LBG algorithm. The dots represent the original data points, circles represent the centroids from the K-means algorithm, and the lines show the boundaries or decision surfaces which separate clusters

Effros *et al.* [43] described a scheme called *weighted universal transform coding* (WUTC), where the quantisation process is split into two stages. In the general weighted universal coding scheme, for each vector, the first stage selects a code from a family of codes, while the second stage quantises the vector using that code [43]. In the context of WUTC, each transform code consists of a transform and bit allocation [43]. The family of codes used in the first stage is designed using an entropy-constrained version of the generalised Lloyd algorithm (GLA). When applied to image coding, the WUTC was able to achieve an SNR of up to 3 dB over a single transform-based coder [43].

2.4.3 Using K-Means Clustering

In this section, we consider the case of using the basic K-means clustering algorithm and designing block quantisers for each cluster. This process is suboptimal because the vector space partitioning and quantiser design are performed independently and not optimised jointly to minimise distortion. However, the advantage of this type of scheme is *bitrate scalability*. That is, the bitrate of the quantisation scheme can be dynamically changed, without requiring any re-training of the models or quantiser. This is because the scalar quantiser levels used are predefined and only the bit allocations are adaptively changed. The adaptive transform coding schemes mentioned in the previous sub-section, which include a constraint for minimising quantiser distortion [43, 11, 10], partition the vector space adaptively and produce transform and quantisation parameters that are dependent on the operating bitrate. Also, this K-means hard clustering scheme will serve as a useful comparison with the soft GMM-based clustering scheme, which will be discussed in the next section. The only difference between the two is the *a priori* assumption of the PDFs of each cluster being Gaussian and the use of the Expectation-Maximisation (EM) algorithm, in the GMM-based block quantisation scheme.

The K-means [104] or LBG [100] algorithm classifies vectors into classes or clusters, based on the minimum Euclidean distance. If the distortion measure is MSE, these centroids form the codebook of a vector quantiser, which is the best quantiser in the MSE sense, for a given dimension. The clusters are bounded by the intersection of various hyperplanes and these form regions called Voronoi regions, as shown in Figure 2.11. Then a block quantiser is designed for each cluster. Bits may be distributed to each block quantiser uniformly, or non-uniformly based on a constrained minimisation formula (similar to the one contained in Section 2.5.3) that minimises the MSE.

This simple scheme is expected to perform better than the traditional block quantiser because it employs multiple transforms that adapt to the local statistics of different regions. Therefore, we refer to this scheme as the *K-means-based multiple transform block quantiser*. The performance of this scheme will depend on the PDF of the vectors within each cluster. Though the KLT decorrelates vectors, it will not produce independent coefficients unless the vectors are Gaussian distributed. Decorrelation and even statistical independence, are not sufficient for optimal transform coding, as was shown in [44], unless the source is Gaussian. Because the K-means algorithm partitions vectors based on the unweighted Euclidean distance, with no regard for the underlying PDF, then there is no guarantee that the vectors inside each cluster will be Gaussian distributed. Hence the block quantisers will perform suboptimally. However, this optimality condition of Gaussian PDFs for block quantisers can be satisfied by imposing an *a priori* distribution on each cluster, which is the main novelty of the GMM-based block quantiser, to be discussed in the next section.

The K-means-based multiple transform block quantiser will be evaluated in image coding experiments, in Chapter 4.

2.5 Gaussian Mixture Model-Based Block Quantisation

2.5.1 Source Modelling for Quantiser Design

The probability density functions (PDF) of real life sources are rarely Gaussian. Since the scalar quantisers in traditional block quantisers are designed to quantise Gaussian sources efficiently, any PDF mismatch will invariably cause a degradation in performance. Rather than assuming the source PDF to be a standard function such as Gaussian, Laplacian, etc., one can design a quantiser that matches the source PDF as close as possible. The K-means algorithm, otherwise known in the literature as the generalised Lloyd algorithm (GLA) and Linde-Buzo-Gray (LBG) algorithm[15], allows one to design vector quantisers which match the PDF of training data and quantise it with minimum distortion, via the Lloyd conditions [55, 100].

There have been numerous studies in the coding literature on source PDF modelling for quantiser design. These can be broadly classified as either *non-parametric* or *parametric modelling*. Ortega and Vetterli [118] estimated the source model in a non-parametric fashion using piecewise linear approximation. No side information is required as they used a backward adaptation scheme which allows the decoder to derive the source model based on the previous quantised values. Similarly, multidimensional histograms were used by Gardner and Rao [52] to model the PDFs of line spectral frequencies (LSF) in order to evaluate the high-rate bounds of split vector quantisers.

In relation to parametric modelling, Su and Mercereau [181] applied Gaussian mixture models (GMM) in the estimation of the PDF of DC (zero frequency) transform coefficients in DCT-based transform coding of images while the remaining AC (higher frequency) coefficients were modelled using generalised Gaussians (GG). It has been known, with regards to the distribution of DCT coefficients, that the DC coefficient has a Gaussian-like PDF[16] while the AC coefficients have a Laplacian-like PDF [143]. Archer and Leen [10, 13] used GMMs to form a probabilistic latent variable model from which they derived optimal transform coding design algorithms.

[15]Notable differences between the K-means algorithm and GLA/LBG algorithm are the initialisation used as well as the former's sequential nature [100]. Due to its sequential nature, the K-means algorithm is therefore more suited for clustering applications than optimal quantiser design. Since the refinement steps, which are based on the Lloyd conditions for optimality, are essentially the same, we will use the terms 'K-means algorithm', 'GLA', and 'LBG algorithm', interchangeably in this paper.

[16]Originally, a Rayleigh and Gaussian distribution were thought to be appropriate models for DC and AC coefficients, respectively [143].

On the speech side, Hedelin and Skoglund [66] used GMMs for designing and evaluating high-rate (approximately 20 bits) vector quantisers of LSFs. The codebook size of these vectors quantisers is very large and in order to prevent 'overfitting', which causes suboptimal performance on disjoint data, there needs to be a very large set of training vectors [66]. The idea is to find a GMM that can accurately model the PDF of line spectral frequencies. This GMM can then be used to artificially generate a large training set of vectors which can then be used to design vector quantisers. They made two modifications to the GMM estimation procedure which are optimised for vector quantiser design. One modification constrained the shape the multivariate Gaussians to match the boundaries of certain PDFs more accurately, otherwise known as *GMMs with bounded support* [66]. The next modification involved applying gradient-based optimisation to the Expectation-Maximisation algorithm to produce the high rate optimised (HRO) estimation algorithm [66]. The aim is minimise quantiser distortion rather than log-likelihood [66].

Subramaniam and Rao [183] used GMMs in combination with transform coding to quantise line spectral frequencies. The GMM-based block quantiser[17] of [183] models any arbitrary source of data vectors as a mixture of individual and overlapping Gaussian sources (or, clusters). GMMs can be used to approximate arbitrary densities and have been used in many applications, such as speech recognition [204] and speaker identification [146].

One useful view of the GMM, where the data vectors are assumed to be incomplete and that there is hidden and unobserved data, is that each observation vector is assumed to be generated by *one of the Gaussian sources*, depending on the weight or probability of that cluster [22]. Using the GMM framework for source modelling allows us to decompose any complex and arbitrary PDF into a series of Gaussian basis PDFs and for each Gaussian basis source, we can then design efficient quantisers. This is analogous to the transform coding idea of breaking the signal down into basis components and designing individual quantisers for each one.

Block quantisers [72] are known to be efficient for correlated Gaussian sources [56], hence one is designed for each cluster. Because an observation vector is assumed to be

[17]The original scheme of Subramaniam and Rao [183] was termed as *parametric vector quantisation*, where the features of scalability, compact representation, and bitrate independent complexity were highlighted as advantages over vector quantisers. We have taken a different interpretation of this scheme and view it as an improved version of the block quantiser, that is matched to arbitrary PDFs rather than the standard Gaussian.

generated by one Gaussian source only, then the cluster block quantiser for that source will code it with minimal distortion, in the absence of other overlapping cluster block quantisers. However, the overlapping of Gaussian clusters coupled with the relatively low complexity of block quantisers allows us to use a soft-clustering approach to choose the best cluster block quantiser based on minimum distortion, rather than maximum likelihood [183].

This quantisation scheme differs from the adaptive transform coders [40, 43, 13] found in the image coding literature, where the vector space is partitioned into *disjoint* regions and transform coders are designed for each region. The aim of these adaptive transform coders is to partition the vector space into regions of similar statistics, in conjunction with the design of transform coders, in order to minimise quantiser distortion [13]. Whereas for the GMM-based block quantiser, the aim is to approximate the source PDF accurately using a GMM, decomposing it into multiple overlapping Gaussian clusters [183]. Block quantisers, which can code correlated Gaussian sources efficiently, are then designed for each cluster.

The main advantages of the GMM-based block quantiser, compared with vector quantisers, are [183]:

- Compact representation of the source PDF, stored at the encoder and decoder;
- bitrate scalability with 'on-the-fly' bit allocation; and
- low search complexity and memory requirements which are independent of the bitrate of the system.

Since the GMM parameters for the estimated PDF and the KLT transformation matrices are properties of the source and thus independent of the bitrate, this procedure only needs to be done once (training). The GMM parameters and KLT transformation matrices are stored at the encoder and decoder, hence there is no transmission overhead associated [183]. With regards to bitrate scalability, closed-form expressions exist for bit allocation among GMM clusters as well as within each cluster, which allows for fast computation by both the encoder and decoder, should there be a change in the bitrate. The non-uniform scalar quantisers are also bitrate scalable and can be implemented as a rounding function accompanied by appropriate companding and expanding functions. This efficient scalar quantiser implementation also limits the number of searches required in the GMM-based block quantiser to be dependent only on the number of clusters in the GMM, and not the bitrate [183].

The GMM-based block quantiser can be broken down into two stages: the training stage (PDF estimation) and encoding stage (bit allocation and minimum distortion block quantisation). These will be described in the following sub-sections. In the training stage, a GMM is calculated, using the training vectors, to produce a mean, weight, and covariance matrix for each Gaussian cluster (or, mixture component). An eigenvalue decomposition is performed on the covariance matrix of each cluster, to produce a KLT matrix, $\boldsymbol{P}$, and a set of eigenvalues, $\{\lambda\}$. In the encoding stage, the cluster weights and eigenvalues are used to determine the bit allocation, given the target bitrate. The rest of the parameters, such as the cluster mean and KLT matrix are then used in the minimum distortion block quantisation of the vectors.

2.5.2 PDF Estimation Using Gaussian Mixture Models and the EM Algorithm

The probability density function, P, as a mixture of multivariate Gaussians, $\mathcal{N}(\boldsymbol{x};\boldsymbol{\mu},\boldsymbol{\Sigma})$, is given by:

$$P(\boldsymbol{x}|\mathcal{M}) = \sum_{i=1}^{m} c_i \mathcal{N}(\boldsymbol{x};\boldsymbol{\mu}_i,\boldsymbol{\Sigma}_i) \tag{2.48}$$

$$\mathcal{M} = [m, c_i, \ldots, c_m, \boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_m, \boldsymbol{\Sigma}_1, \ldots, \boldsymbol{\Sigma}_m] \tag{2.49}$$

$$\text{where } \mathcal{N}(\boldsymbol{x};\boldsymbol{\mu},\boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{\frac{n}{2}}|\boldsymbol{\Sigma}|^{\frac{1}{2}}} e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})} \tag{2.50}$$

$$\text{and } \sum_{i=1}^{m} c_i = 1 \tag{2.51}$$

where $\boldsymbol{x}$ is a source vector, m is the number of mixture components (or, clusters), and m is the dimensionality of the vector space. $c_i, \boldsymbol{\mu}_i$ and $\boldsymbol{\Sigma}_i$ are the probability, mean, and covariance matrix of the ith mixture, respectively. Note the words 'mixture component' and 'cluster' will be used interchangeably in this paper.

The parametric model parameters, $\mathcal{M}$, are initialised by applying the LBG algorithm [100] on the training vectors representing the source distribution and m clusters are produced, each represented by a mean or centroid, $\boldsymbol{\mu}$, a covariance matrix, $\boldsymbol{\Sigma}$, and a cluster weight, c. These form the initial parameters ($k = 0$) for the GMM estimation procedure. Using the Expectation-Maximisation (EM) algorithm [37], the maximum-likelihood estimate of the parametric model is computed iteratively and a final set of means, covariance matrices, and weights are produced. The EM algorithm is described below.

Expectation Step (E-step)

In the $(k+1)$th E-step, the *a posteriori* probability of the ith cluster, given the model, $\mathcal{M}$, and vector, $\boldsymbol{x}_j$, is calculated for all s vectors and m clusters:

$$p(i|\boldsymbol{x}_j,\mathcal{M}^{(k)}) = \frac{c_i^{(k)}\mathcal{N}(\boldsymbol{x}_j;\boldsymbol{\mu}_i^{(k)},\boldsymbol{\Sigma}_i^{(k)})}{\sum_{m=1}^{M} c_m^{(k)}\mathcal{N}(\boldsymbol{x}_j;\boldsymbol{\mu}_m^{(k)},\boldsymbol{\Sigma}_m^{(k)})} \tag{2.52}$$

where $i = 1, 2, \ldots, m$ and $j = 1, 2, \ldots, s$.

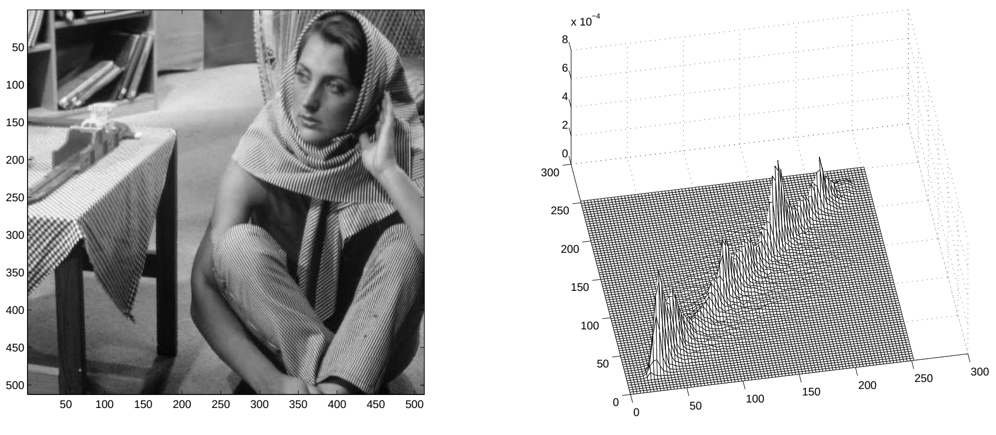


Figure 2.12: The image 'barbara' and its PDF in two dimensions

Maximisation Step (M-step)

With the *a posteriori* probabilities from the E-step, the $(k+1)$th M-step re-estimates the cluster weights, means, and covariance matrices.

$$c_i^{(k+1)} = \frac{1}{s}\sum_{j=1}^{s} p(i|\boldsymbol{x}_j, \mathcal{M}^{(k)}) \tag{2.53}$$

$$\boldsymbol{\mu}_i^{(k+1)} = \frac{\sum_{j=1}^{s} p(i|\boldsymbol{x}_j, \mathcal{M}^{(k)})\boldsymbol{x}_j}{\sum_{j=1}^{s} p(i|\boldsymbol{x}_j, \mathcal{M}^{(k)})} \tag{2.54}$$

$$\boldsymbol{\Sigma}_i^{(k+1)} = \frac{\sum_{j=1}^{s} p(i|\boldsymbol{x}_j, \mathcal{M}^{(k)})(\boldsymbol{x}_j - \boldsymbol{\mu}_i^{(k+1)})(\boldsymbol{x}_j - \boldsymbol{\mu}_i^{(k+1)})^T}{\sum_{j=1}^{s} p(i|\boldsymbol{x}_j, \mathcal{M}^{(k)})} \tag{2.55}$$

Each iteration of the EM algorithm is guaranteed to increase or leave unchanged the likelihood of the GMM, which converges to a local optimum.

$$\mathcal{L}(\mathcal{M}^{(k+1)}|\boldsymbol{x}) \geq \mathcal{L}(\mathcal{M}^{(k)}|\boldsymbol{x}) \tag{2.56}$$

where the log-likelihood of the GMM, given the vectors, $\{\boldsymbol{x}_i\}_{i=1}^{s}$, is expressed as:

$$\mathcal{L}(\mathcal{M}|\boldsymbol{x}) = \frac{1}{s}\sum_{j=1}^{s} \ln P(\boldsymbol{x}_j|\mathcal{M}) \tag{2.57}$$

$$= \frac{1}{s}\sum_{j=1}^{s} \ln\left[\sum_{l=1}^{m} c_l \mathcal{N}(\boldsymbol{x}_j; \boldsymbol{\mu}_l, \boldsymbol{\Sigma}_l)\right] \tag{2.58}$$

In order to appreciate the role and effectiveness of using a GMM to model the PDF, two dimensional vectors (consecutive horizontal pixels) were extracted from the image

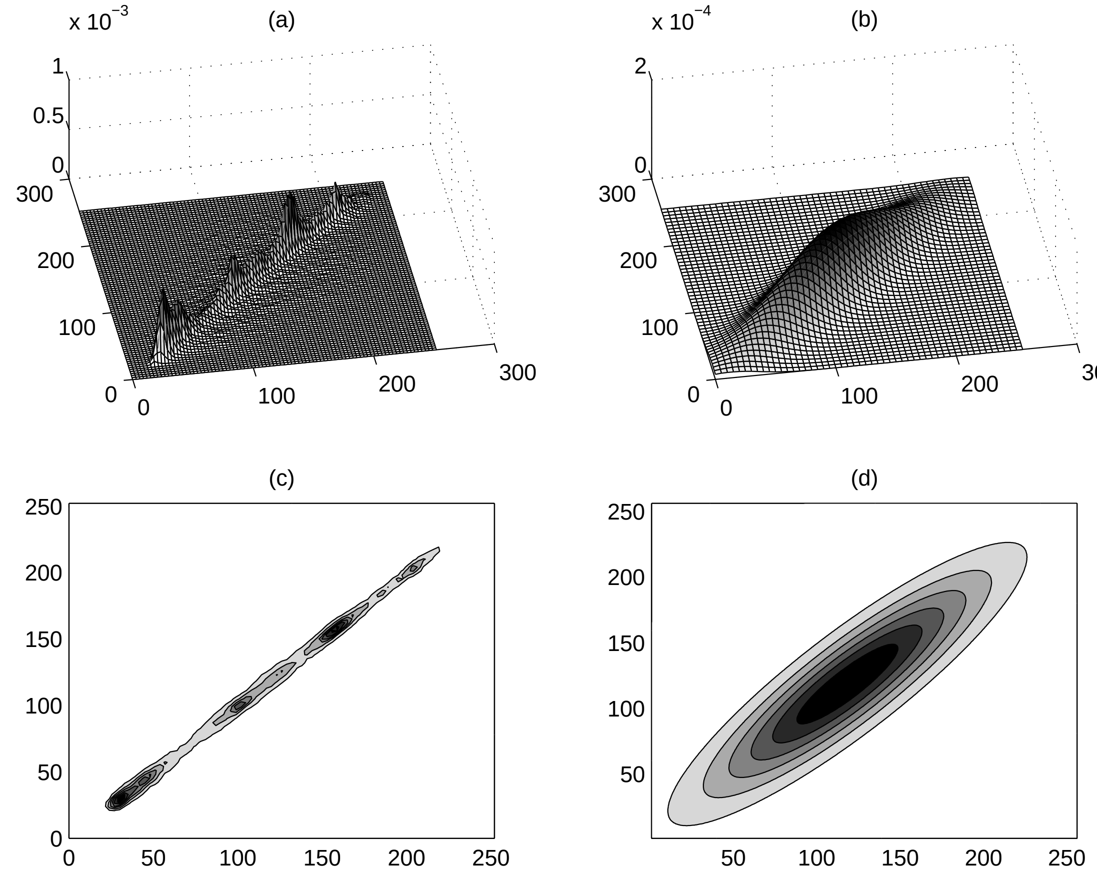


Figure 2.13: PDF and GMM of the image 'barbara': (a) Original PDF; (b) Single cluster GMM (Gaussian); (c) Contour plot of original PDF; (d) Contour plot of the single cluster GMM (Gaussian)

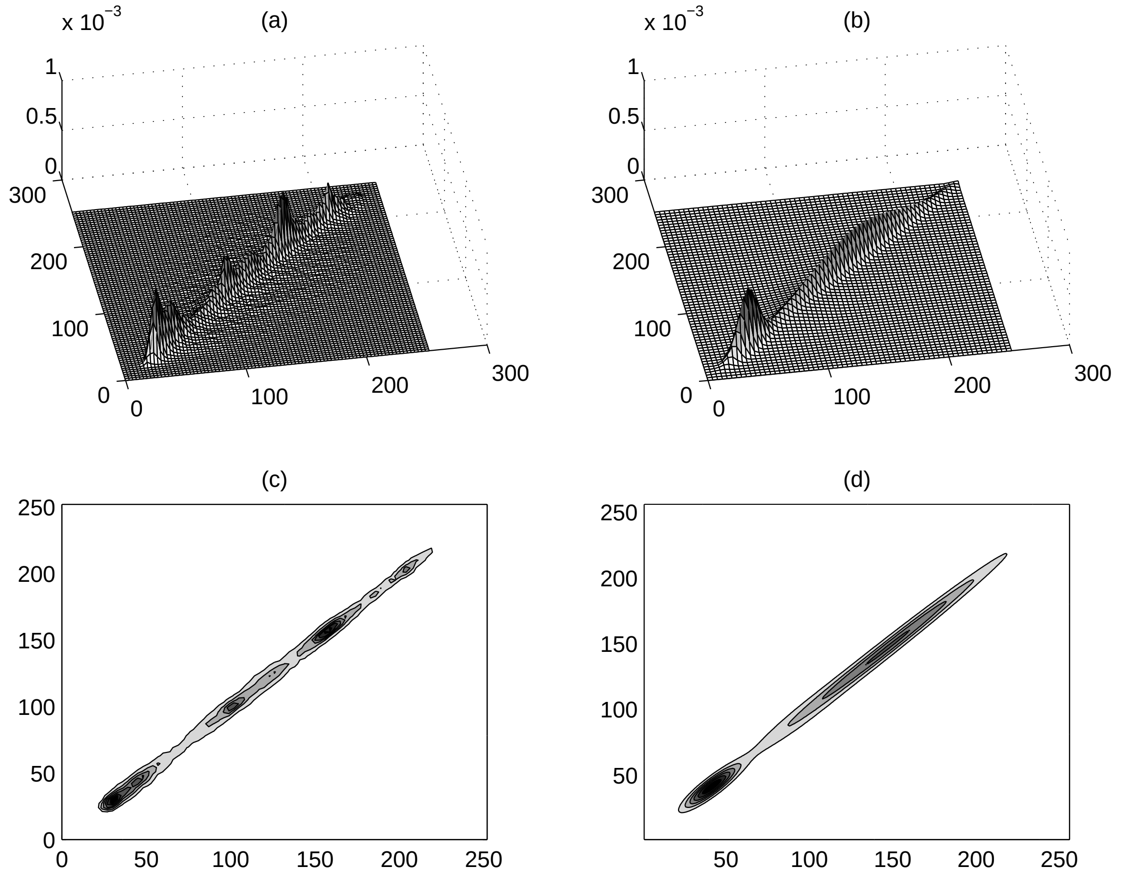


Figure 2.14: PDF and GMM of the image 'barbara': (a) Original PDF; (b) 4 cluster GMM; (c) Contour plot of original PDF; (d) Contour plot of the 4 cluster GMM

'barbara'. The image and the PDF of the vectors are shown in Figure 2.12. Firstly, it can be seen from the PDF that consecutive pixels are highly correlated since the image has a lot of smooth and uniform regions. Also, it can observed that the PDF is multimodal which implies that a single unimodal Gaussian is insufficient and this is confirmed in Figures 2.13(b) and (d), though the GMM does capture the global correlation (tilt). In single transform block quantisation, a Lloyd-Max scalar quantiser lattice is designed to be aligned with this unimodal Gaussian and consequently there will be a loss of performance due to mismatch with the complex multimodal PDF of the original data. Also, local correlation in the source cannot be exploited since the global KLT aligns the scalar quantiser lattice to the predominant tilt of the PDF, though it can seen that at least for 2D image vectors, local correlations appear somewhat uniform and point in the same 'direction' as the global correlation. This interesting observation will be revisited again in a later section which will detail a DCT-based quantisation scheme.

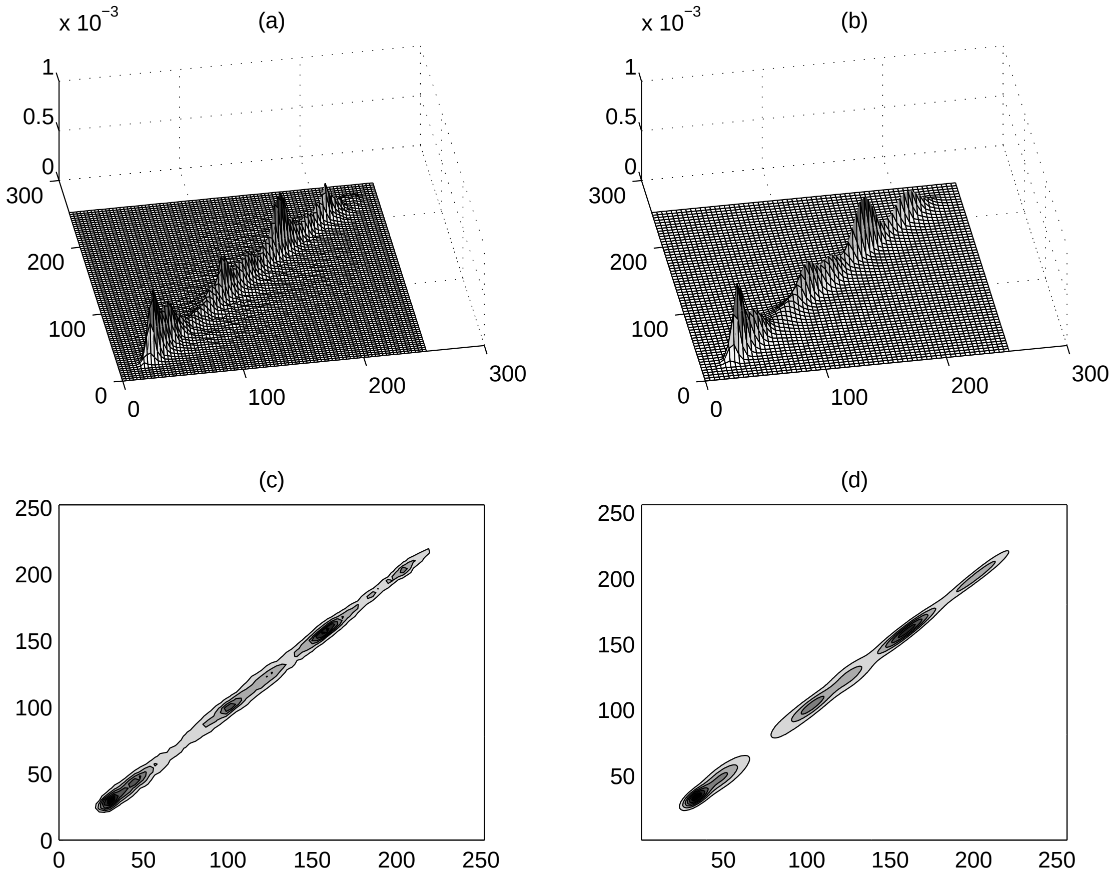


Figure 2.15: PDF and GMM of the image 'barbara': (a) Original PDF; (b) 16 cluster GMM; (c) Contour plot of original PDF; (d) Contour plot of the 16 cluster GMM

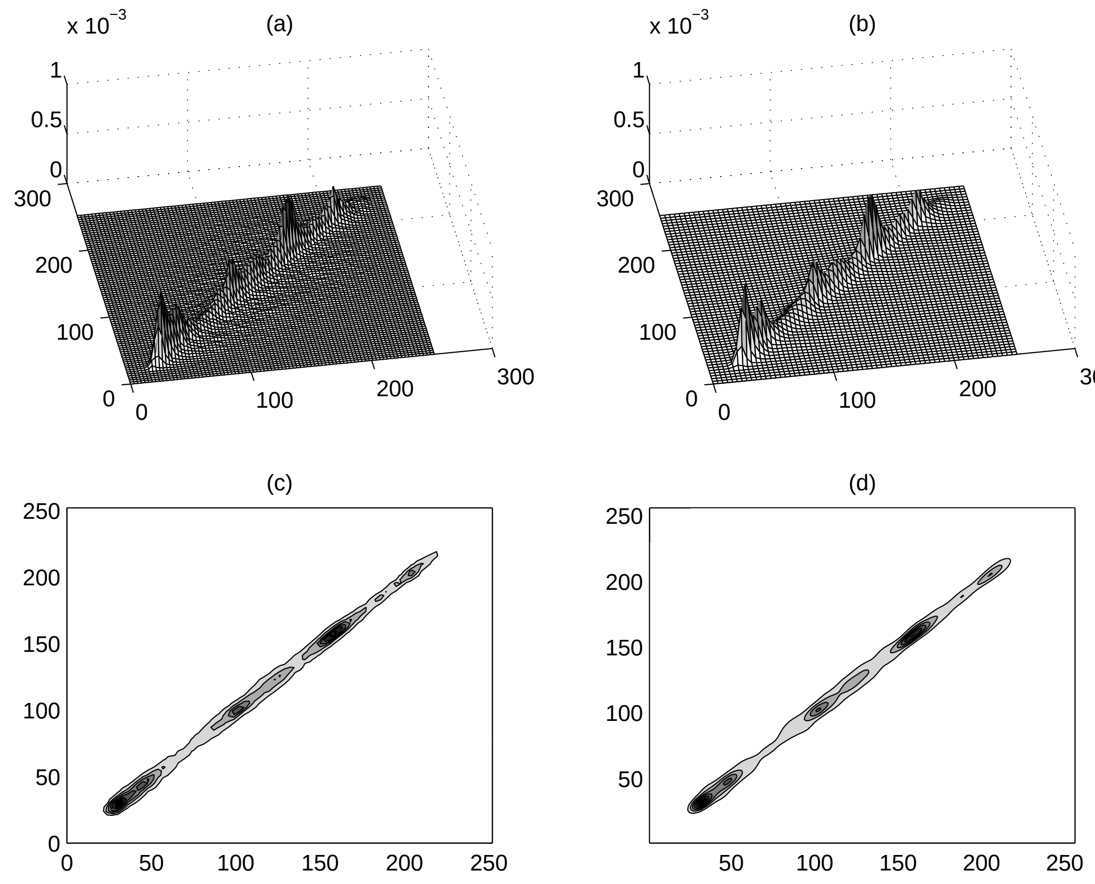


Figure 2.16: PDF and GMM of the image 'barbara': (a) Original PDF; (b) 32 cluster GMM; (c) Contour plot of original PDF; (d) Contour plot of the 32 cluster GMM

Figure 2.14 shows a 4 cluster GMM. It can be seen that the two peaks in the lower left corner of the original have been captured by a Gaussian mixture component. In the GMM-based block quantiser, a block quantiser is designed for each mixture component of the GMM, hence one will be designed for the mixture component in the lower left, which will exploit the local correlation and match the density more efficiently than the single Gaussian case of Figure 2.13.

Increasing the number of clusters to 16 and 32, as shown in Figures 2.15 and 2.16, provides much better modelling of the original multimodal PDF by the GMM. Most of the peaks in the original PDF have been captured by the GMM and consequently, block quantisers will be positioned in these regions to match the local statistics. It is worth noting that these Gaussian mixture components will overlap each other and depending on the number of bits assigned, the resulting scalar quantiser lattices of each block quantiser will also overlap. This suggests that the GMM provides a soft partitioning of the vector space so a soft decision will need to made when choosing which cluster block quantiser to use for quantising a given vector.

In the incomplete data interpretation of the GMM, each vector is assumed to be generated by one Gaussian source at a time and the unobservable random variable is the cluster number, whose distribution is determined by the cluster probabilities, c. The GMM can be used to generate random vectors which possess the same PDF characteristics via the following procedure:

1. Form cumulative probabilities from the c's, which partition the range of $[0 \ldots 1]$ to each mixture component;
2. use a random variable of uniform distribution between 0 and 1 to select the mixture component;
3. generate Gaussian random vector with the mean and covariance of the selected mixture component.

Using this process, one can generate a large number artificial training vectors for designing quantisers, which is what Hedelin and Skoglund [66] have done in their GMM-based vector quantiser for the coding of line spectral frequencies.

2.5.3 Bit Allocation

There are two types of bit allocation that are required: *intercluster bit allocation* and *intracluster bit allocation*. Bitrate scalability is a feature of the GMM-based block quantiser [183]. The bit allocation, depending on the chosen bitrate, is done 'on-the-fly' by both the encoder and decoder, based on the common GMM and KLT parameters. Therefore, the bit allocation is fixed and synchronised between the encoder and decoder. For the purpose of 'on-the-fly' bitrate scalability, the bit allocation procedure needs to be computationally efficient. For this reason, classical bit allocation algorithms, based on the high-resolution performance of Gaussian scalar quantisers [72], are preferred over the more optimal allocation algorithm of Riskin [147].

Intercluster Bit Allocation

With intercluster bit allocation, the range of quantiser levels is partitioned to each of the m cluster block quantisers. The derivation given in [182, 184] is presented below. For a fixed-rate quantiser, the total number of quantiser levels is fixed:

$$2^{b_{tot}} = \sum_{i=1}^{m} 2^{b_i} \tag{2.59}$$

where b_{tot} is the total number of bits in the bit budget, b_i is the number of bits assigned to cluster i, and m is the number of clusters. Since the cluster weights can be thought of as probabilities of occurrence of each cluster [183], the average distortion is approximated by[18]:

$$D_{tot} = \sum_{i=1}^{m} c_i D_i(b_i) \tag{2.60}$$

Using the high resolution approximation for distortion of a single Lloyd-Max scalar quantiser, the total distortion of a block quantiser is [183]:

$$\begin{aligned} D_i(b_i) &= K n \lambda_i 2^{-2\frac{b_i}{n}} && (2.61) \\ \lambda_i &= \left[\prod_{j=1}^{n} \lambda_{i,j}\right]^{\frac{1}{n}} && (2.62) \\ &\text{for } i = 1, 2, \ldots, m \end{aligned}$$

[18]Note that this is based on the assumption is that there is negligible overlap between clusters.

where n is the dimension of the vectors, m is the number of clusters, $\lambda_{i,j}$ is the jth eigenvalue of cluster i, and K is a constant which is asymptotically equal to $\frac{\pi\sqrt{3}}{2}$ for Gaussian sources [199, 59].

Using Lagrangian minimisation, where the multiplier is β:

$$\begin{aligned}\frac{\partial}{\partial b_j}\left[\sum_{i=1}^{m} c_i K n \lambda_i 2^{-2\frac{b_i}{n}} + \beta(\sum_{i=1}^{m} 2^{b_i} - 2^{b_{tot}})\right] &= 0 \\ \beta 2^{b_j} - 2Kc_j\lambda_j 2^{-2\frac{b_j}{n}} &= 0\end{aligned}$$

Rearranging this equation to separate out the constants:

$$\begin{aligned}\frac{2^{b_j(\frac{n+2}{n})}}{c_j\lambda_j} &= \frac{2K}{\beta} \\ &= G \\ 2^{b_j} &= (c_j\lambda_j)^{\frac{n}{n+2}} G^{\frac{n}{n+2}} \qquad (2.63)\end{aligned}$$

Substituting into the constraint (2.59):

$$\begin{aligned}2^{b_{tot}} &= \sum_{i=1}^{m} 2^{\log_2(c_i\lambda_i G)^{\frac{n}{n+2}}} \\ &= G^{\frac{n}{n+2}} \sum_{i=1}^{m} (c_i\lambda_i)^{\frac{n}{n+2}} \\ G^{\frac{n}{n+2}} &= \frac{2^{b_{tot}}}{\sum_{i=1}^{m}(c_i\lambda_i)^{\frac{n}{n+2}}} \qquad (2.64)\end{aligned}$$

Substituting this back into (2.63):

$$2^{b_j} = 2^{b_{tot}} \frac{(c_j\lambda_j)^{\frac{n}{n+2}}}{\sum_{i=1}^{m}(c_i\lambda_i)^{\frac{n}{n+2}}} \qquad (2.65)$$

The number of quantiser levels given to the cluster block quantiser is proportional to the geometric mean of the eigenvalues and the probability of that cluster.

Intracluster Bit Allocation

After the bit budget is partitioned to each cluster, the bits need to be allocated to each of the n components within each cluster block quantiser using existing bit allocation techniques for transform coding [55, 183]:

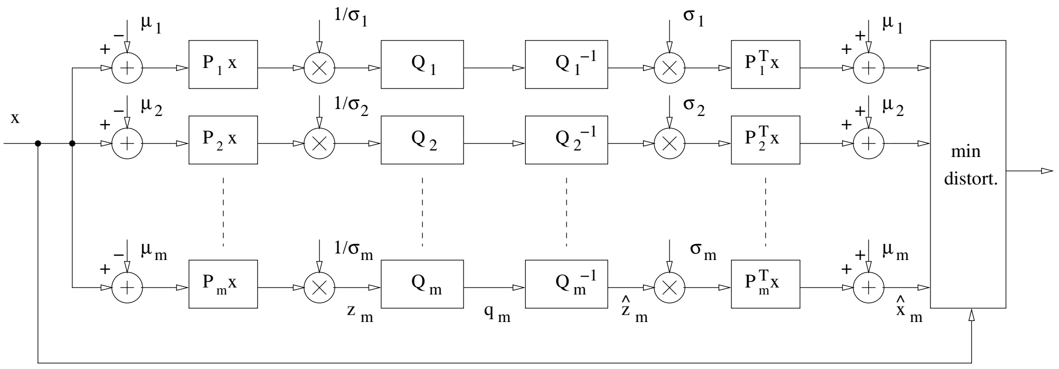


Figure 2.17: Minimum distortion block quantisation (Q – cluster block quantiser)

$$b_{i,j} = \frac{b_i}{n} + \frac{1}{2}\log_2 \frac{\lambda_{i,j}}{\left(\prod_{j=1}^{n} \lambda_{i,j}\right)^{\frac{1}{n}}} \tag{2.66}$$

$$\text{for } i = 1, 2, \ldots, m$$

where $\lambda_{i,j}$ and $b_{i,j}$ are the jth eigenvalue and number of bits allocated to component j of cluster i, respectively.

2.5.4 Minimum Distortion Block Quantisation

Figure 2.17 shows a diagram of minimum distortion block quantisation. At first glance, it can be seen to consist of m independent Gaussian block quantisers, Q_i, each with their own orthogonal matrix, $\boldsymbol{P}_i$, and bit allocations, $\{b_{i,j}\}_{j=1}^{n}$. The following coding process is also described in [183].

To quantise a vector, $\boldsymbol{x}$, using a particular cluster i, the cluster mean, $\boldsymbol{\mu}_i$, is first subtracted and its components decorrelated using the orthogonal matrix, $\boldsymbol{P}_i$, for that cluster. The variance of each component is then normalised by the standard deviation to produce a decorrelated, mean-subtracted, and variance-normalised vector, $\boldsymbol{z}_i$:

$$\boldsymbol{z}_i = \frac{\boldsymbol{P}_i(\boldsymbol{x} - \boldsymbol{\mu}_i)}{\boldsymbol{\sigma}_i} \tag{2.67}$$

These are then quantised using a set of n Gaussian Lloyd-Max scalar quantisers as described in [72] with their respective bit allocations producing indices, $\boldsymbol{q}_i$[19]. These are de-

[19]The non-uniform scalar quantisers may be replaced with uniform scalar quantisers with appropriate companding and expanding, as is done in [183].

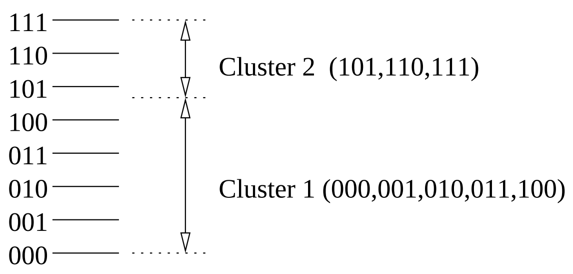


Figure 2.18: Example of quantiser level encoding and cluster number partitioning

coded to give the approximated normalised vector, $\widehat{\boldsymbol{z}}_{\boldsymbol{i}}$, which is multiplied by the standard deviation and correlated again by multiplying with the transpose, $\boldsymbol{P}_i^T$, of the orthogonal matrix. The cluster mean is then added back to give the reconstructed vector, $\widehat{\boldsymbol{x}}_{\boldsymbol{i}}$.

$$\widehat{\boldsymbol{x}}_{\boldsymbol{i}} = \boldsymbol{P}_i^T \boldsymbol{\sigma}_i \widehat{\boldsymbol{z}}_{\boldsymbol{i}} + \boldsymbol{\mu}_i \tag{2.68}$$

The distortion between this reconstructed vector and original is then calculated, $d(\boldsymbol{x} - \widehat{\boldsymbol{x}}_i)$.

The above procedure is performed for all clusters in the system and the cluster, k, which gives the *minimum distortion* is chosen:

$$k = \operatorname*{argmin}_i d(\boldsymbol{x} - \widehat{\boldsymbol{x}}_i) \tag{2.69}$$

A suitable distortion measure is chosen based on the application. For example, in spectral quantisation for speech coding, a suitable distortion measure would be spectral distortion, while for image coding, a psychovisually-inspired weighted distance measure may be used. The simplest and most general distortion measure is mean-squared-error. It is noted that the search complexity is a function of the number of clusters, m, rather than the bitrate [183]. Thus it is a computational advantage over full search vector quantisers where the search complexity is an exponential function of the bitrate [59].

2.5.5 Quantiser Index Encoding

Each quantised vector has associated with it, a number identifying which cluster was used for coding. As proposed in [183], this side information can be made inherent in the encoding. For an m cluster system operating at b bits per vector, $\log_2 m$ bits are required to uniquely identify each cluster. Therefore, on average, $b - \log_2 m$ bits are available for quantising each vector which is equivalent to $2^b/m$ quantiser levels. Hence, our range of quantiser levels has been partitioned into m segments.

In effect, this partitioning of the range of quantiser levels allows the cluster number to be found by determining which partition the block code belongs to. An example of this encoding scheme is shown in Figure 2.18 where a total of 3 bits are available to encode each block and the system uses 2 clusters. Cluster 1 has been assigned 5 levels (2.322 bits) while cluster 2 has the remaining 3 levels (1.583 bits). If cluster 1 was deemed the most suitable for encoding the current block, its quantiser levels would be contained within the range of $000 \ldots 100$. Therefore, the decoder can easily determine which cluster the block belongs to by working out which partition the code falls into. Hence this removes the need for extra side information to be transmitted[20].

The example also shows that the number of levels assigned to the block quantiser belonging to each cluster are not powers of two and hence the bits assigned to the quantisers are fractional.

2.5.6 Computational Complexity and Memory Requirements

As described by Subramaniam and Rao [183], one of the salient features of the GMM-based block quantiser is the independence of computational complexity and memory requirements to the bitrate. This contrasts with the unconstrained vector quantiser, whose codebook, and therefore storage requirements as well as search complexity, grows exponentially with the number of bits. Rather than using a non-uniform scalar quantiser, where quantiser levels need to be stored, a uniform scalar quantiser with appropriate companding and expanding functions is a fast and efficient alternative [183].

Table 2.1 shows complexity of each operation of the m cluster, n dimensional GMM-based block quantiser. It can be observed that the complexity of the scheme is dependent on the number of clusters, m, and the complexity of the distortion measure used, n_{dist}.

The memory requirements of the GMM-based block quantiser, as given in [183], is given by:

$$2^{n_{CE}+1} + m(n^2 + 3n) \text{ floats} \tag{2.70}$$

[20]If the partitions were of the same size, then this scheme is equivalent to explicitly sending bits for identifying the cluster number.

Table 2.1: Bitrate independent computational complexity of the GMM-based block quantiser (after [183])

Operation	Complexity (flops)
Mean subtraction	mn
Decorrelation	$m(2n^2 - n)$
Scaling	mn
Compander + rounding + expander	$mn(2n_{CE} + 2)$
Rescaling	mn
Correlation	$m(2n^2 - n)$
Mean addition	mn
Distortion calculation	mn_{dist}
Final comparison	m
Total	$2mn(2 + 2n + n_{CE}) + mn_{dist} + m$

2.6 GMM-based Block Quantisation using the Discrete Cosine Transform

By observing Figure 2.17, we can see that a transformation and inverse transformation is performed for all clusters. Each cluster, with its own local statistics, will possess its own unique transformation matrix. In additional to this, because the orthogonal bases of the transform space vary between clusters, then any distance measure such as MSE will need to be calculated in the original space, thus the need for inverse transformations. It all comes down to the source dependent nature of the KLT. We may remove the extra steps of computation by replacing the KLT with a source independent transform that possesses similar properties. If we assume that the role of the KLT is to decorrelate samples before scalar quantising them independently, and that the source is Gauss-Markov with a high degree of correlation, then the discrete cosine transform (DCT) is a good candidate as it has been shown to approach the optimality of the KLT under these conditions [116]. Therefore, this DCT-based scheme, which we refer to as the *GMM-DCT-based block quantiser*, will be particularly suited to the coding of images, where the source is known to be well approximated by a Gauss-Markov process.

Suboptimality in the GMM-DCT-based block quantiser will be a result of the fixed transform as well as the suboptimality of the DCT itself. With the KLT-based scheme, multiple transforms are designed for each cluster which are adapted to the correlation of that region. Therefore, we should expect the GMM-DCT-based block quantiser to

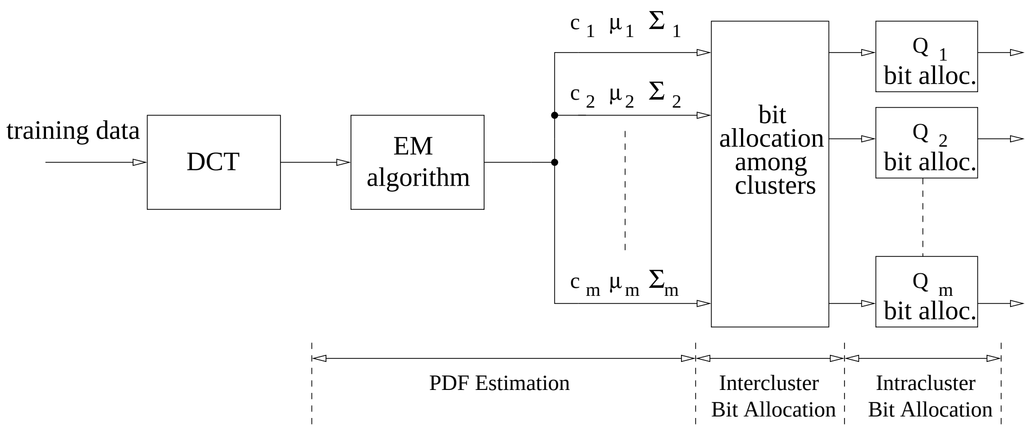


Figure 2.19: PDF estimation and bit allocation procedure for the GMM-DCT-based block quantiser

perform reasonably well for sources which have a dominant global correlation. As we have observed in the PDF examples in Section 2.5.2, there is a large dominant global correlation in consecutive pixels of images, which is mostly due to the large proportion of smooth and uniform regions, where pixels tend to have similar values of luminance. Therefore, we will apply the GMM-DCT-based block quantiser to low complexity image coding. Despite the fixed transform, the GMM-DCT-based block quantiser retains the advantage of accurate PDF modelling and the capturing of multiple modes, thus it is expected to perform better than traditional block quantisers.

2.6.1 PDF Estimation

Figure 2.19 shows the block diagram of the PDF estimation procedure for the GMM-DCT-based block quantiser. By using a GMM, the PDF is modelled as a composite source of Gaussians. Assuming that these mixture components are themselves Gauss-Markov processes, then applying a DCT on each mixture component should not impact much on the distortion performance. Since the DCT is source independent and fixed, then only one transform needs to be performed. Therefore, each image block is transformed first by the DCT and the GMM is estimated based on the DCT coefficients. It is assumed that the vectors will be decorrelated by the DCT, hence only diagonal covariance matrices are used in the EM algorithm.

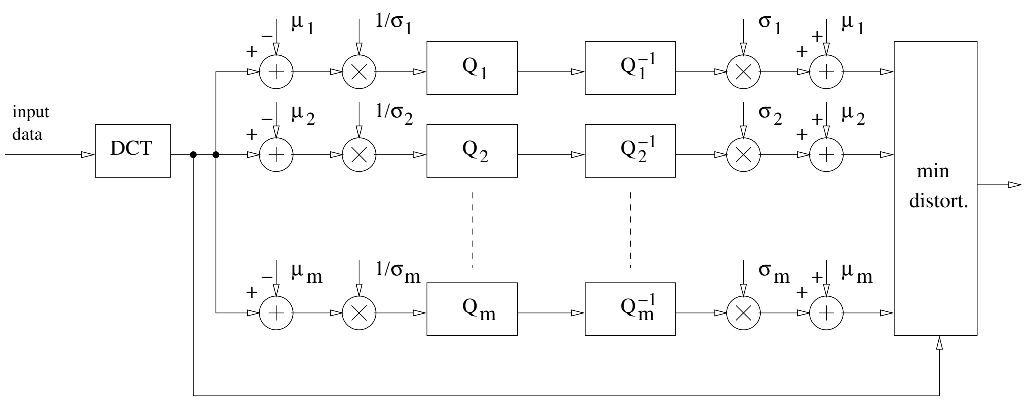


Figure 2.20: Schematic of the modified GMM-based block quantiser based on DCT (Q – block quantiser)

2.6.2 Minimum Distortion Block Quantisation

Figure 2.20 shows the block diagram of minimum distortion block quantisation for the GMM-DCT-based coder. As the DCT bases are constant, there is no need for multiple transformations as well as inverses for distance measurements. That is, the MSE distortion calculation can be performed in the DCT domain. It is helpful to compare the computational complexity (in flops[21]) of the two types of transformations. The computational complexity of a single KLT is $2p^4 - p^2$ or 8128, assuming 8×8 blocks. For an m-cluster system, the number of flops is therefore $4mp^4 - 2mp^2$ (including inverse transformation). For the case of the DCT, the computational complexity is constant at $4p^3 - 2p^2$ or 1920 for a block of 8×8 for all values of m. Therefore, for a 16 cluster GMM-based block quantiser, the number of flops used for the transformation step in the DCT-based scheme is less than 1% of that required in the KLT-based scheme.

2.7 GMM-Based Block Quantisation with Memory

2.7.1 Introduction

Correlation or memory often exists between the components of successive vectors which constitutes coding redundancy. The block quantiser and GMM-based block quantiser described thus far, exploit intravector (within a vector) correlation only. Therefore, coding

[21]In this study, we consider each multiplication and addition to represent one floating point operation (flop)

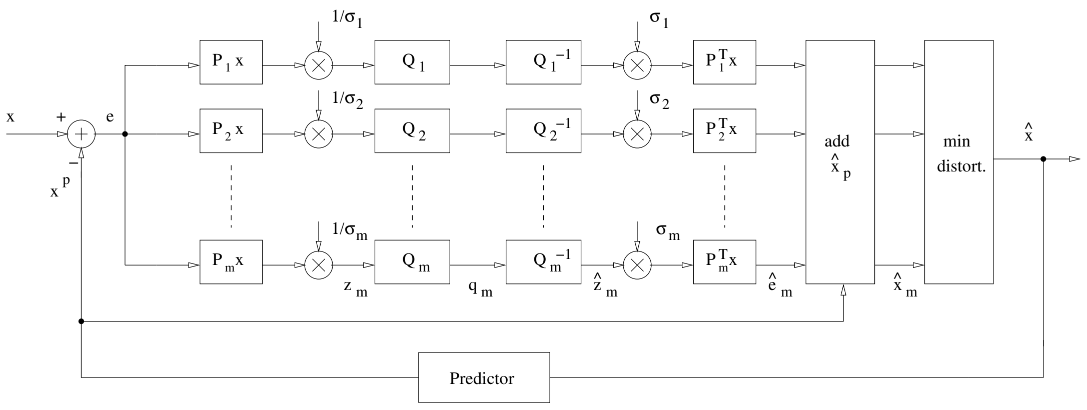


Figure 2.21: Schematic of the 'modified case' predictive GMM-based block quantiser using first-order predictor (Q – block quantiser)

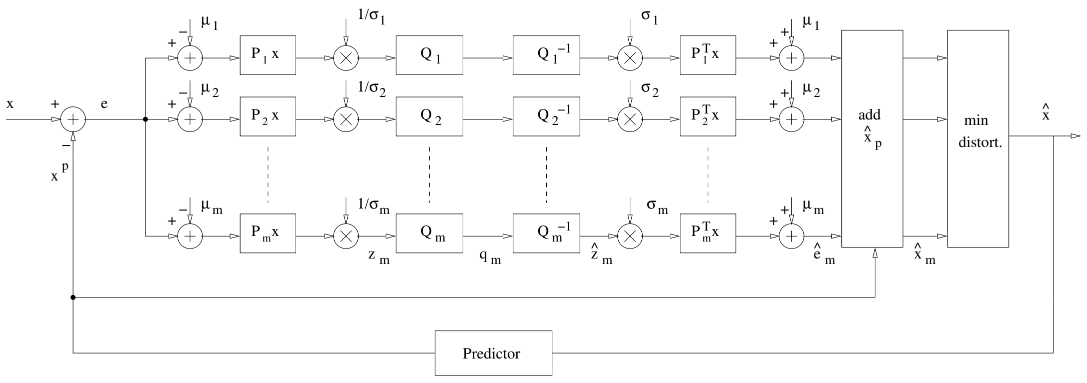


Figure 2.22: Schematic of the 'trained case' predictive GMM-based block quantiser using first-order predictor (Q – block quantiser)

gains are expected to be achieved if intervector (across vectors) correlations are exploited by the quantisation scheme. In this section, two GMM-based block quantisation schemes which exploit memory across vectors are described.

2.7.2 Predictive GMM-Based Block Quantisation

Two configurations of a GMM-based block quantiser with memory were described in [183], where the differences between successive LSF frames are quantised, similar to differential pulse code modulation (DPCM) and predictive vector quantisation [183, 55]. In the first configuration (Figure 2.21), referred to as the 'modified case' [183], existing codebooks and transformation matrices for the memoryless scheme are used for quantising the difference vectors with no subtraction of the cluster mean. In the second configuration (Figure

2.22), referred to as the 'trained case' [183], the codebooks and transformation matrices are trained based on vector differences.

The first-order predictor tries to predict the current vector, $\boldsymbol{x}_i$, using the previous quantised vector, $\hat{\boldsymbol{x}}_{i-1}$, using the following linear prediction equation:

$$\boldsymbol{x}_i^p(j) = a_j \hat{\boldsymbol{x}}_{i-1}(j) \text{ where } j = 1, 2, \ldots, n \tag{2.71}$$

where a_j is the jth linear prediction coefficient and $\boldsymbol{x}_i^p$ is the predicted vector of the current vector, $\boldsymbol{x}_i$. Given the training vector set, we use the previous training vector rather than the quantised one. Assuming each vector has a dimension of n and the training vector set has N vectors, we wish to minimise the mean squared prediction error[22]:

$$E = \sum_{i=2}^{N} \sum_{j=1}^{n} [\boldsymbol{x}_i(j) - a_j \boldsymbol{x}_{i-1}(j)]^2 \tag{2.72}$$

$$\frac{\partial}{\partial a_k} \left\{ \sum_{i=2}^{N} \sum_{j=1}^{n} [\boldsymbol{x}_i(j) - a_j \boldsymbol{x}_{i-1}(j)]^2 \right\} = 0$$

$$\sum_{i=2}^{N} [\boldsymbol{x}_i(k) - a_k \boldsymbol{x}_{i-1}(k)] \boldsymbol{x}_{i-1}(k) = 0 \tag{2.73}$$

Rearranging to find the kth linear prediction coefficient:

$$a_k = \frac{\sum_{i=2}^{N} \boldsymbol{x}_i(k) \boldsymbol{x}_{i-1}(k)}{\sum_{i=2}^{N} [\boldsymbol{x}_{i-1}(k)]^2} \text{ where } k = 1, 2, \ldots, n \tag{2.74}$$

The predicted vector, $\boldsymbol{x}^p$, is subtracted from the input vector, $\boldsymbol{x}$, to give an error vector, $\boldsymbol{e}$, which is quantised by the minimum distortion block quantiser. The predicted vector is added to the quantised error vectors on the output of each cluster block quantiser and the one which incurs minimum distortion is chosen. This quantised vector, $\hat{\boldsymbol{x}}$, is also fed back to the predictor.

As with most predictive coding schemes, problems occur when the input vectors have rapidly changing statistics, where the intervector correlations become low [45]. In these cases, the predictor performs poorly which increases the occurrence of quantised vectors having large errors. In order to reduce the number of these outlier vectors, a 'safety-net' scheme [45] may be used, where each vector is quantised using both the memoryless and memory-based quantiser. The quantised vector from both schemes are compared with the

[22]The following derivation is equivalent to the covariance method of linear prediction [14].

original and the one with the least distortion is chosen.

2.7.3 Multi-Frame GMM-Based Block Quantisation

Another method of exploiting correlation across components is to use the KLT. Also, an important result of Shannon's rate-distortion theory [155] is that better efficiency is gained by increasing the dimensionality of the vectors to be quantised. Because KLTs are present in the block quantisers of the GMM-based block quantisation scheme, decorrelation between p successive vectors of dimension n can be achieved by concatenating them into a longer vector of dimension np and performing an $np \times np$ KLT. That is, we concatenate p successive vectors, $\{\boldsymbol{x}^{(i)}\}_{i=1}^{p}$, to form longer vectors:

$$\left[x_1^{(1)}, x_2^{(1)}, \dots, x_n^{(1)}\right]^T + \left[x_1^{(2)}, x_2^{(2)}, \dots, x_n^{(2)}\right]^T + \dots + \left[x_1^{(p)}, x_2^{(p)}, \dots, x_n^{(p)}\right]^T \Longrightarrow$$
$$\left[x_1^{(1)}, x_2^{(1)}, \dots, x_n^{(1)}, x_1^{(2)}, x_2^{(2)}, \dots, x_n^{(2)}, \dots, x_1^{(p)}, x_2^{(p)}, \dots, x_n^{(p)}\right]^T \quad (2.75)$$

By doing this, the redundancies that exist across p consecutive frames can be exploited by the KLT, which decorrelates the LSFs within each frame, as well as of other frames. Also, the dimensionality of the vectors is increased, which as a result of rate-distortion theory, should lead to improvements in the distortion-rate performance.

The multi-frame GMM-based block quantiser is equivalent to the idea of *matrix quantisation* [187], where matrices of vectors are quantised. The distortion measure between each matrix is the sum or average of the distortion measure of each vector within the matrix. If the individual distortion measures are convex functions and differentiable, then their sum or average will also have the same properties [187].

The disadvantages of this scheme include higher computational complexity, delay, and memory requirements. Also, correlation between vectors on the boundary of the concatenated vectors is not exploited by this scheme. As opposed to the predictive schemes, however, the multi-frame GMM-based block quantiser is less prone to producing outlier vectors with large errors that are caused by rapidly changing statistics. Also, only minimal structural changes to the GMM-based block quantisation scheme are required.

2.8 Non-Integer Bit Allocation and Encoding at a Fixed-Rate

2.8.1 Introduction

Traditionally, the design of individual scalar quantisers for each respective component of a block is done by allocating quantiser bits to components based on their relative variances. This is the method presented in [72] which leads to the non-uniform bit allocation formula (2.43). An inherent problem with fixed-rate block quantisation is how to handle binary words of unequal lengths, as a result of the non-uniform bit allocation [198]

The constraint used in the minimisation derivation is that the sum of the bits to encode a block is constant, and so fixed-rate coding is assumed. That is:

$$b_{tot} = \sum_{i=1}^{n} b_i \tag{2.76}$$

where n is the size of the blocks, b_i is the number of bits allocated to the ith component, and b_{tot} is the total number of bits in the bit budget. However, the number of bits in (2.43) is assumed to be an unbounded continuous variable, which is not true in practice [55]. The number of bits for scalar quantisation is typically a non-negative integer, hence the optimality, in the high resolution sense, is not preserved. The solution to this is to set negative bit allocations to zero, truncate fractional numbers of bits to integers, and compensate for the bits gained or lost from the previous two procedures heuristically in order to approach the fixed bitrate. Segall [154] derived a bit allocation formula which added a further constraint, namely that the number of bits assigned to each component must be non-negative.

The closest one can get to a continuous number of bits in fixed-rate coding is to use integer quantiser levels [199], which is similar to what is done in variable-rate coders like JPEG [196]. By noting that the relation between quantiser levels, l, and quantiser bits, b, is $b = \log_2 l$, then (2.76) can be written as:

$$\begin{aligned} \log_2 l_{tot} &= \sum_{i=1}^{n} \log_2 l_i \\ &= \log_2 \prod_{i=1}^{n} l_i \end{aligned}$$

Table 2.2: Example of integer bit allocation table

3	2	1	1

Table 2.3: Example of binary coding a block

101	11	0	0	x

$$l_{tot} = \prod_{i=1}^{n} l_i \tag{2.77}$$

which effectively says that in terms of quantiser levels, the *product* of all the individual levels, l_i, must equal the total, $l_{tot} = 2^{b_{tot}}$. While this conversion from bits and levels is fairly easy, the encoding of quantiser levels in a block is not so straightforward. To demonstrate the problem, it is best to use a simple example.

Consider a fixed-rate cluster block quantiser, in the context of GMM-based block quantisation, that uses blocks of size 4 and the number of bits allocated to coding each block is 7.492 bits. This means that the target bitrate is 1.873 bits per sample. Also, assume the integer bit allocation calculated for the source is shown in Table 2.2. One can see that in terms of integer bits, the allocated bitrate is only 1.75 bits per sample, rather than the target 1.873 bits per sample. Therefore, under-allocation has occurred. However, the encoding of a block is fairly straightforward. If the quantiser indices for each component are shown in Table 2.3 (where the 'x' shows an unused bit that is padded on to ensure constant bitrate). Then this block would be encoded as 10111000_2 or 184_{10}.

However, consider an allocation table based on integer levels in Table 2.4. The total levels allocated are $9 \times 5 \times 2 \times 2 = 180$ while the target number of levels is $2^{7.492} \approx 180$. The effective bitrate achieved when using quantiser levels is 1.8729 bits per sample which is very close to the target bitrate of 1.873. Therefore when using quantiser levels, more of the bit budget is utilised and this should translate to better performance. Another way of achieving a fractional bitrate is using variable-rate entropy coding. This is what is

Table 2.4: Example of fractional bit allocation table

9	5	2	2

essentially done in JPEG, where scalar quantisers, based on quantiser levels, are applied on discrete cosine transform coefficients [196]. However, in contrast with the levels-based block quantiser considered in this paper, JPEG uses uniform scalar quantisation coupled with runlength and Huffman encoding which is known to significantly outperform fixed-rate block quantisation, at high resolutions [198]. However, we are only considering a fixed-rate quantiser which does not have the added complexity of variable-rate schemes, such as buffering.

However, the method of encoding these levels is not as straightforward as encoding bits. The complication arises from the fact that fractional bits are used and it is not well-defined as to how one can allocate a fractional part of the bit budget to each component, remembering that the total number of levels consists of a *product* of the individual component levels rather than a *sum*. For example, in the first component of Table 2.4, where 9 levels are allocated, this corresponds to approximately 3.17 bits while in the second component, 3 levels corresponds to approximately 2.322 bits, etc. That is, how is one to combine these fractional bits so that the block can be coded into an integer of 180 levels or 7.492 bits? In order to develop an understanding into how one may approach the solution to this problem, one needs to investigate positional value number systems and its generalisations.

2.8.2 Positional Value Number Systems

Most common number systems used are *positional value systems* where the value of a digit is determined by its position [186]. For example, consider the number 3512_{10}. The value is determined by multiplying each digit with its positional value or weight, w. That is, $3512_{10} = 3 \times 10^3 + 5 \times 10^2 + 1 \times 10^1 + 2 \times 10^0$. Each positional weight is related to the number of states or levels that can occur in the previous position. For example, in the decimal system:

- The least significant position (position 1) can take on 10 different levels or numerals (0, 1, 2, 3, 4, 5, 6, 7, 8, 9);
- the next least significant position (position 2) can take on 10 different levels as well. Each level in position 2 represents 10 'lots' of levels of position 1, thus the positional weight is 10;

Table 2.5: System of positional weights

Position	n	...	4	3	2	1
Num. of levels	l_n	...	l_4	l_3	l_2	l_1
w	$\prod_{i=0}^{n-1} l_i$	...	$l_3 l_2 l_1$	$l_2 l_1$	l_1	1

Table 2.6: Positional weights for the S number system

Position	4	3	2	1
Num. of levels	7	8	2	5
w	80	10	5	1

- each level in position 3 represents 10 'lots' of levels of position 2 which each have the weight of 10, thus the positional weight is $10 \times 10 = 100$; and so forth.

This system of determining the positional weights is summarised in Table 2.5. As shown in the Table, the positional weights are determined by:

$$w_n = \prod_{i=0}^{n-1} l_i \text{ where } l_0 = 1 \tag{2.78}$$

For the decimal case, the number of levels, l, is equal to 10 for all positions. With this system defined, it allows other number systems to be created where each position may have different levels. For example, consider the number system, S, whose positional weighting system is shown in Table 2.6. To find the decimal equivalent of a number 2013_S:

$$\begin{aligned} 2013_S &= 2 \times 80 + 0 \times 10 + 1 \times 5 + 3 \times 1 \\ &= 168_{10} \end{aligned}$$

There are two ways of converting a decimal number into the S number system. Method 1 involves repeated division with each positional weight. One obtains the digits by truncating the quotient and continuing the division on the remainder as shown in Table 2.7. Method 2 involves repeated division with the number of levels. One obtains the digits via the remainders, as shown in Table 2.7.

It can be seen that the mapping from any number system of defined levels to the decimal system is a one-to-one mapping and is reversible.

Table 2.7: Example of converting decimal to the S number system via method 1

80	168	
10	rem 8	**2**
5	rem 8	**0**
1	rem 3	**1**
	rem 0	**3**

Table 2.8: Example of converting decimal to the S number system via method 2

5	168	
2	33	rem **3**
8	16	rem **1**
7	2	rem **0**
	0	rem **2**

2.8.3 Fractional Bit Encoding

Using the concept of positional value number systems where any arbitrary system of defined levels can be mapped to a decimal number, it is possible to apply the technique to quantising blocks using a fractional number of bits, within a fixed-rate framework.

Consider a fixed-rate GMM-based block quantiser which operates on vectors of dimension n with the total bit budget, b_{tot}. After the application of (2.65), b_i bits are assigned to the ith cluster block quantiser, where b_i may be fractional. After the application of (2.43), the bit allocation is obtained for each component of this cluster block quantiser, $\boldsymbol{b}_i = [b_{i,1}, b_{i,2}, b_{i,3}, \dots, b_{i,n}]$, which are fractions and these are converted to integer quantiser levels, $\boldsymbol{l}_i = [l_{i,1}, l_{i,2}, l_{i,3}, \dots, l_{i,n}]$. The encoding number system for this cluster block quantiser is the same as Table 2.5 consisting of a series of positional weights, $\boldsymbol{w}_i = [w_{i,1}, w_{i,2}, \dots, w_{i,n}]$, which are calculated via (2.78). These positional weights remain fixed since the bit allocation is based on the static GMM and KLT parameters. Each transformed block, $\boldsymbol{y} = [y_1, y_2, \dots, y_n]$, is quantised to produce levels indices, $\boldsymbol{q} = [q_1, q_2, \dots, q_n]$. These level indices of the block are then encoded into a b_{tot}-bit integer, z, via the following formula:

$$z = \boldsymbol{w}\boldsymbol{q}^T \tag{2.79}$$

$$= \sum_{i=1}^{n} w_i q_i \tag{2.80}$$

In order to decode this code, z, into the respective levels for the block, the procedure shown in either Table 2.7 or 2.8 is followed.

It is useful to show that the range of integers that result from this fractional bit encoding is within that of a b_{tot}-bit integer. The maximum level for component i is $l_i - 1$, thus using (2.78) and (2.80), the maximum possible integer is given by:

$$\begin{aligned} z &= \sum_{j=1}^{n} \left[(l_j - 1) \prod_{i=0}^{j-1} l_i \right] \\ &= \sum_{j=1}^{n} \left(l_j \prod_{i=0}^{j-1} l_i - \prod_{i=0}^{j-1} l_i \right) \\ &= l_n \prod_{i=0}^{n-1} l_i - \prod_{i=0}^{n-1} l_i + l_{n-1} \prod_{i=0}^{n-2} l_i \\ &\quad - \prod_{i=0}^{n-2} l_i + l_{n-2} \prod_{i=0}^{n-3} l_i - \ldots - 1 \end{aligned} \tag{2.81}$$

The second term and third term cancel each other and the fourth cancels with the fifth, etc. in (2.81). Thus only the first term and last remain.

$$z = l_n \prod_{i=0}^{n-1} l_i - 1 \tag{2.82}$$

$$= l_n \prod_{i=1}^{n-1} l_i - 1 \text{ since } l_0 = 1 \tag{2.83}$$

$$= \prod_{i=1}^{n} l_i - 1 \tag{2.84}$$

$$= l_{tot} - 1 \tag{2.85}$$

$$= 2^{b_{tot}} - 1 \tag{2.86}$$

Therefore it has been shown that the range of the block code, z, is from 0 to $2^{b_{tot}} - 1$ which is also that of a b_{tot}-bit integer.

2.8.4 Heuristic Algorithms for Compensating Quantiser Levels

Under-allocation

'Under-allocation' can occur because of the truncation of the number of levels to make them integers. The remaining levels can then be assigned heuristically based on 'Fixed

Slope' or Pareto optimality [59] and is similar to Riskin's algorithm [147]. The idea is to approximate which component results in the most distortion drop when given an extra level while at the same time ensuring the product of the levels is equal to or below the target.

The high resolution performance of a Lloyd-Max scalar quantiser provides a way of approximating which component would lead to the largest drop in distortion if an extra level was assigned. The formula can be modified to be in terms of levels by substituting $b = \log_2 l$.

$$\begin{aligned} D(b) &= \lambda K 2^{-2b} && (2.87) \\ D(l) &= \frac{\lambda K}{l^2} && (2.88) \end{aligned}$$

The distortion drop that would result when adding an extra level is therefore approximated by:

$$\begin{aligned} \Delta D &= D(l) - D(l+1) && (2.89) \\ &= \frac{\lambda K}{l^2} - \frac{\lambda K}{(l+1)^2} && (2.90) \\ &= \lambda K \left[\frac{2l+1}{l^2(l+1)^2} \right] && (2.91) \end{aligned}$$

Therefore the variables which determine the distortion drop are the variance of the component, λ, as well as the number of levels, l, already assigned to that component. It is apparent that when l is large, the denominator of (2.91) increases faster than the numerator. Or in other words, as the operating rate of the scalar quantiser becomes higher, the performance improvement resulting from an extra level 'thins out'. Hence this agrees with the convex exponential behaviour of scalar quantisers predicted by high resolution analysis. The high resolution approximation may not be accurate at low rates so (2.91) serves as a guide only.

Once the estimated distortion drops of each component are determined, then the next step is to choose the component whereby an increase in levels will result in the total levels not exceeding the target. This can be done by calculating the change in the total product of levels if one component is increased by one.

$$l_{tot}^{(0)} = \prod_{i=1}^{n} l_i \tag{2.92}$$

If a level is added to component j:

$$\begin{aligned} l_{tot}^{(1)} &= (l_j + 1) \prod_{i=1, i \neq j}^{n} l_i && (2.93) \\ &= \prod_{i=1}^{n} l_i + \prod_{i=1, i \neq j}^{n} l_i && (2.94) \end{aligned}$$

Hence the increase in the number of levels is:

$$\begin{aligned} \Delta l &= l_{tot}^{(1)} - l_{tot}^{(0)} && (2.95) \\ &= \prod_{i=1, i \neq j}^{n} l_i && (2.96) \end{aligned}$$

If the increase in the total number of levels resulting from giving a level to a certain component is more than what is required, then the component with the second most distortion drop is tested and so forth. Once a component has been chosen, its expected distortion drop is updated.

Over-allocation

'Over-allocation' occurs when the target bitrate is low and some of the allocated bits become negative. These are set to zero which makes the total number of bits exceed the desired total. The components which, when having a quantiser level taken away, induce the least distortion increase are chosen. This procedure is repeated until the total product of the levels falls below the allocated number. Therefore, the over-allocation situation becomes one of under-allocation and the process outlined in the previous section can be followed.

In order to approximate the increase in distortion as a result of removing a quantiser level from a component, a similar derivation to the under-allocation case can be performed:

$$\begin{aligned} D(b) &= \lambda K 2^{-2b} && (2.97) \\ D(l) &= \frac{\lambda K}{l^2} && (2.98) \end{aligned}$$

The distortion increase that would result when subtracting a level is therefore approximated by:

$$\Delta D = D(l-1) - D(l) \tag{2.99}$$

$$= \frac{\lambda K}{(l-1)^2} - \frac{\lambda K}{l^2} \tag{2.100}$$

$$= \lambda K \left[\frac{1-2l}{l^2(l-1)^2} \right] \tag{2.101}$$

2.9 Chapter Summary

This chapter provided a general introduction to block quantisation, which is an example of a transform coder. The decorrelating properties of the Karhunen-Loève transform and its role in block quantisation were described. We also reviewed the discrete cosine transform as a useful alternative transform to the KLT. For sources which have Gauss-Markov properties, the DCTs decorrelating ability is similar to that of the KLT, hence the DCT is popularly used in image coding.

We provided a literature review of adaptive transform coding schemes, which resolve the problems of data non-stationarity by partitioning the vector space into local regions and designing transforms adapted to the statistics of each region. Additionally, a simple scheme using K-means clustering and local block quantisers was described and this formed a useful baseline for evaluating the recent schemes that utilise Gaussian mixture models for estimating the PDF. Following this, we gave a detailed summary of the GMM-based block quantisation scheme of [183].

In Section 2.6, we presented our modification to the GMM-based block quantiser, that replaces the KLT with a DCT. Due to the data independence property and fixed orthogonal bases of the DCT, the complexity of the new GMM-DCT-based block quantiser is considerably lowered. This modified scheme is expected to be competitive with the KLT-based GMM-based block quantiser for image coding, since images tend to have Gauss-Markov statistics and are highly correlated. In Section 2.7.3, we described our multi-frame GMM-based block quantiser, that exploits interframe correlation using the KLT. Successive frames are concatenated to produce larger frames.

In Section 2.8, a new bit encoding technique was introduced that allows the use and encoding of fractional bits in a fixed-rate block quantiser. This scheme uses the concept of a generalised positional number system and is simple in implementation. To complement this fractional bit technique, we also described some heuristic algorithms for dealing with bit allocation issues.

Chapter 3

Efficient Vector Quantisation

3.1 Abstract

In this chapter, we firstly review the vector quantiser and discuss its ‘advantages’ over the scalar quantiser. Specifically, these advantages are the space-filling advantage, the shape advantage, and the memory advantage. It is important to understand why vector quantisers always perform better than any other quantisation scheme over a given vector dimension, because this will provide the basis for our investigation on improving structured and constrained vector quantiser schemes which, despite having much lower computational and memory requirements, have suboptimal quantisation performance.

The main focus of this chapter is on improving the efficiency of the split vector quantiser (SVQ), in terms of computational complexity and rate-distortion performance. In split vector quantisers, vectors are partitioned into subvectors of lower dimensionality and these are quantised by individual vector quantisers. Though it has lower computational and memory requirements than those of the exhaustive-search vector quantiser, the vector splitting process adds numerous constraints to the codebook, which leads to suboptimal quantisation performance. Specifically, the reduced dimensionality affects all three vector quantiser advantages. Therefore, we introduce a new type of hybrid vector quantiser, called the switched split vector quantiser (SSVQ). We show that by using a switch vector quantiser stage, which is a full dimension, unconstrained vector quantiser, the SSVQ addresses the memory and shape suboptimality of SVQ, leading to better quantiser performance. In addition to this, the SSVQ has lower computational complexity than the

SVQ, though these improvements come at the expense of higher memory requirements for storing the codebooks.

Publications resulting from this research: [166, 169, 170, 171, 172, 173]

3.2 Introduction

Vector quantisation (VQ) can be viewed as a generalisation of scalar quantisation where, instead of mapping scalar values to a finite set of reproduction scalars, it maps vectors to a finite set of reproduction code-vectors. The basic definition of a vector quantiser Q of dimension n and size K is a mapping of a vector from n dimensional Euclidean space, $\mathcal{R}^n$, to a finite set, $\mathcal{C}$, containing K reproduction *code-vectors* [55]:

$$Q : \mathcal{R}^n \to \mathcal{C} \tag{3.1}$$

where $\mathcal{C} = \{\boldsymbol{y}_i; i \in \mathcal{I}\}$ and $\boldsymbol{y}_i \in \mathcal{R}^n$ [55]. Associated with each reproduction code-vector is a partition of $\mathcal{R}^n$, called a *region* or *cell*, $\mathcal{S} = \{S_i; i \in \mathcal{I}\}$ [59]. The most popular form of vector quantiser is the *Voronoi* or *nearest neighbour* vector quantiser [55], where for each input source vector, $\boldsymbol{x}$, a search is done through the entire codebook to find the nearest code-vector, $\boldsymbol{y}_i$, which has the minimum distance [152]:

$$\boldsymbol{y}_i = Q[\boldsymbol{x}] \quad \text{if } d(\boldsymbol{x}, \boldsymbol{y}_i) < d(\boldsymbol{x}, \boldsymbol{y}_j) \quad \text{for all } i \neq j \tag{3.2}$$

where $d(\boldsymbol{x}, \boldsymbol{y})$ is a distance measure between the vectors, $\boldsymbol{x}$ and $\boldsymbol{y}$. The regions in a nearest neighbour vector quantiser are also called *Dirichlet* or *Voronoi regions* [103] and these are shown in Figure 3.1, where the mean squared error (MSE) is used as the distance measure. Depending on the coding application, other more meaningful distance measures may be used such as the Mahalanobis distance [105], Itakura-Saito distance [76], or other perceptually-weighted distance measures [108].

If the dimension of the vectors is n and a codebook of K code-vectors is used, each vector will be represented as a binary code of length $\lceil \log_2 K \rceil$ bits. Hence the bitrate of the vector quantiser is given by $\frac{1}{n} \lceil \log_2 K \rceil$ bits/sample [152].

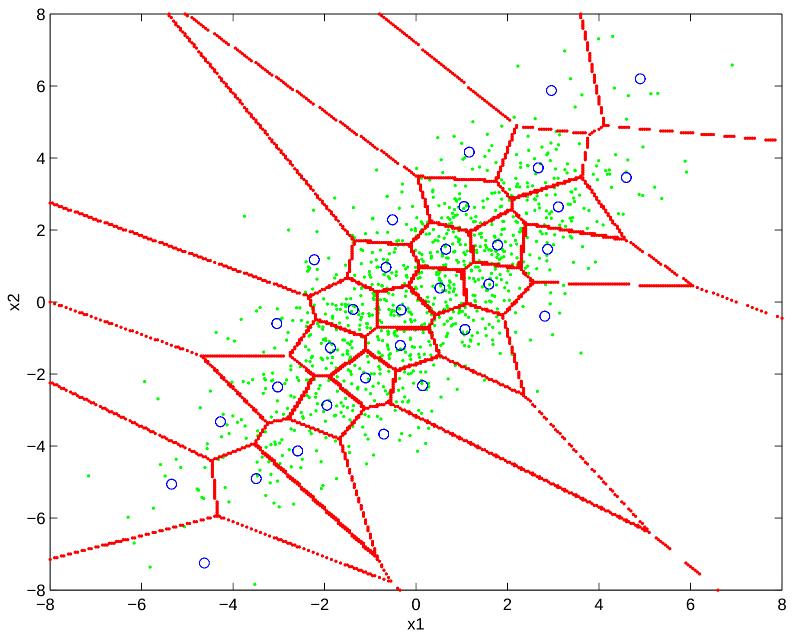


Figure 3.1: Voronoi regions of a nearest neighbour (MSE) vector quantiser, showing the input vectors (dots), code-vectors (circles), and hyperplanes defining each Voronoi region (lines)

3.3 Vector Quantiser Design Using the Linde-Buzo-Gray Algorithm

The Lloyd conditions for optimality form the basis of the methods of Lloyd [101] and Max [111] for designing minimum distortion non-uniform scalar quantisers. These conditions are stated as follows [101]:

1. The best reproduction value within a partition is the centroid; and
2. the best partition is formed by values which are closest to the centroid (nearest neighbour condition)[1].

Using these conditions, it is possible to design optimum non-uniform scalar quantisers for any arbitrary density in an iterative fashion by continually finding new centroids (condition 1), adjusting the cell boundaries (condition 2), re-calculating centroids (condition 1), *ad infinitum*. The distortion is guaranteed to decrease or remain the same after each iteration, giving a local optimal solution. Lloyd [101] and Max [111] derived tables of quantiser levels

[1]For the scalar case, the boundaries of each partition are mid-way between the reproduction values.

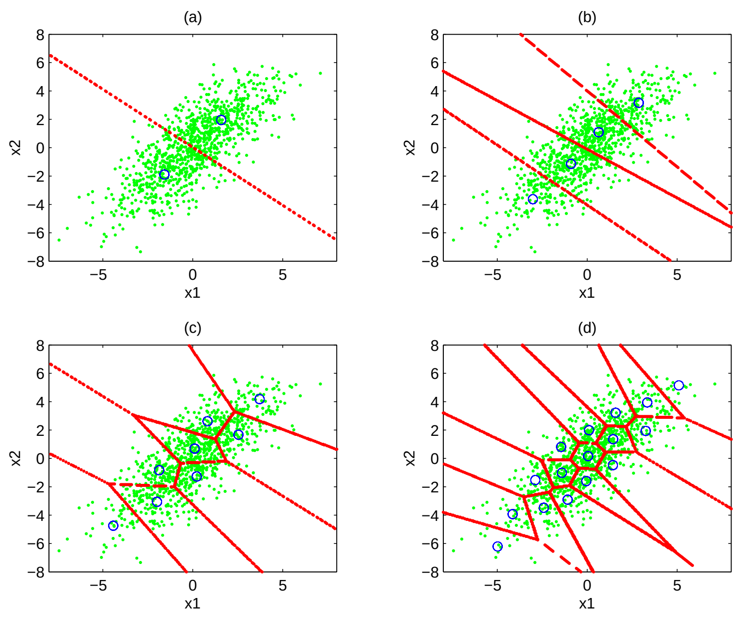


Figure 3.2: Successive code-vectors and Voronoi regions during the design of a 4 bit vector quantiser using the LBG algorithm: (a) After first split and refinement; (b) after second split and refinement; (c) after third split and refinement; (d) after final split and refinement.

for the Gaussian density while Paez and Glisson [120] provided quantiser tables for other densities such as gamma and Laplacian.

The *Linde-Buzo-Gray* (LBG) algorithm, also known as the *generalised Lloyd algorithm* (GLA), is an extension of the iterative Lloyd method I [101], for use in vector quantiser design. Because the LBG algorithm is not a variational technique[2], it can be used for cases where: the probability distribution of the data is not known *a priori*; we are only given a large set of training vectors; and the source is assumed to be ergodic [100]. The LBG algorithm involves refining an initial set of reproduction code-vectors using the Lloyd conditions, based on the given training vectors. The iterative procedure is stopped after the change in distortion becomes negligible.

Linde *et al.* [100] also introduced a 'splitting' technique for initialising the LBG algorithm, rather than assume an arbitrary set of reproduction code-vectors. In the LBG splitting technique, the centroid, $\boldsymbol{y}_1^{(1)}$, of the entire training set, is split into two code-vectors, via a perturbation procedure, $\boldsymbol{y}_1^{(1)} + \boldsymbol{\epsilon}$ and $\boldsymbol{y}_1^{(1)} - \boldsymbol{\epsilon}$. Then the training set is classified to these two code-vectors, based on the nearest neighbour criterion, and the cen-

[2]That is, it does not require the evaluation of derivatives, as opposed to Lloyd method II [100].

troids of these two clusters are refined using the Lloyd conditions to give the code-vectors for a 1 bit vector quantiser, $\boldsymbol{y}_1^{(2)}$ and $\boldsymbol{y}_2^{(2)}$. These code-vectors are split and the process continues until we reach the desired number of code-vectors, $\{\boldsymbol{y}_i^{(k)}\}_{i=1}^{K}$, at the kth step. Figure 3.2 shows the code-vectors and Voronoi regions after each successive step of the LBG algorithm.

3.4 Advantages of Vector Quantisation

Shannon [156] showed that quantising a vector of data points is more efficient than quantising individual scalar values, in the rate-distortion sense. When the dimension of the vectors is arbitrarily large, the operational rate-distortion function of the vector quantiser can approach the Shannon limit [59]. Therefore, for a given bitrate and dimension, the vector quantiser will, theoretically, incur the least distortion of any quantiser. The question to ask is why, and by how much, does a vector quantiser gain over the scalar quantiser?

Makhoul [108] noted four properties of vectors that should be exploited by quantisers in order for them to "result in optimal performance". These are namely: *linear dependency*, *non-linear dependency*, *probability density function shape*, and *dimensionality*. Each of these properties are described briefly below.

3.4.1 Linear and Non-Linear Dependency

Linear dependency refers to the correlation between vector components. As seen in the previous chapter on block quantisation, correlation between components constitutes redundancy. A quantiser that can exploit linear dependencies or correlation between vector components will result in better quantiser performance. With the exception of vectors originating from a Gaussian source, non-linear dependency between vector components remains, even after decorrelation [108]. Block quantisers and transform coders, which utilise the KLT to decorrelate the vectors, cannot exploit non-linear dependencies, hence they are suboptimal. An optimum quantiser should be able to exploit both linear and non-linear dependencies between vector components.

3.4.2 Dimensionality

In higher dimensions, cell shapes[3] become an important factor in determining quantiser performance. Working in higher dimensions allows an optimal quantiser to have the flexibility of using different cell shapes. For example, scalar quantising each component of a two dimensional vector with a uniformly distributed PDF results in square cells only, with the reproduction code-vector appearing in the centroid [108]. If we are considering MSE as the distortion measure, then the MSE incurred when representing all random vectors within a square cell by its centroid, is given by [108]:

$$E_s = \frac{\Delta^4}{6} \tag{3.3}$$

where Δ is the length of each side of the square cell. If we consider hexagonal-shaped quantiser cells, which are possible only for quantisers that operate at and are 'aware' of two (or, higher) dimensions, then the MSE incurred when representing vectors within the cell by its centroid, is given by [108]:

$$E_h = \frac{5\sqrt{3}}{8}\delta^4 \tag{3.4}$$

where δ is the length of each side of the hexagonal cell. Assuming that the cell sizes are equal[4] and neglecting edge effects, comparing the MSEs of the two cell shapes [108]:

$$\begin{aligned} \frac{E_h}{E_s} &= \frac{5\sqrt{3}}{9} \\ &= 0.962 \end{aligned} \tag{3.5}$$

Therefore, we can see that using hexagonal cells results in approximately 0.17 dB less MSE than when using the square cells of scalar quantisation, for the same number of bits. If the distortions of both shapes are made equal, then it can be shown that the hexagonal cells are about 1.94% larger than square cells, hence they can cover the same area and incur the same quantiser distortion with lesser cells, which corresponds to a saving of 0.028 bits [108].

[3]In higher dimensional space, regions bounded by numerous hyperplanes are termed as *polytopes* [108]. However, for the sake of simplicity, we will refer to them as cells.

[4]Which means that they have equal areas, hence the number of square cells to cover a certain area is the same as that of hexagonal cells. Since we have the same number of cells, then they require the same number of bits.

In summary, going to higher dimensions allows more freedom in choosing cell shapes that either minimise distortion for the same number of bits, or use less bits to achieve the same amount of distortion.

3.4.3 Probability Density Function Shape

The distortion of a quantiser is also highly dependent on how well it matches the probability density function of the data. Therefore, an optimal quantiser places more quantiser reproduction values in regions of high probability and less in regions of low probability. *Cell size*, rather than shape, is important in this regard, as smaller cell sizes allow a closer packing of reproduction vectors (or, cell centroids).

Lookabaugh and Gray [103] described three 'advantages' of the vector quantiser over the scalar quantiser and showed that they addressed Makhoul's properties of vectors [108]. They quantitatively defined the vector quantiser advantage, $\Delta(n, r)$, where n is the dimensionality and r is the power of the distortion measure[5], as the "ratio of the distortion of repeated scalar quantisation to that due to vector quantisation". Assuming a stationary source, the vector quantiser advantage can be decomposed into three parts:

$$\Delta(n, r) = F(n, r)S(n, r)M(n, r) \tag{3.6}$$

where each of the terms on the right denote the *space-filling advantage*, the *shape advantage*, and the *memory advantage*, respectively [103]. These will be described in the next subsections.

3.4.4 Space-Filling Advantage

According to [103], the space-filling advantage is the ratio of the coefficient of quantisation of scalar quantisation, $C(1, r)$, to that of n-dimensional vector quantisation, $C(n, r)$. That is:

$$F(n, r) = \frac{C(1, r)}{C(n, r)} \tag{3.7}$$

[5]$r = 2$ is mean squared error [103]

Since Gersho [53] conjectured the coefficient of quantisation to be dependent on the inverse of the polytope or cell volume. This means that a quantiser whose cells can fill more space will have a smaller coefficient of quantisation. For a two-dimensional ($n = 2$) vector quantiser using MSE ($r = 2$) as the distortion measure, the optimal cell shape is the hexagon [55, 103]. As we have observed with regards to the dimensionality property of vectors, hexagonal cells occupy more area than square cells (scalar quantisation). Hence $C(2, 2) < C(1, 2)$ which means $F(2, 2) > 1$.

Therefore, due to the space-filling advantage, which utilises Makhoul's dimensionality property [108], vector quantisers will always do better than scalar quantisers, regardless of the source PDF, in the high resolution sense.

3.4.5 Shape Advantage

According to [103], the shape advantage, $S(n, r)$, is dependent on the shape of the marginal PDF. Values of $S(n, 2)$, calculated for standard densities (Gaussian, Laplacian, Gamma), show that the vector quantiser is expected to perform at least 1.14 dB (for a Gaussian density and $n = 2$) better than the scalar quantiser, due to the shape advantage alone [103]. In this case, the vector quantiser is able to exploit the PDF shape property described by Makhoul [108].

3.4.6 Memory Advantage

According to [103], the memory advantage, $M(n, r)$ is the ratio of the $n/(n+r)$th "norms of the vector's probability density and the product of its marginals". Therefore, the memory advantage is dependent on how much statistical dependency exists between vector components, and is therefore related to Makhoul's linear and non-linear dependence property of vectors [108]. For independent and identically distributed (iid) vectors, the source PDF is equivalent to the product of the marginal PDFs, thus the memory advantage for iid vector sources is equal to 1 [103]. In other words, the vector quantiser has no memory advantage when quantising vectors whose components are statistically independent.

An interesting result that can be drawn from these vector quantiser advantages, is that when quantising vectors that are statistically independent, though there will be no

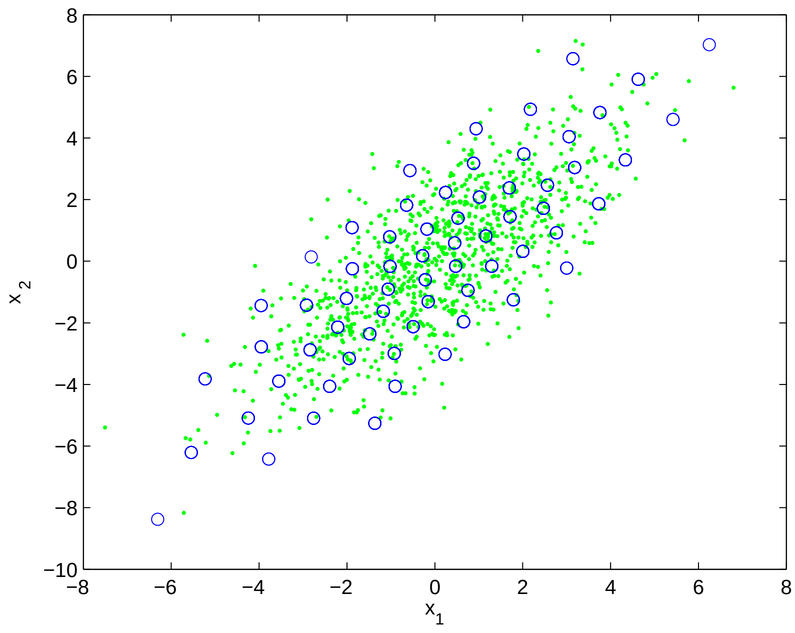


Figure 3.3: Vector quantisation using 3 bits/sample (SNR=17.21 dB) of random Gaussian vectors with covariance matrix given by (2.33)

memory advantage, the vector quantiser will manage to outperform the scalar quantiser because of the space-filling and shape advantages [103]. Figure 3.3 shows a 3 bits/sample vector quantiser applied on random Gaussian vectors with a covariance matrix give by (2.33). Because the source is Gaussian, the KLT will produce independent components, or in other words, remove all dependencies (linear and non-linear). This fact allows us to compare the block quantiser with the vector quantiser having no memory advantage and relying solely on the space-filling and shape advantages. Based on the high resolution results of [103], the expected vector quantiser advantage, $\Delta(2,2) = F(2,2)S(2,2)$, is 1.31 dB. The SNR of the 3 bits/sample block quantiser was 16.70 dB while the SNR of the vector quantiser is 17.21 dB, giving an advantage of roughly 0.5 dB. Therefore, even on independent vectors, the vector quantiser will always perform better than the scalar quantiser. The discrepancy between high resolution theory and experiment is mostly due to the low bitrate. Comparing the locations of reproduction code-vectors in Figures 2.6 and 3.3, it can be said that the code-vectors of the scalar quantiser are constrained to lie in a rectangular grid, unlike those of the vector quantiser, which are not constrained but are free to appear anywhere, forming arbitrary cell shapes.

3.5 Practical Limitations of Unconstrained Vector Quantisation

Though the unconstrained vector quantiser is the optimal quantiser for achieving the lowest distortion at a given bitrate and dimension, its exponentially-growing computational complexity and memory requirements often render it impractical for applications where a high bitrate is required. A b bit, n dimensional vector quantiser will possess a codebook of 2^b code-vectors. In terms of the memory requirements and computational complexity, this vector quantiser needs to store $n2^b$ floating point values and compute $3n2^b - 1$ kflops/vector[6], respectively, when using the mean squared error as the distance measure.

For image coding applications, blocks of $8 \times 8 = 64$ are typically quantised by transform coders such as JPEG [196], and this is considered the optimum block size [198, 116]. Quantising each block using a full search, unconstrained vector quantiser at a bitrate of 0.25 bits/pixel, requires 262 kflops/pixel of computations and 4.2 million floats of memory for storing the codebook. In narrowband speech coding, linear predictive coding (LPC) parameter vectors of dimension 10 need to be quantised at 20 bits/vector using a full search, unconstrained vector quantiser, to achieve transparent quality. This corresponds to a computational complexity of about 42 Mflops/vector and a memory requirement of 10.5 million floating point values.

Another problem related to vector quantiser design is that at high bitrates and dimensionality, where the codebook is large, the LBG algorithm will require a larger amount of training vectors in order to design a codebook that is representative of the source. Otherwise, there will be too much 'over-fitting' of the training set [66].

3.6 Product Code Vector Quantisers

3.6.1 Introduction

In order to make vector quantisers practical for large dimension and high bitrates, a structure can be imposed on the codebook to decrease the search complexity and/or storage requirements. One way of achieving this is to use decompose the codebook into

[6]In our calculations, each addition, multiplication, and comparison is considered a floating point operation (flop).

a Cartesian product of smaller codebooks [59, 55]. The idea of using product codes was first introduced by Sabin and Gray [149] with their shape-gain vector quantiser, in which vectors are quantised using both a shape codebook and gain codebook. Indices for each codebook are then transmitted to the decoder, which reconstructs the vector using its stored shape and gain codebook.

The definition of a product code vector quantiser is one with a codebook, $\mathcal{C} = \{\boldsymbol{y}_i\}_{i=1}^{K}$, that consists of m codebooks, $\{\mathcal{C}_i\}_{i=1}^{m}$, where $\mathcal{C}_i = \{\boldsymbol{u}_{i,j}\}_{j=1}^{K_i}$, such that [55]:

$$\mathcal{C} = \mathcal{C}_1 \times \mathcal{C}_2 \times \ldots \times \mathcal{C}_m \tag{3.8}$$

where $\times$ symbolises the Cartesian product. The code-vectors, $\boldsymbol{u}$, in each of the smaller codebooks, combine in a permutative fashion to form the product code-vectors, $\boldsymbol{y}$. Therefore, the *effective* size of the product codebook is [55]:

$$K = \prod_{i=1}^{m} K_i \tag{3.9}$$

However, the actual size of the product codebook, which consists of the sum of the individual codebook sizes, is generally smaller than that of the effective product codebook [55]:

$$K \geq \sum_{i=1}^{m} K_i \tag{3.10}$$

The advantages are that with smaller codebooks, code-vector searches are reduced and in most cases, the memory requirements are reduced as well. These come at the cost of suboptimal coding performance as the product code vector quantiser codebook has structural constraints [55]. Also, the lowest complexity product code vector quantisers typically use sequential or independent searching and design and this leads to suboptimal product code-vectors being chosen or generated [55]. Also the issue of bit allocation among the various codebooks arises [59], which can often complicate design and lead to further suboptimality.

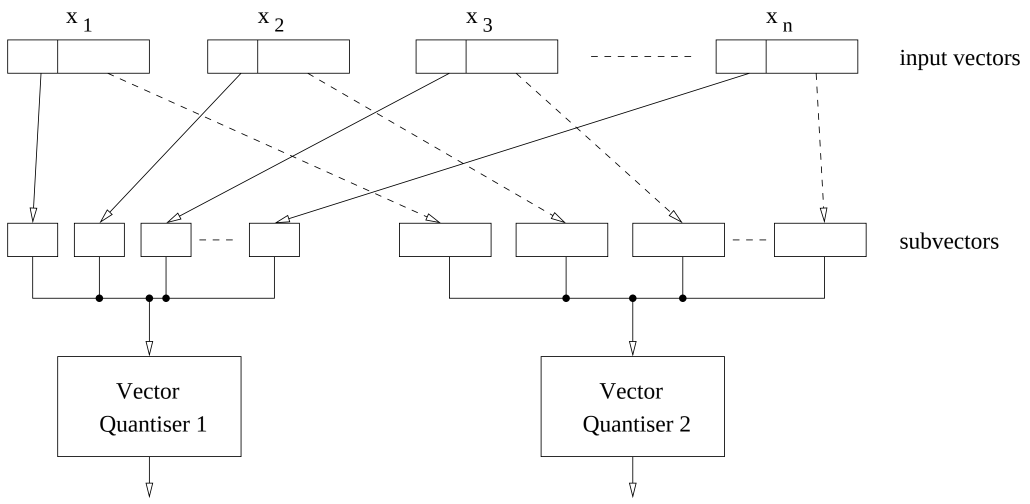


Figure 3.4: Block diagram of a two part split vector quantiser (after [87])

3.6.2 Split Vector Quantisers

An m part, n dimensional *split vector quantiser* (SVQ)[7] [123] operating at b bits/vector, divides the vector space, $\mathcal{R}^n$, into m lower dimensional subspaces, $\{\mathcal{R}_i^{n_i}\}_{i=1}^m$, where $n = \sum_{i=1}^m n_i$. Independent codebooks, $\{\mathcal{C}_i\}_{i=1}^m$, operating at $\{b_i\}_{i=1}^m$ bits/vector, where $b = \sum_{i=1}^m b_i$, are then designed for each subspace.

Figure 3.4 shows the block diagram of a two part split vector quantiser. In order to quantise a vector of dimension n, the vector is split into subvectors of smaller dimension. Each of these subvectors are then encoded using their respective codebooks. The memory and computational requirements of the SVQ codebook are smaller than that of an unstructured VQ codebook. In terms of the number of floating point values for representing the SVQ codebooks as opposed to that of unstructured VQ:

$$\sum_{i=1}^{m} n_i 2^{b_i} \leq n 2^b \tag{3.11}$$

while the effective number of code-vectors of the resulting product codebook is the same as that of unstructured VQ at the same bitrate:

$$\prod_{i=1}^{m} 2^{b_i} = 2^b \tag{3.12}$$

Therefore, the computational complexity and memory requirements of SVQ can be reduced

[7]Split vector quantisation is also known to as *partitioned vector quantisation* [55]

considerably, by splitting vectors into more parts. In fact, the transform coder can be considered a special case of SVQ (when $m = n$) operating on transform coefficients [59].

However, reductions in complexity and storage come at the cost of suboptimal coding performance. More specifically, since SVQ is quantising vectors of smaller dimension, this reduces the vector quantiser's ability to exploit the dimensionality property of vectors [108], or alternatively speaking, it reduces the space-filling advantage [103]. Due to the structural constraints of the vector splitting, the lower dimensionality also reduces the shape advantage [103]. In addition to this, the independent quantisation of the subvectors means that linear and non-linear dependencies between the components cannot be exploited, and this reduces the memory advantage. Therefore, splitting vectors into more parts decreases the coding efficiency of the vector quantisers. The transform coder makes up for this loss of coding performance through decorrelation but it cannot make up for the losses due to non-linear dependencies as well as inefficient quantiser cell shapes.

Because there are independent vector quantisers, the bits need to be allocated to each of them. Generally, assigning each subvector quantiser an equal number of bits, whenever possible, results in a good compromise between quantisation performance and complexity [123]. The split vector quantiser was first introduced by Paliwal and Atal [122, 123] for quantisation of line spectral frequencies (LSF) in narrowband CELP speech coders and is used in the Adaptive Multi-Rate Narrowband (AMR-NB) codec [2]. SVQ is also used for quantising Mel-frequency cepstral coefficients (MFCCs) in the ETSI Distributed Speech Recognition (DSR) standard [47]. These will be discussed in later chapters.

3.6.3 Multistage Vector Quantisers

Figure 3.5 shows the block diagram of another type of product code vector quantiser[8], first introduced by Juang and Gray [80], called the *multistage vector quantiser* (MSVQ). It is also referred to as a multistep, residual or cascaded vector quantiser [59]. Each vector, $\boldsymbol{x}$, is quantised by a coarse vector quantiser to produce an approximate vector, $\hat{\boldsymbol{x}}_1 = Q[\boldsymbol{x}]$. The quantisation error or residual, $\boldsymbol{e}_1 = \boldsymbol{x} - \hat{\boldsymbol{x}}_1$, is calculated and this is quantised by another vector quantiser, giving a quantised version of the residual vector, $\hat{\boldsymbol{e}}_1 = Q[\boldsymbol{e}_1]$. This process of determining the residual vector, quantising it, etc. can be continued,

[8]The codebook of the multistage vector quantiser is formed by the direct sum of codebooks from each stage. However, it can also be viewed as a product code vector quantiser, in the sense that the final codebook is formed from a Cartesian product of the individual codebooks [59].

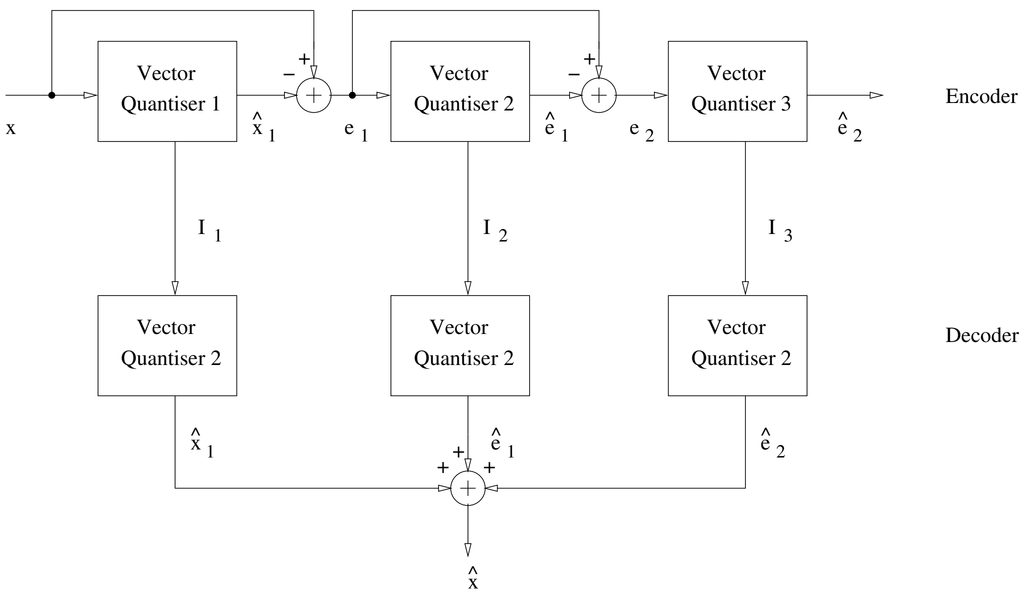


Figure 3.5: Block diagram of a three-stage multistage vector quantiser (I_i denotes the ith quantiser index)

depending on how many stages are used. The decoder takes the indices, I_i, from each vector quantiser stage and adds them to get the reconstructed vector, $\hat{\boldsymbol{x}}$:

$$\hat{\boldsymbol{x}} = \hat{\boldsymbol{x}}_1 + \sum_{i=1}^{m-1} \hat{\boldsymbol{e}}_i \tag{3.13}$$

where m is the number of stages.

Because each stage is an independent vector quantiser, bits need to be allocated to each of them. With each vector quantiser operating at a lower bitrate, the memory and computational requirements are reduced. Comparing the total size (in number of floating point values) of the codebooks of an m stage, n dimensional MSVQ operating at b bits/vector, with that of an equivalent unconstrained vector quantiser of the same bitrate and dimensionality:

$$\sum_{i=1}^{m} n2^{b_i} \leq n2^b \tag{3.14}$$

where b_i is the number of bits given to the ith stage vector quantiser and $b = \sum_{i=1}^{m} b_i$. We can see that search and memory requirements of the MSVQ, while lower than those of the unconstrained vector quantiser, are not as low as those of the split vector quantiser, where the dimensionality of the codebooks is reduced, in addition to the bitrate.

The MSVQ is generally suboptimal because of the constrained structure of the code-

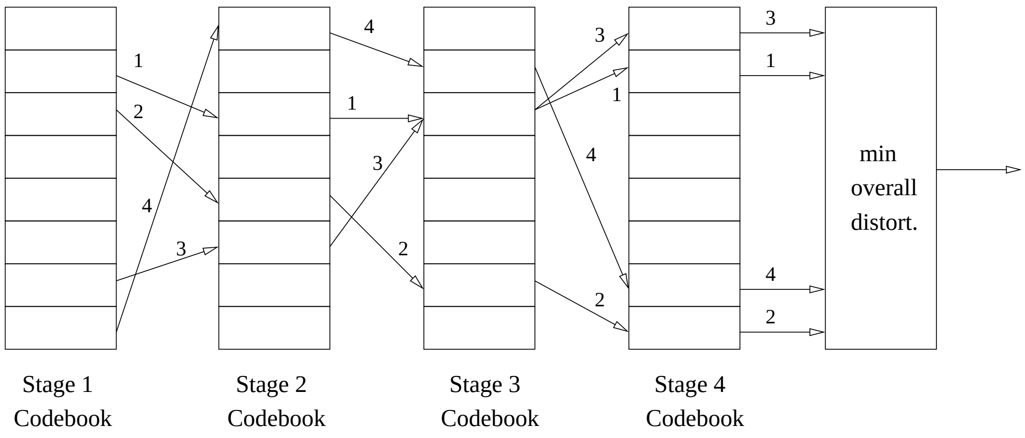


Figure 3.6: Block diagram showing codebook searches in an M-L searched four-stage multistage vector quantiser (where $M = 4$). Each of the 4 paths is labelled.

books as well as the sequential nature of the design and code-vector searches [59, 93, 90]. The greedy-based, sequential design of MSVQs, where each stage is designed independently using the LBG algorithm to minimise the distortion of the error from the previous stage, is suboptimal, in general [59]. The sequential design algorithm at each stage assumes that "subsequent stages are populated with zero vectors only" [93]. Also, the greedy-based sequential searching, where a codevector is independently selected, such that it minimises the distortion at each stage, does not generally give optimal product code-vectors[9] [59, 90], since at each stage, it is assumed that "the vectors from all subsequent stages are zero" [93].

The *M-L searched multistage vector quantiser*, also known as the tree-searched multistage vector quantiser, was introduced by LeBlanc *et al.* [93] which used a more optimal searching algorithm than the traditional sequential search. At the first stage of the MSVQ, M code-vectors are selected such that they give the least distortion. M residual vectors are calculated and a search of the second stage codebook is performed for each of the M residual vectors. This causes M paths to be created in the MSVQ. At the final stage, the path which gives the lowest overall distortion is chosen [93]. The M-L search codebook search procedure is shown in Figure 3.6 where $M = 4$. It was shown that the performance of the M-L searched MSVQ approached that of the full-search vector quantiser for small values of M [93].

[9]In contrast, sequential code-vector searches in the split vector quantiser are optimal (given the codebook), in the mean squared sense. SVQ loses its performance in the splitting of vectors, which reduces dimensionality and adds constraints to the subvector codebooks [93].

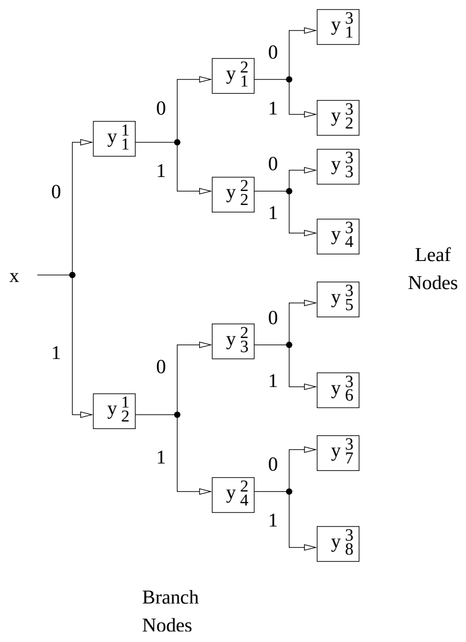


Figure 3.7: A 3 bit tree structured vector quantiser

LeBlanc *et al.* [93] also proposed several codebook design algorithms which attempt to minimise overall distortion of the reproduction algorithms. *Sequential optimisation* trains the codebook for each stage sequentially, fixing previous stage codebooks, in order to minimise overall distortion [93]. *Iterative sequential optimisation* initialises with the traditional sequential design. Then for each stage, the codebook is re-trained while keeping all other stages fixed, in order to minimise overall distortion [93].

3.7 Tree Structured and Classified Vector Quantisers

The tree structured vector quantiser (TSVQ) [26], as shown in Figure 3.7, enforces a tree structure on the vector quantiser codebook [55]. By doing so, the number of code-vector searches is considerably reduced. The greedy method of designing a TSVQ is to recursively run the LBG algorithm on training vectors that are classified to each branch [59].

To quantise a vector, $\boldsymbol{x}$, it is firstly compared with the code-vector in each branch node. The branch node code-vector which is closest to x determines the path through the tree until we reach the code-vector at the leaf node, which completes the search. For a b bit TSVQ, only $2b$ distortion calculations are required, as opposed to 2^b searches in an unconstrained, exhaustive search vector quantiser. Therefore, the computational complexity of TSVQ is very low [59]. However, the memory requirements of the TSVQ codebook are higher than the unconstrained vector quantiser. For an n-dimensional, b bit TSVQ, the total memory requirement (in floats) is:

$$memory_{TSVQ} = n \sum_{i=1}^{b} 2^{i} \tag{3.15}$$

Hence for our 3 bit TSVQ example, we need to store 14 code-vectors of dimension n.

The performance of TSVQ is generally suboptimal due to the sequential searching algorithm and the structural constraint on the codebook. The path that is chosen through minimising the distortion at each stage, does not necessarily terminate at the optimum code-vector [55]. Also, the greedy method of designing the TSVQ does not necessarily produce an optimal TSVQ codebook either [59].

Related to the TSVQ are the *classified vector quantiser* (CVQ), introduced by Ramamurthi and Gersho [141] and the *switched vector quantiser*, used by Wang and Gersho [197]. Instead of having the binary branch structure, CVQ and switched VQ use m branches, each leading to a codebook representing a certain class. A classifier is used on vectors to be quantised, in order to determine the best codebook.

3.8 Switched Split Vector Quantisation

3.8.1 Hybrid Vector Quantisation Schemes

In the literature, hybrid vector quantisers have been investigated, where two or more of the VQ schemes are combined. Examples include the two stage vector quantisation-lattice vector quantiser (VQ-LVQ) by Pan and Fischer [132] and tree structured two stage vector quantisation-pyramidal lattice vector quantiser (TSVQ-PLVQ) by Pan [133]. A hybrid of split and classified vector quantisation was investigated by Zhou and Shoham [205]. The computational cost is reduced by replacing the full search vector quantisers with classified vector quantisers while the quantisation performance remained about the same as that of the full search-based SVQ [205].

In this section, we investigate the use of a hybrid of switch vector quantisation and split vector quantisation, called the *switched split vector quantiser* (SSVQ). It is different than the hybrid scheme considered by Zhou and Shoham [205], where they split vectors to be quantised by CVQs. In our scheme, vectors are classified using an exhaustive search switch vector quantiser and are then quantised by individual split vector quantisers. The advantage of classifying before splitting is that global dependencies between vector components are exploited in the first stage. Also, the suboptimality of splitting is then limited to local regions rather than the entire vector space. Hence, it will be shown that the SSVQ provides a better trade-off than traditional split vector quantisers in terms of bitrate and distortion performance, and offers a lower computational complexity, though at the cost of an increase in memory requirements.

3.8.2 Suboptimality of Split Vector Quantisers

As we have discussed in Section 3.6.2, the coding performance of split vector quantisation is suboptimal because vector quantisers are designed independently within each lower dimensional subspace. This condition causes linear and non-linear dependencies that exist across these subspaces to not be exploited [55]. In addition to this, the vector splitting constrains the shape of the quantiser cells as well as the ability of the code-vectors to match the marginal PDF shape. In essence, all three vector quantiser advantages described by Lookabaugh and Gray [103] are affected by the vector splitting. In order to illustrate the shortcomings of SVQ and how it may be improved, we consider the simple case of designing a two-part SVQ for two dimensional vectors and examine each type of suboptimality.

Loss of the Memory Advantage

It is readily apparent that when the number of parts, m, is equal to the vector dimension, n, SVQ becomes equivalent to scalar quantising each vector component independently [123]. Similar to that from [55], Figure 3.8(a) shows the probability density function of vector data with two correlated components, x and y. For a 4 bit SVQ (2 bits per partition), each vector, $[x, y]$, is partitioned and a codebook is designed for each partition based on the marginal PDFs shown in Figs. 3.8(b) and (c). The resulting SVQ has an effective codebook of 16 code-vectors, as shown in Figure 3.8(d), while coding requires only $2^2 + 2^2 = 8$ searches. It can be observed that the resulting SVQ is suboptimal, as there are 8 product code-vectors which do not fall in areas of finite probability. In other words, the vector

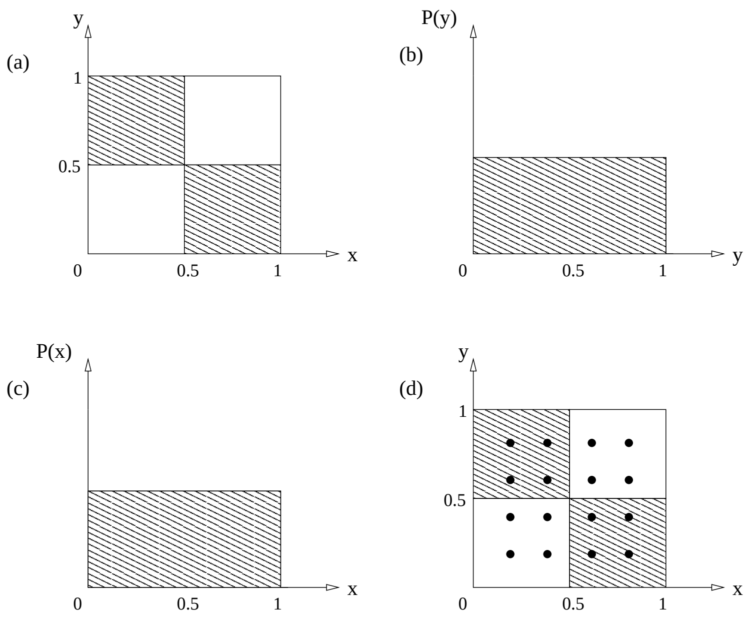


Figure 3.8: Illustrating the memory suboptimality of a 4 bit two-part split vector quantiser in two dimensions (after [55]) (a) Two dimensional PDF (shaded areas indicate uniform probability, white areas indicate zero probability); (b) Marginal PDF of component y; (c) Marginal PDF of component x; (d) Product codebook of 4 bit split vector quantiser

partitioning in SVQ does not allow for the exploitation of dependencies between subvectors, x and y. One would expect an improvement in the quantisation performance if subvectors are first 'decorrelated' before independent quantisers are designed. This is analogous to *block quantisation* or *transform coding* [72], where a Karhunen-Loève Transform (KLT) is used to decorrelate individual vector components before they are independently coded using non-uniform scalar quantisers.

Loss of the Space Filling Advantage

Figure 3.9 shows the product code-vectors of the two-part SVQ operating on two dimensional Gaussian random vectors that are statistically independent. The covariance matrix of this random source is:

$$\mathbf{\Sigma} = \begin{bmatrix} 6 & 0 \\ 0 & 1 \end{bmatrix} \tag{3.16}$$

For each vector quantiser, the best polytope in one dimension is the *interval* [103]. This results in the product code-vectors lying on a rectangular grid, hence the Voronoi regions are rectangular. However, in two dimensions, the optimum cell shape in two dimensions

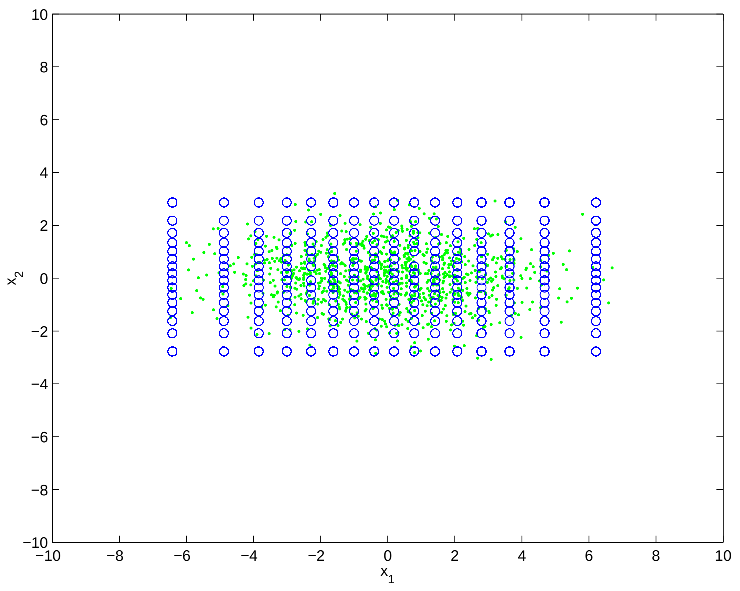


Figure 3.9: Illustrating the space-filling and shape suboptimality of an 8 bit two-part split vector quantiser in two dimensions

is the *hexagon* [53]. Therefore, we can see that the lower dimensionality has constrained the quantiser cell shape in a way that reduces the space-filling advantage of vector quantisation. For our example of two dimensional vectors, the reduced dimensionality results in a theoretical loss of 0.17 dB, according to the results of [103].

Loss of the Shape Advantage

Each of the subvector quantisers has no problem with matching the marginal PDF shape of each dimension as they are equivalent to the Lloyd-Max non-uniform scalar quantiser, which are optimal (at least, locally) for *one dimension*. However, Figure 3.9 also shows that the product code-vectors of the two-part SVQ do not match the elliptical shape of the marginal PDF in *two dimensions*. Therefore, the reduced dimensionality has constrained the shape advantage, which in the case of our example, results in a theoretical loss of 1.14 dB for a Gaussian source, according to the results of [103].

3.8.3 Switched Split Vector Quantisers

As have seen in the previous section, using the two dimensional vector space analogy, we have shown how the split vector quantiser effectively becomes equivalent to the scalar

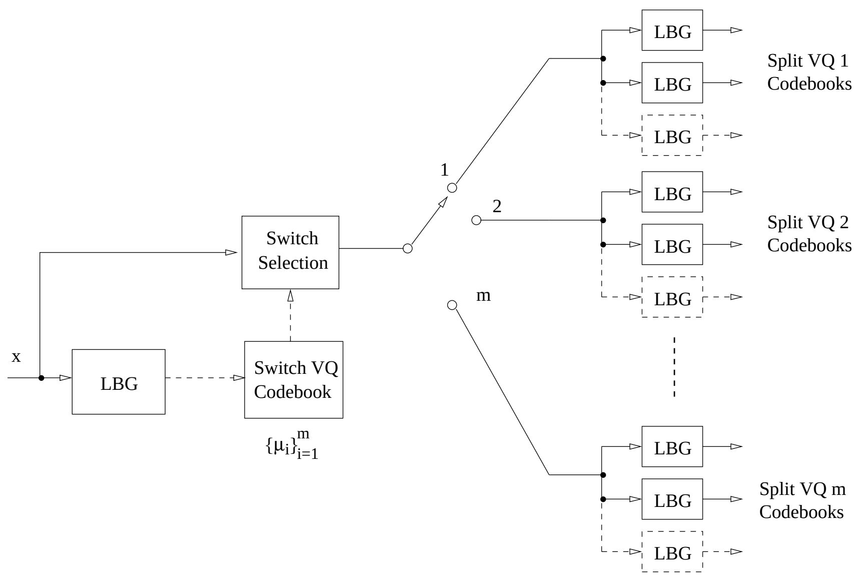


Figure 3.10: Switched split vector quantiser (training)

quantising of the vector components when the number of parts is equal to the vector dimension. In block quantisation, applying a decorrelating linear transformation on the vectors can improve the coding efficiency of the independent scalar quantisers. Likewise in the SVQ case, we need to find a way of exploiting dependencies (linear and non-linear) between the subvectors before quantising them independently using vector quantisers. This leads to our new product code vector quantiser called the switched split vector quantiser.

Unconstrained vector quantisers are well known to have the ability of exploiting linear (ie. correlation) and non-linear dependence among vector components [55]. Therefore, in SSVQ, an initial unconstrained vector quantiser (the switch vector quantiser) is used to classify[10] the vector space into Voronoi regions or clusters, which allows the exploitation of linear and non-linear dependencies across all dimensions. Then for each cluster, a local split vector quantiser is designed. The novelty of SSVQ is to populate the vector space with a number of different split vector quantisers such that they are positioned to exploit global dependencies. Each split vector quantiser is adapted to the local statistics of the Voronoi region and the suboptimalities of SVQ are localised.

Figure 3.10 shows a schematic of SSVQ codebook training. The LBG algorithm [100] is first applied on all vectors to produce m centroids (or means), $\{\boldsymbol{\mu}_i\}_{i=1}^{m}$. In the Euclidean distortion sense, these centroids are the best representation of all the vectors in that

[10]As opposed to other classified vector quantisation schemes, we use unsupervised classification in order to exploit vector component dependencies.

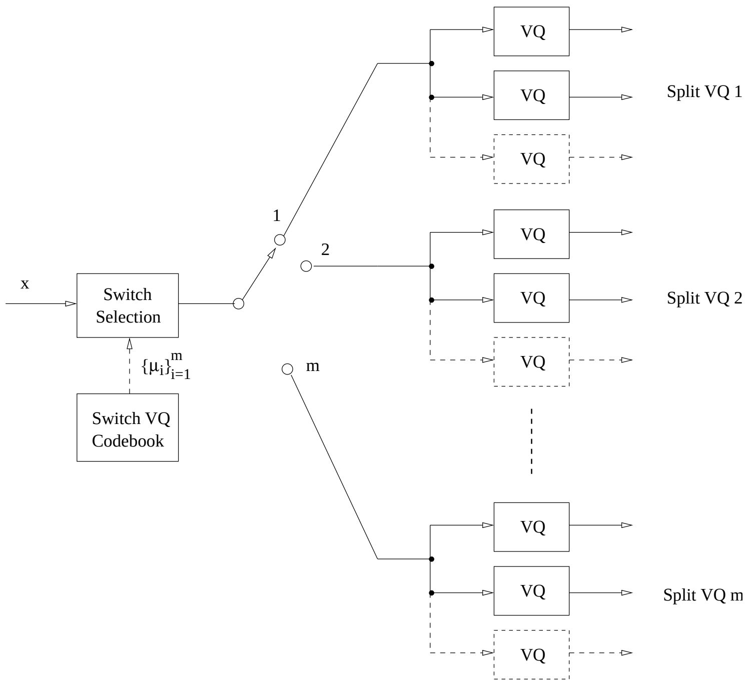


Figure 3.11: Switched split vector quantiser (coding)

region. Hence, we can use them to form the *switch vector quantiser codebook* which will be used for switch-direction selection. All the training vectors are classified based on the nearest-neighbour criterion:

$$j = \underset{i}{\operatorname{argmin}}\, d(\boldsymbol{x} - \boldsymbol{\mu}_i) \tag{3.17}$$

where $\boldsymbol{x}$ is the vector under consideration, $d(\boldsymbol{x} - \boldsymbol{\mu}_i)$ is a suitable distortion measure between two vectors, and j is the cluster (or, switching direction) to which the vector is classified. With the training vectors classified to the m clusters, local SVQ codebooks are designed for each cluster (or, switching direction) using the corresponding training vectors.

Figure 3.11 shows a block diagram of SSVQ coding. Each vector to be quantised is first switched to one of the m possible directions based on the nearest-neighbour criterion defined by (3.17), using the switch VQ codebook, $\{\boldsymbol{\mu}_i\}_{i=1}^m$, and then quantised using the corresponding SVQ.

The SSVQ decoder needs to know which SVQ was chosen to quantise each vector. This can be done by transmitting the switching direction number as explicit side information. However, a more efficient way is to implicitly code this information by partitioning the quantiser index range, similar to the one in GMM-based block quantisers [183]. Assuming

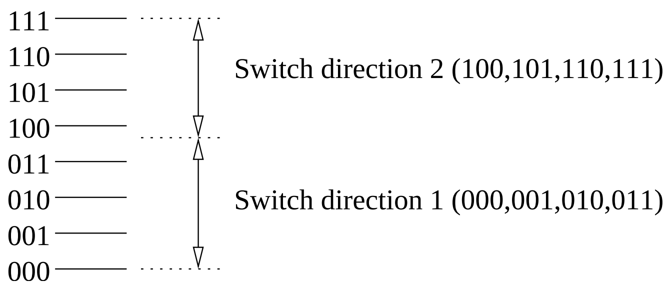

111
110
101
100
011
010
001
000
Switch direction 2 (100,101,110,111)
Switch direction 1 (000,001,010,011)

Figure 3.12: Quantiser index partitioning for 3 bit SSVQ with two-way switch

an SSVQ operating at b bits per vector and m-direction switch, we require $\log_2 m$ bits to uniquely identify all possible switching directions. Therefore, each split vector quantiser would be given a budget of $(b - \log_2 m)$ bits or $2^b/m$ indices. This suggests that the overall quantiser index range of 2^b can be divided into m partitions, each containing the valid indices which a particular SVQ can assign. The decoder can determine which SVQ was used for encoding a vector by finding which partition the quantiser index belongs to. Figure 3.12 shows an example of dividing the quantiser index range for a 3 bit SSVQ with a two-directional switch. Since one bit is needed to uniquely identify each switching direction, 2 bits are allocated to each of the SVQs. The binary indices, $000, 001, 010, 011$, are valid for the SVQ of switching direction 1 while $100, 101, 110, 111$, are valid indices for the SVQ of switching direction 2. Note that if the same number of levels were assigned to each partition, this scheme would be equivalent to explicitly sending bits that identify the switch number.

In a minimum-distortion sense, it would be logical to quantise each vector using all the split vector quantisers and then pick the one which results in the least distortion. This is an approach taken in multiple transform domain split vector quantisation [114] and similarly, in Gaussian mixture model-based block quantisation [183]. The disadvantage with this 'soft' decision scheme is the high computational complexity which results from each vector being quantised multiple times. SSVQ reduces the complexity by employing a switch, which makes a 'hard' decision based on nearest neighbour classification in order to determine the SVQ that would most likely quantise with the least distortion. As such, there will be a distortion penalty incurred by using the 'hard' decision, though this is offset by a considerable reduction in computational complexity.

One modification that can be made, in order to lower the suboptimality introduced by

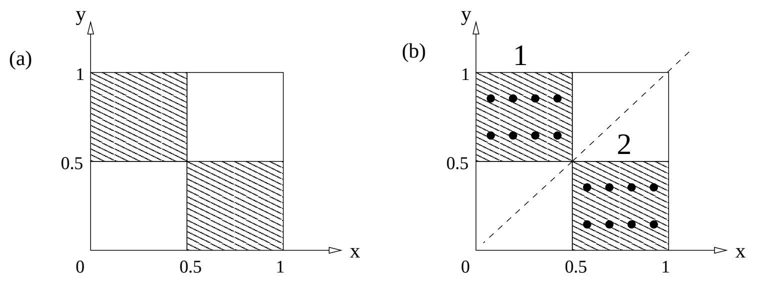


Figure 3.13: Illustrating the memory advantage of the switched split vector quantiser (a) Two dimensional PDF (shaded areas indicate uniform probability, white areas indicate zero probability); (b) Product codebook of 4 bit switched split vector quantiser (dashed line shows boundary between switch cells while numbers indicate switching direction)

the 'hard' decision made by the switch vector quantiser, is to adopt a scheme similar to the M-L search MSVQ. Specifically, instead of choosing one switching direction, we choose M switching directions, quantising using their respective SVQ codebooks, and pick the one which incurs the least quantiser distortion. This is a compromise between the 'hard' decision ($M = 1$) and a full-search, 'soft' decision ($M = m$), and quantiser performance is expected to improve, though at the expense of an increase in computational complexity.

3.8.4 Advantages of the Switched Split Vector Quantiser

The Memory Advantage of SSVQ

Returning to our two dimensional PDF example, as shown in Figure 3.13, the SSVQ (with two switching directions) quantises the entire vector space with two initial code-vectors and classifies them into two clusters. Then two-part split vector quantisers are designed for each cluster. As we can see in Figure 3.13(b), the initial unconstrained vector quantiser (which we referred above as the switch vector quantiser) has exploited the global dependencies between the two vector components and two-part split vector quantisers are positioned to reflect this dependency. In contrast to the product code-vectors of Figure 3.8(d), the SSVQ product code-vectors are better placed. Therefore, the switch vector quantiser, which is unconstrained, allows the SSVQ to exploit at least some of the global dependencies that would, otherwise, have not been exploited by SVQ. Therefore, SSVQ should recover some of the memory advantage lost due to vector splitting.

For a quantitative look at SSVQ, correlated Gaussian random vectors of two dimensions with a covariance matrix of:

$$\mathbf{\Sigma} = \begin{bmatrix} 4.5 & 4.0 \\ 4.0 & 6.0 \end{bmatrix} \tag{3.18}$$

were generated and quantised using a two-part SVQ and two-part SSVQ at 8 bits/vector. The SSVQ uses 16 switching directions and each part was allocated 2 bits. Figure 3.14(a) and (b) shows the resulting product code-vectors of the SVQ and SSVQ, respectively. It can be observed in Figure 3.14(a) that the SVQ has not exploited the correlation in the vectors, characterised by the tilt. This has resulted in a large amount of code-vectors that fall in areas where there are no vectors. In comparison, Figure 3.14(b) shows the switch code-vectors (marked as crosses) have captured the major dependency between the components, x_1 and x_2. Each of the local SVQs have been placed according to these switch code-vectors where we can see a better utilisation of the code-vectors. The 8 bits/vector SSVQ has made a 2.58 dB gain in SNR over the SVQ.

The Shape Advantage of the SSVQ

In order to observe the shape advantage of the SSVQ over SVQ, we have generated some memoryless Gaussian random vectors with a covariance of (3.16). Because the vectors have no memory, any gains in SNR are mostly due to the shape advantage. In this example, where the same vector splitting is used for both SSVQ and SVQ, there will be no space-filling advantage of SSVQ over SVQ since both split vectors into subvectors of the same dimensionality, which constrains the quantiser cell shapes to be the same (in this case, rectangular).

Figures 3.15(a) and (b) show the product code-vectors of an 8 bits/vector split vector quantiser and 8 bits/vector switched split vector quantiser, respectively. In both cases, there are a total of $2^8 = 256$ code-vectors. For the SSVQ, 16 switching directions were used and 2 bits were assigned to each subvector in the local SVQs. When observing Figure 3.15(a), we can see that the lattice of SVQ code-vectors does not match the marginal shape of the two dimensional PDF, which is elliptical. In comparison, the code-vectors resulting from SSVQ, as shown in Figure 3.15(b), match the elliptical shape of the marginal PDF more closely than the rectangular shape of the SVQ. In terms of SNR, we see that the shape advantage gain of SSVQ over SVQ is approximately 0.5 dB.

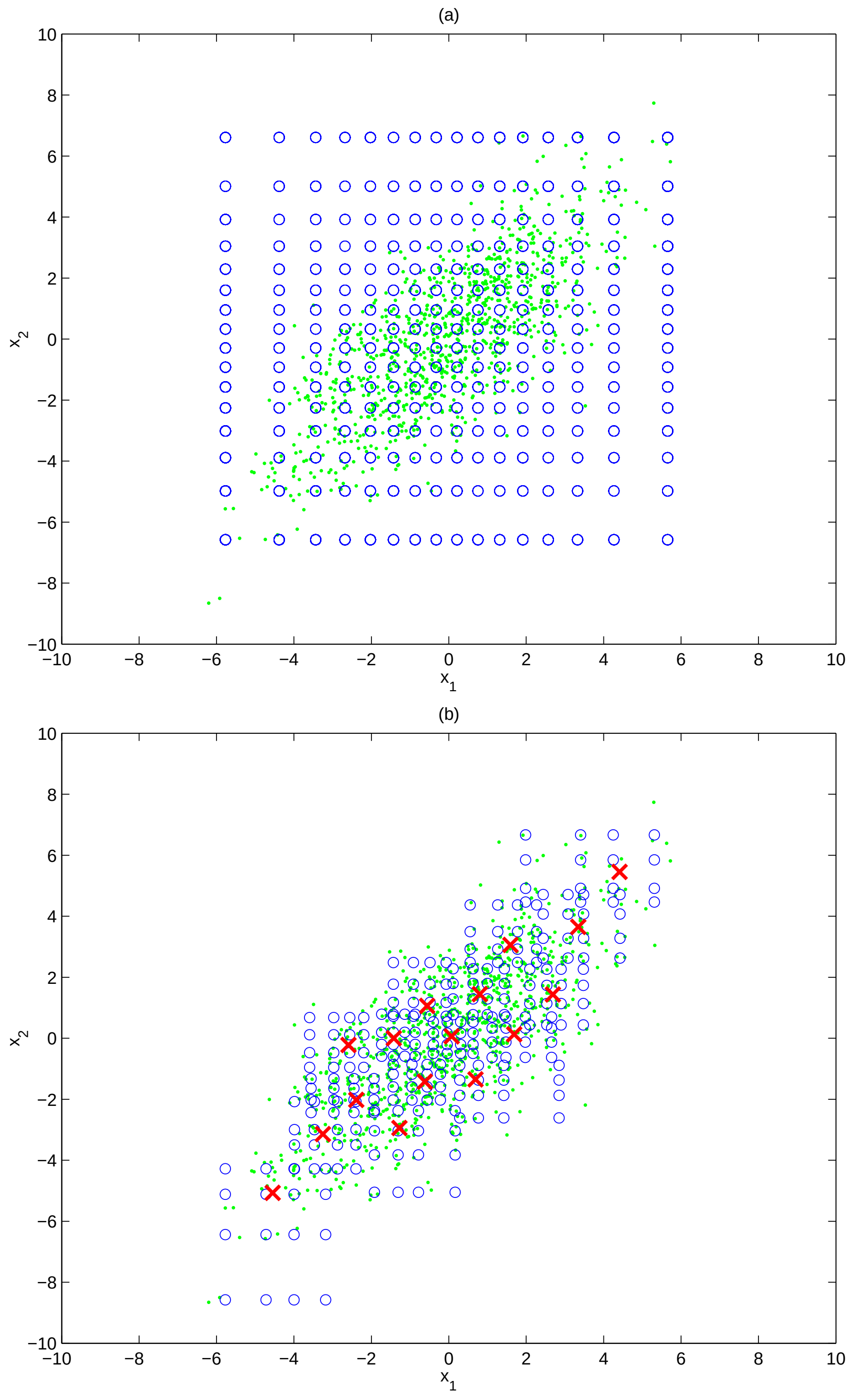


Figure 3.14: Comparing product code-vectors of: (a) 8 bits/vector split vector quantiser (SNR=20.32 dB); with (b) 8 bits/vector switched split vector quantiser (SNR=22.9 dB). The correlated Gaussian random vectors are represented as dots, the code-vectors as circles, and the switch code-vectors of SSVQ as crosses.

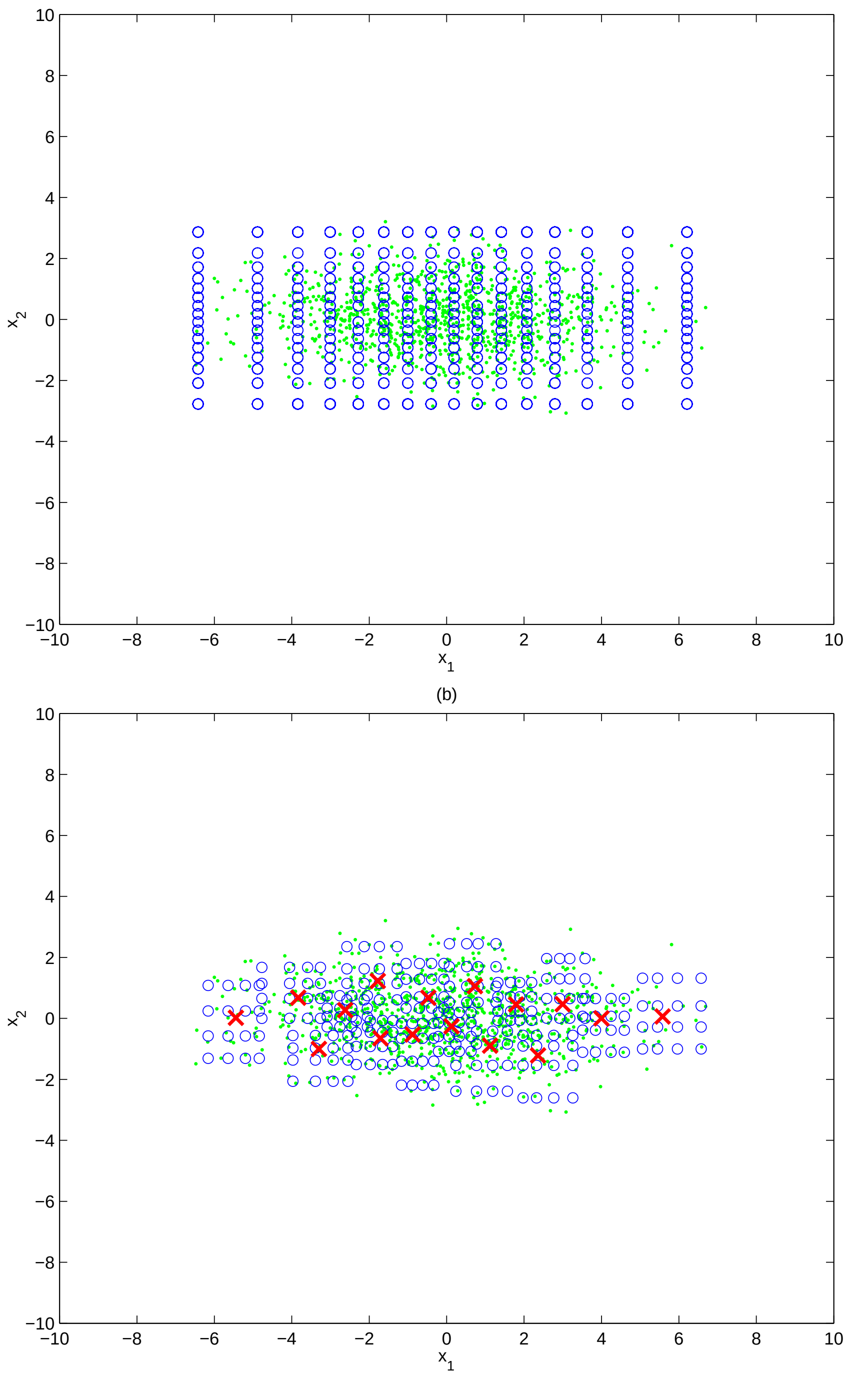


Figure 3.15: Comparing product code-vectors of: (a) 8 bits/vector split vector quantiser (SNR=21.29 dB); with (b) 8 bits/vector switched split vector quantiser (SNR=21.78 dB). The memoryless Gaussian random vectors with covariance of (3.16) are represented as dots, the code-vectors as circles, and the switch code-vectors of SSVQ as crosses.

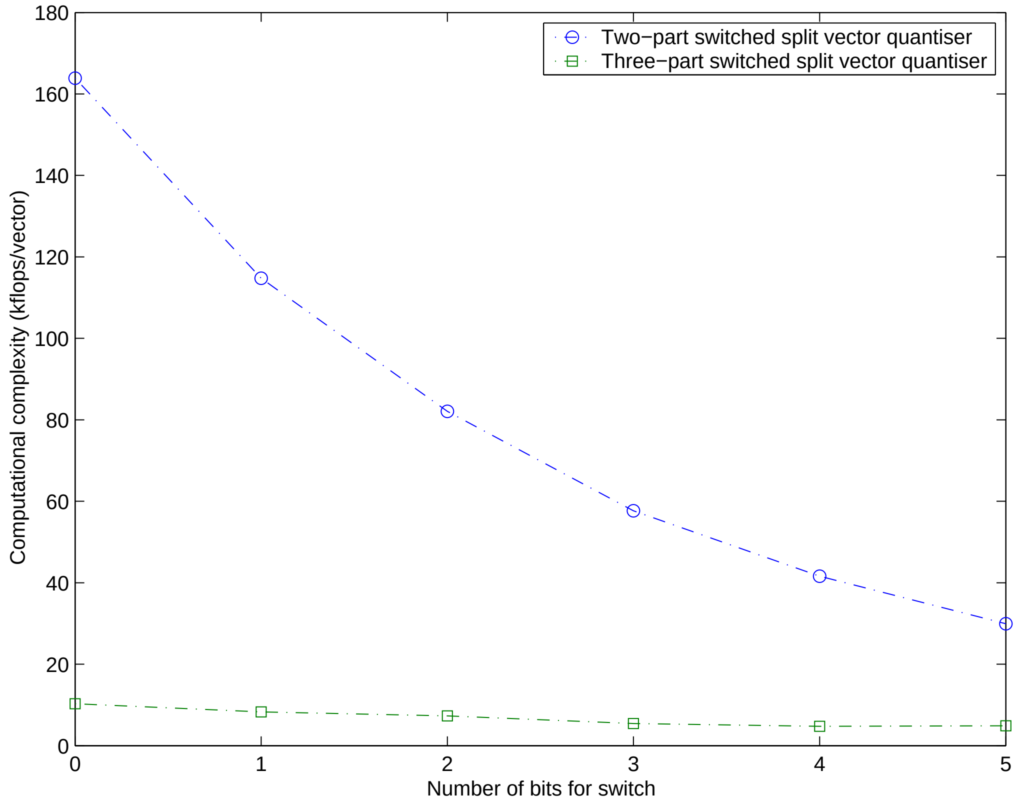


Figure 3.16: Computational complexity (in kflops/frame) of two-part and three-part 24 bits/vector SSVQ (dimension 10) as a function of number of bits for switch vector quantiser.

3.8.5 Computational Complexity and Memory Requirements

The main advantages of SSVQ are the better trade-off between bitrate and distortion and low computational complexity. In this section, we examine the latter characteristic and compare it to that of the split vector quantiser. Computational complexity will be measured in flops/vector, where each addition, multiplication, and comparison is counted as one floating point operation (flop). We also assume the use of a weighted mean squared error distance measure, similar to the one proposed in [123]. This increases the complexity of the vector quantiser from $3n2^b - 1$ to $4n2^b - 1$ flops, where n and b are the vector dimension and number of bits, respectively.

There are a number of design parameters of the SSVQ and these are stated below:

- dimension of the vectors, n;
- the number of bits allocated to the switch vector quantiser, b_m, and number of

switching directions, $m = 2^{b_m}$;

- the total number of bits, b_{tot};
- number of parts in vector splitting, s;
- dimension of the subvectors, $\{n_i\}_{i=1}^{s}$, where $n = \sum_{i=1}^{s} n_i$; and
- bits allocated to each of the subvectors, $\{b_i\}_{i=1}^{s}$, where $\sum_{i=1}^{s} b_i = b_{tot} - b_m$.

The computational complexity of the switch vector quantiser is given by:

$$complexity_{switch} = 4n2^{b_m} - 1 \tag{3.19}$$

while the complexity of each split vector quantiser is given by:

$$complexity_{SVQ} = \sum_{i=1}^{s}(4n_i 2^{b_i} - 1) \tag{3.20}$$

Therefore, the total complexity of the switched split vector quantiser is given by:

$$complexity_{SSVQ} = 4n2^{b_m} - 1 + \sum_{i=1}^{s}(4n_i 2^{b_i} - 1) \tag{3.21}$$

In order to get an idea of the reduction in computational complexity that SSVQ requires as opposed to that of the split vector quantiser, Figure 3.16 shows the computational complexity of a two-part SSVQ as a function of the number of bits, b_m, for the switch vector quantiser. The bitrate is 24 bits/vector and the vectors are of dimension 10. Vectors are split into $(4, 6)$ or $(3, 3, 4)$ for the two-part and three-part SSVQ, respectively. Bits are assigned uniformly to the subvectors whenever possible. When $b_m = 0$, the SSVQ reverts to a single split vector quantiser. We can see that the single SVQ has the highest computational complexity and as we use more switching directions, the computational complexity of SSVQ drops. This is because the number of bits available for each SVQ decreases as b_m increases. Also shown in Figure 3.16 is the computational complexity of a three-part SSVQ where we can see that the further split results in a quantiser of much lower complexity.

The memory requirements, as a number of floating point values, of the SSVQ is given

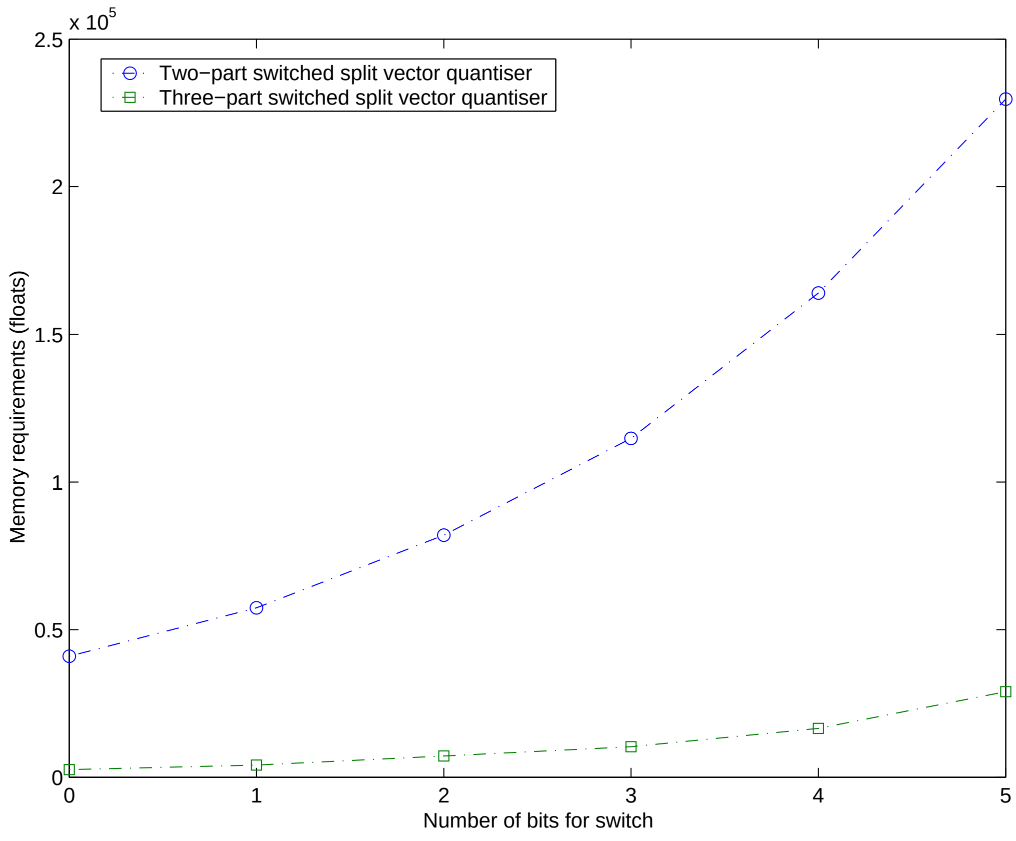


Figure 3.17: Memory requirements (in number of floats) of two-part and three-part 24 bits/vector SSVQ (dimension 10) as a function of number of bits for switch vector quantiser.

by:

$$memory_{SSVQ} = n2^{b_m} + 2^{b_m} \sum_{i=1}^{s} n_i 2^{b_i} \tag{3.22}$$

Figure 3.17 shows the memory requirements, as a number of floats, of the codebooks for the same two-part and three-part SSVQ considered above, as a function of the number of bits for the switch vector quantiser. We can see that for SSVQ, the gains made in quantiser performance and reductions in computational complexity come at the expense of an exponential increase in the memory requirements, with the SVQ ($b_m = 0$) having the lowest memory. This is a disadvantage of the SSVQ scheme and may be minimised by using more vector splitting. Looking at Figure 3.17, the memory requirements of the three-part SSVQ are not as high as the two-part SSVQ. Therefore, at this bitrate and vector dimension, the three-part SSVQ is a more practical scheme, with a moderate memory requirement.

3.9 Chapter Summary

This chapter provided a general review of vector quantisation, its advantages over the scalar quantiser, and its limitations, with regards to its exponential growth of complexity as a function of the number of bits and dimensionality. Product code vector quantisers, such as the split and multistage vector quantiser, alleviate the complexity issue by dividing the quantisation process into codebooks of lower dimensionality, or sequential and independent stages, respectively. These structural constraints though cause suboptimal quantisation performance. We have also identified and analysed the main source of suboptimality in the split vector quantiser (SVQ), namely the vector splitting which degrades the memory advantage, the shape advantage, and the space-filling advantage. In order to address at least two of these suboptimalities, we have introduced a new type of product code vector quantiser called the switched split vector quantiser (SSVQ), which consists of a hybrid of a full-dimension, unconstrained switch vector quantiser and numerous split vector quantisers. The first stage (ie. switch vector quantiser) allows the SSVQ to exploit global statistical dependencies as well as match the marginal PDF shape of the data, which would otherwise have not been exploited by normal SVQ. Also, the tree structured characteristic of the switch vector quantiser provides a dramatic reduction in search complexity. We have shown via computer simulations of 2-D vector quantisation how SSVQ is superior to SVQ in terms of quantisation performance and computational complexity. The only disadvantage of SSVQ is the increase in memory requirements.

Chapter 4

Lossy Image Coding

4.1 Abstract

This chapter investigates the application and performance of efficient block quantisation schemes to the lossy coding of greyscale images. We begin the chapter with some preliminaries of digital images, their representation, as well as their statistical properties. Also, mean squared error (MSE) and peak signal-to-noise ratio (PSNR) are defined, which are used to objectively measure the quality of the coded images. Following this, we provide a general review of image coding techniques that have been extensively investigated in the literature, focusing mainly on vector quantisation, transform coding, subband coding, and wavelet-based coding. Due to the complications and difficulties involved in the filtering of finite length signals and achieving perfect reconstruction in subband and wavelet decompositions, we provide an extensive summary of the non-causal filtering and signal extension techniques from [106]. Included are examples as well as MATLAB simulations which demonstrate, step-by-step, the process of non-expansive filtering via symmetric extension.

Following the literature review, we present results and discussion of our image coding experiments using various block quantisation schemes. The performance of the traditional block quantiser, which assumes Gaussian data and uses the Karhunen-Loève transform, is used as a baseline to highlight the improvements gained from more advanced schemes. Because images are non-stationary, correlation is not uniform throughout the image vector space, thus the need for multiple transforms, each of which are adapted to a region of rel-

ative stationarity. We investigate a simple scheme that demonstrates this concept, called the K-means-based multiple transform block quantiser. In this quantisation scheme, the image space is partitioned using the K-means algorithm and block quantisers are designed for each Voronoi region or cluster. While there are tangible improvements in the rate-distortion performance over the traditional block quantiser, there are certain inadequacies that affect the efficiency of the KLT. This leads us to GMM-based block quantisation and we show through PSNR results and visual comparisons of the reconstructed images, that this scheme performs better, in the rate-distortion sense, than the block quantisation schemes considered thus far. We also compare a GMM-based block quantiser that uses fractional bit allocations with one that uses integer bits and show that there is a finite gain in PSNR. Following this, we examine the performance of the GMM-DCT-based block quantiser, which is computationally less expensive than the KLT-based scheme. Finally, a simple method for reducing block artifacts in GMM-based block quantisation is presented, which uses the discrete wavelet transform (DWT) as a pre-processing step. We show that at the same bitrate, number of clusters, and vector dimensionality, the preprocessed GMM-based block quantiser achieves higher PSNRs and reduced block artifacts than the same scheme operating on spatial blocks only.

Publications resulting from this research: [126, 127, 128, 129, 165]

4.2 Introduction

In contrast with digital speech and audio data, image data spans over space rather than over time. The image signals are usually band-limited analog signals[1], which are converted to digital form through sampling at or above the Nyquist sampling rate [142]. As images are two-dimensional, discretisation is done in both dimensions. Each discretised point in an image is known as a *picture element*, a *pixel*, or a *pel*.

For greyscale images, each pixel has a certain scalar quantised value which represents the *luminance*. For a b-bit greyscale image, the luminance goes from 0 (black) to $2^b - 1$ (white). Therefore, for a one bit image, luminance values are either a 0 or 1. Such binary images are used commonly for facsimiles [20]. Greyscale images used in research

[1]If they are not band-limited, low-pass anti-aliasing filters are applied before sampling.

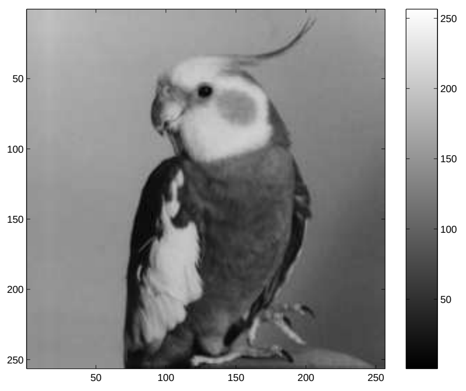


Figure 4.1: A typical 8-bit greyscale image 'bird' with mapping between colour and luminance value

are commonly 8-bit images, where each pixel has a luminance value within the range of 0 to 255 and all values in between are a certain level of grey. An example of an 8-bit greyscale image is given in Figure 4.1.

The rectangular arrangement of pixels in an image allows them to be represented as a two dimensional array. This representation allows the image to be specified as a data type for use in coding algorithms. Individual pixels can be quantised using scalar quantisers or DPCM, which exploits spatial redundancy from neighbouring pixels. Rows and columns of an image can be extracted as one dimensional vectors from the array. This allows direct manipulation by various algorithms such as subband and wavelet coding, where row and column filtering can be performed separately (assuming the use of separable filters). Likewise, due to the two dimensional nature of image data, statistical dependencies exist across both dimensions, which suggests that square or rectangular blocks of pixels are a suitable atomic data representation for block-based quantisation schemes, such as transform coding and vector quantisation, to exploit these dependencies [54].

Assuming an isotropic covariance function[2] for image data and that correlation is uniform in both dimensions, then it can be shown that the covariance function at any pixel, which is indicative of spatial correlation, decreases rapidly beyond the vicinity of 8 neighbouring pixels [20]. Therefore, a popular block size used in block-based image coding is 8×8 pixels, which forms vectors of dimension 64.

[2] $\Sigma(i, j) = \sigma^2 e^{-\alpha\sqrt{i^2+j^2}}$, where σ^2 is the variance of the image and i and j is the distance away from the reference pixel [20].

4.3 Distortion Measures for Image Coding

For images, the amount of compression is usually expressed as the *bitrate* which is measured in *bits per pixel* (bits/pixel or bpp). In other words, it is the average number of bits required to represent one pixel.

In lossy image coding, a measure of the distortion or degradation in quality is required[3]. A common measure for the loss of information is the *mean squared error* (MSE), which is calculated using the following equation:

$$MSE = \frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} \left[\boldsymbol{I}(i,j) - \hat{\boldsymbol{I}}(i,j) \right]^2 \tag{4.1}$$

where m and n are the dimensions of the image, $\boldsymbol{I}(i,j)$ and $\hat{\boldsymbol{I}}(i,j)$ are the pixel values at location (i,j) in the input and output image, respectively.

The objective quality of the resulting images in comparison with the original is usually represented by the *signal-to-noise ratio* (SNR), in decibels:

$$SNR = 10 \log_{10} \frac{\sum_{i=1}^{m} \sum_{j=1}^{n} [\boldsymbol{I}(i,j)]^2}{\sum_{i=1}^{m} \sum_{j=1}^{n} \left[\boldsymbol{I}(i,j) - \hat{\boldsymbol{I}}(i,j) \right]^2} \tag{4.2}$$

or in terms of the mean squared error:

$$SNR = 10 \log_{10} \frac{\frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} [\boldsymbol{I}(i,j)]^2}{MSE} \tag{4.3}$$

The numerator is the average power of the input image while the denominator is the average power of the error between the two images.

For images, *peak signal-to-noise ratio* (PSNR) in decibels, is commonly used as well:

$$PSNR = 10 \log_{10} \frac{(2^b - 1)^2}{MSE} \tag{4.4}$$

where b is the number of bits used to represent a pixel in the input image. The numerator is the square of the largest possible pixel value in the input image. For an 8-bit greyscale image, the numerator is equivalent to 255^2.

[3]Whereas for lossless compression, no information is lost during the compress and hence the distortion is zero

4.4 Overview of Image Coding Techniques

4.4.1 Vector Quantisation

As we have discussed in the previous chapter, vector quantisers are the best quantisers for achieving minimum distortion for a given bitrate and vector dimension. In fact, as the dimensionality increases, the operating rate-distortion performance approaches the Shannon limit.

Because of the two dimensional nature of image data, statistical dependencies exist across these dimensions. Therefore, image vectors are normally formed from two dimensional square blocks of size, $k \times k$, and vector quantisers are expected to exploit the dependencies (the memory advantage) between pixel values within the block.

Figure 4.2 shows the image 'lena', that was quantised using a 9 bit full-search vector quantiser. The bitrate was 0.5625 bits/pixel and each vector is formed from a block of 4×4 pixels. The codebook, which has 1024 entries, was trained using 18 greyscale images of the same dimension, of which 'lena' was a part of.

Gersho and Ramamurthi [54] were the first to apply this powerful technique to the problem of image coding. Square blocks of $k = 2, 3, 4$ were quantised at $6, 7, 8$ bits, respectively [54]. One observation was that the decoded images were of remarkable quality, given the very low bitrate. However, there were noticeable artifacts in edge areas, referred to as the 'staircase effect', which became more noticeable as k was increased [54]. The 'staircase effect' can be seen in Figure 4.2(b). These artifacts are attributed to the lack of suitable edge blocks in the vector quantiser codebook. This was remedied by using a classifier, which determined whether an image block contained edges or not, and two separate codebooks were then designed for each class [54]. In fact, this is equivalent to supervised classified vector quantisation.

Despite the optimality, unconstrained vector quantisers for image coding are limited to small dimensions and low bitrates due to their computational complexity and memory requirements growing exponentially. Furthermore, the training of a high bitrate vector quantiser requires a very large training set. Mismatch between the trained codebook and an image to be quantised, which is not part of the training set, also leads to degraded quality. Therefore, the training set should be large and representative of the images to

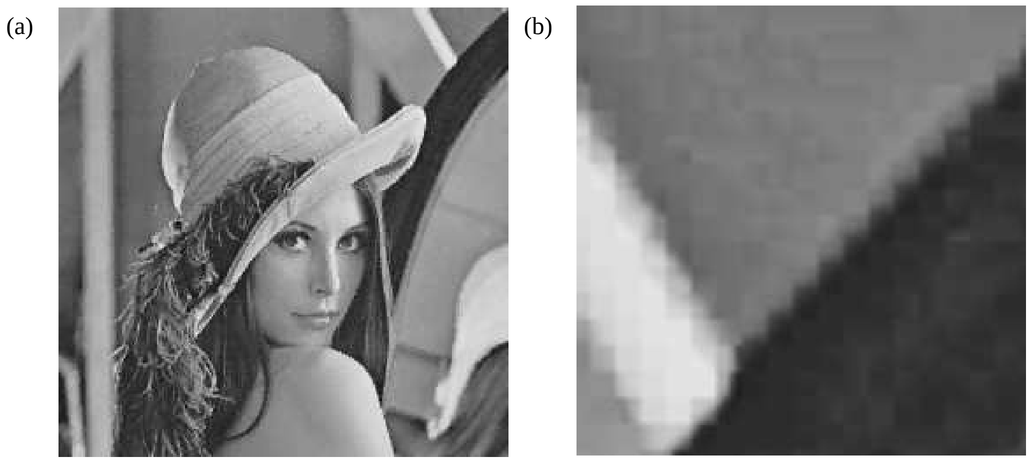


Figure 4.2: The image 'lena' (512×512) coded using a vector quantiser at 0.5625 bits/pixel and 4×4 blocks: (a) Entire coded image (PSNR=31.29 dB); (b) Close-up of an edge region, showing the 'staircase effect'.

be quantised [54]. And finally, compared with scalar quantiser-based schemes, such as transform coding, DPCM, etc., vector quantisers are not scalable in bitrate. That is, if the bitrate is changed, then the codebooks need to be re-trained.

To quantise larger image blocks and at higher bitrates, constrained vector quantisation schemes such as product code vector quantisers are a suitable alternative, with their lower complexity and storage requirements. Transform coding, which will be discussed in the next sub-section, can be considered a special case of a product code vector quantiser operating on transform coefficients.

4.4.2 Transform Coding

As we have discussed in Chapter 2, transform coding is a less complex alternative to vector quantisation and is effective on vector sources that have a high degree of correlation. Because correlation exists across both dimensions, transform coders can exploit this by forming vectors from blocks of $k \times k$ pixels. For typical image data with a correlation coefficient of 0.95, we have mentioned that the covariance function for each pixel, which is a measure of the spatial correlation[4], decreases significantly when more than 8 pixels away [20, 116]. As most of the spatial correlation exists within an 8×8 block, then not much coding gain can be achieved for larger block sizes. In addition, the computational

[4]The covariance function is equal to the correlation when the image has zero mean.

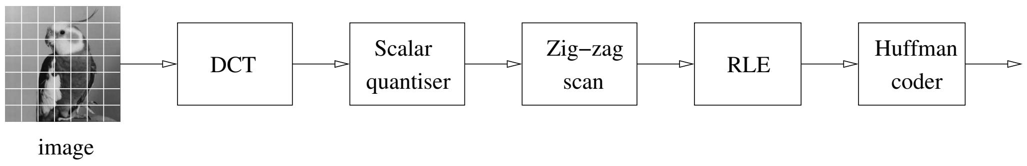


Figure 4.3: Block diagram of the lossy JPEG image coding standard

and memory requirements for performing the transformation for generally increase with block size.

The most popular transform coder for images is the JPEG standard [196], which is shown in Figure 4.3. Non-overlapping blocks of 8×8 pixels are taken from the image to be coded and these are transformed using a two-dimensional DCT. Assuming the image data to be highly correlated and generated by a Gauss-Markov process, the DCTs ability to compact energy into the lower coefficients is similar to that of the KLT [116]. Each transformed block is scalar quantised via division by a quantisation table of step sizes. The step sizes increase as we go higher in frequency and this is relative to the psychovisual importance of the coefficient. As a result of the scalar quantisation process, many high frequency coefficients are zeroed. In order to increase the efficiency of subsequent entropy coding, the quantised DCT coefficients are scanned in a zig-zag fashion, which order large values at the beginning of the data stream followed by a sequence of zeros. The sequence of zeros are truncated and represented by a special end-of-block (EOB) symbol. The DCT coefficients are then coded by runlength encoding (RLE) and Huffman encoding.

It is important to note that the lossy JPEG coder, with its use of entropy coding, is a variable-rate transform coder. This contrasts to the block quantiser, which is a fixed-rate transform coder. At high bitrates, the variable-rate coder has the advantage of better quantisation performance due to the combination of uniform scalar quantisers and entropy coding while the fixed-rate coder has the advantage of lower complexity requirements (ie. no need for buffering). Also, in the case of the JPEG coder, adjusting the bitrate is done via a quality factor, which is used to appropriately scale the fixed quantisation tables of step sizes. Consequently, it is more difficult to meet a certain target bitrate with variable-rate transform coders, like JPEG, than with fixed-rate transform coders, such as the block quantiser.

One of the disadvantages with transform coding is the introduction of block artifacts,

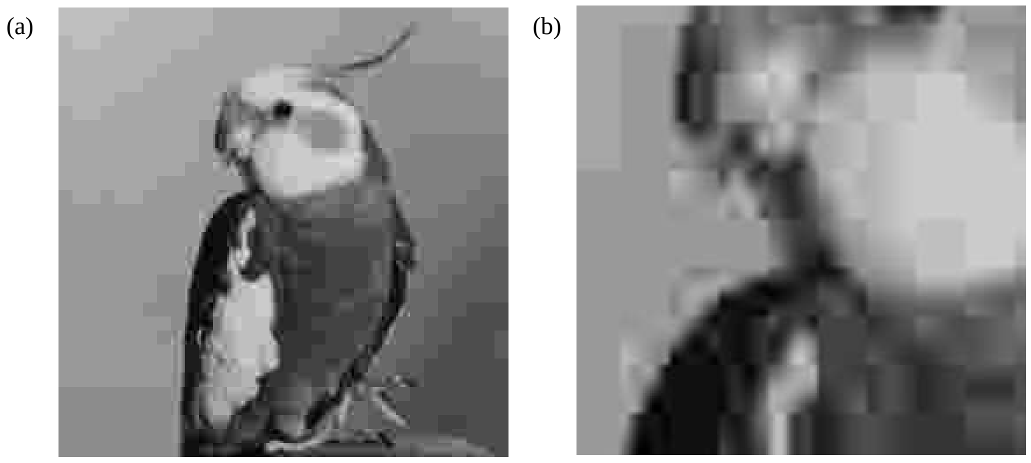


Figure 4.4: Showing the block artifacts of JPEG: (a) The image 'bird' coded at low bitrate (quality 8); (b) Close up of coded image, showing the 8×8 blocks.

as seen in Figure 4.4. This problem is caused by the discontinuities that result from the rectangular windowing of the image data. Various methods of reducing blocking artifacts include the use of overlapping blocks and low-pass filtering boundary pixels [145]. Disadvantages of these methods include an increase in bitrate and blurring, respectively [110]. Malvar and Staelin [110] investigated the *lapped orthogonal transform* (LOT). The idea of the LOT is to map blocks of n samples, to n basis functions which are l samples in length, such that $l > n$. Therefore, longer blocks of l samples are formed from the smaller blocks which overlap each other by $l-n$ samples [110]. This achieves the overlapping effect but since there are only n transform coefficients to quantise, there will be no increase in bitrate. Furthermore, the LOT basis functions decay toward zero at their boundaries, which leads to a reduction in block artifacts [110].

Transform coders which use a single transformation and quantisation scheme, assume that images are stationary, ie. the statistics throughout the image are uniform [13]. Unfortunately, this is not true as images contain edges and textures which have different spectral characteristics than those of smooth regions. Chen and Smith [28] investigated an adaptive scheme, where after each image block was transformed using the DCT, the resulting transformed blocks are classified based on the amount of activity, as measured by the energy level of the AC coefficients. That is, high activity blocks, such as edges or textures, will disperse energy among the high frequency (AC) coefficients, while smooth blocks will tend to concentrate energy in the DC coefficient. Classification is done using an equiprobable partition of the cumulative distribution function of the AC energies, with

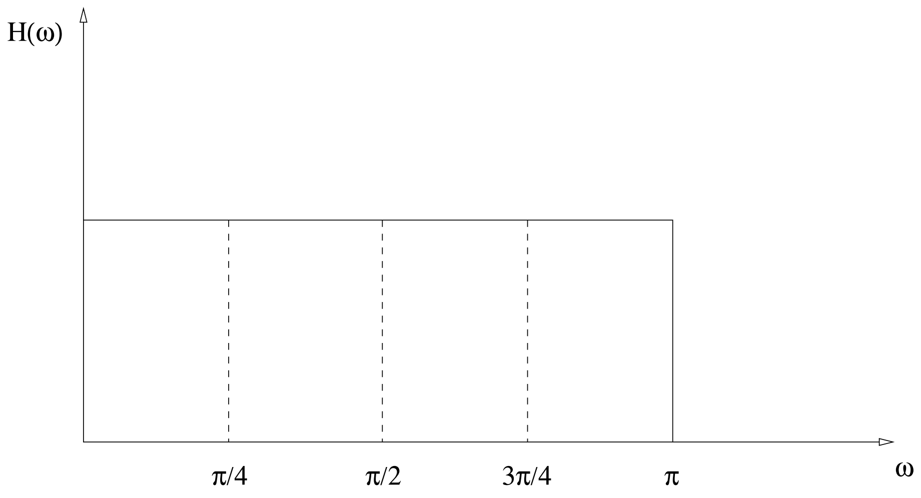


Figure 4.5: Uniform splitting of frequency spectrum into four subbands (after [136])

more bits being assigned to blocks of higher activity or AC coefficient energy and fewer bits given to blocks of lower activity [28]. Therefore, each block to be coded is classified based on activity, and then scalar quantised using the bit allocation of that class.

Adaptive transform schemes [9, 10, 11, 12, 13, 40, 43] discussed in Section 2.4, go further in addressing this problem of the non-stationarity of image data, by designing multiple transformations and quantisers to adapt to the local statistics of the image vector space.

4.4.3 Subband Coding

In subband coding, a set of low-pass and high-pass filters, known as a filterbank, are used to split the frequency spectrum into subbands, which are then quantised separately. Figure 4.5 shows the frequency band splitting performed by a uniform four-band subband coder. The idea is that different bitrates, or even different quantisation schemes, can be used for each subband, depending on the statistics of that band [200].

Referring to Figure 4.6, the input signal, $x(n)$, is passed through the analysis filterbank which consists of a low-pass filter, $h_1(n)$, and a high-pass filter, $h_2(n)$. These filters split the signal spectrum into two subbands. Assuming the input is n samples, this low-pass and high-pass filtering produces two signals, each with n samples, bringing the total number of samples to $2n$. The outputs of the analysis filterbanks are then downsampled by two (or by M for an M-band subband coder) which reduces the number of samples back to n. In effect, this decimates the sampling frequency of each subband and therefore demodulates

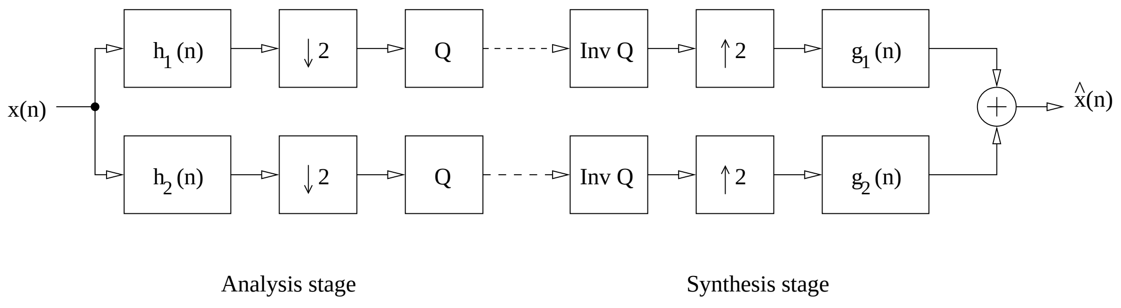


Figure 4.6: Block diagram of a two-band subband coder

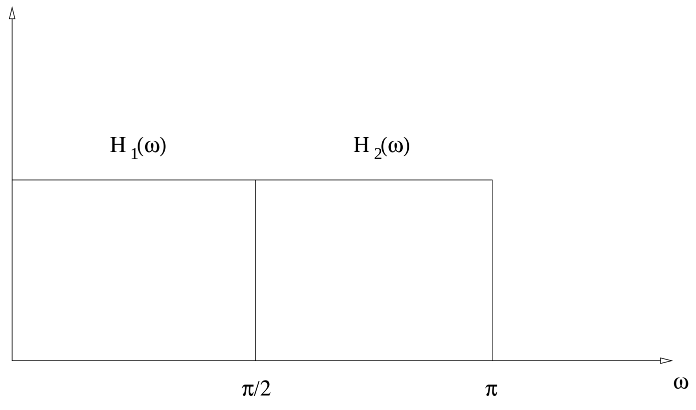


Figure 4.7: Brick-wall filter with ideal filter response

the subband to baseband [201]. After the downsampling, each subband is quantised with an appropriate number of bits, depending on the energy content or significance of that subband, and transmitted to the decoder. At the decoding or synthesis stage, the two bitstreams are dequantised, upsampled by two, and filtered with another set of subband filters. The outputs of the filters are summed together and an appropriate gain applied if necessary, depending on the properties of the filters. The upsampling is done by inserting a zero between each sample.

With no quantiser, the subband coder can be designed to be a lossless technique. That is, when data is coded and decoded using a subband coder, the output data should be the same as the input. This property is known as *perfect reconstruction* and shows that, in theory, the subband coding process is perfectly reversible. In practice, only near-perfect reconstruction can be achieved due to the limited accuracy of the filter coefficients.

The ideal subband filter with a vertical transition band, shown in Figure 4.7, is often called a 'brick-wall' filter and can only be approximated using realisable digital filters.

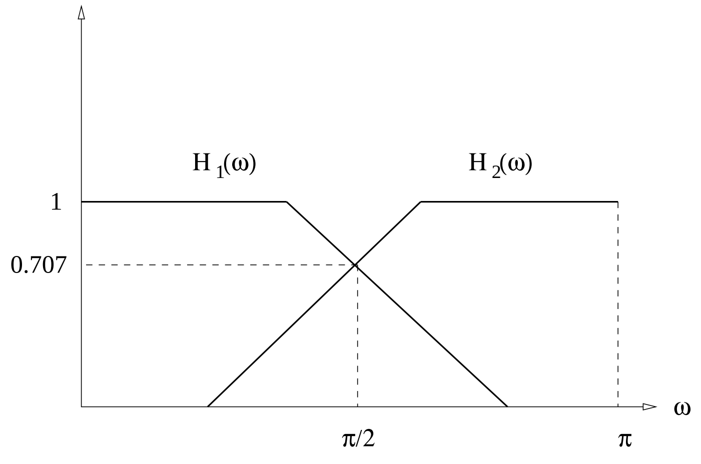


Figure 4.8: Frequency response of quadrature mirror filters (after [79])

Initial problems that occurred when using these included aliasing problems and ringing artifacts [142]. Esteban and Galand [46] introduced a new type of subband filter that avoided the aliasing problems and these were called *quadrature mirror filters* (QMFs), whose ideal frequency response is shown Figure 4.8. The high pass QMF is obtained from the low pass QMF, $h(n)$, by the formula, $(-1)^n h(n)$, therefore $H_1(z) = H(z)$ and $H_2(z) = H(-z)$.

Rather than attempting to approximate the ideal subband filters, QMFs were designed to cancel out the aliasing effect of the filter transition band [201]. The aliasing distortion can be shown to be equal to:

$$D = \frac{1}{2}\left[G_1(z)H(-z) + G_2(z)H(z)\right] X(-z) \tag{4.5}$$

To set the aliasing distortion to zero, the following condition must be satisfied by the QMFs:

$$G_1(z) = H(z) \tag{4.6}$$

$$G_2(z) = -H(-z) \tag{4.7}$$

The perfect reconstruction property can therefore be expressed as (assuming an even-length linear phase QMF) [136]:

$$|H(\omega)|^2 + |H(\omega - \pi)|^2 = 1, \quad \text{for all } \omega \tag{4.8}$$

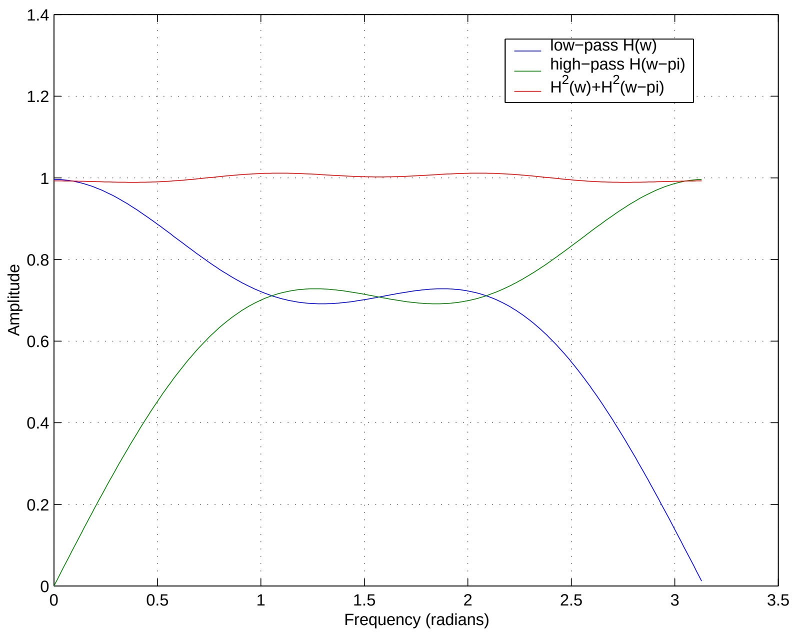


Figure 4.9: Frequency response of the Johnston QMF (8-taps)

Figure 4.9 and 4.10 shows this perfect reconstruction property for the 8-tap and 16-tap Johnston QMFs respectively.

A popular set of QMFs used in audio subband coding are the Johnston filters [79, 142]. Generally, the more taps there are in the QMF, the better the separation ability and therefore efficiency [152]. This is shown before in Figures 4.9 and 4.10 where the magnitude responses of an 8-tap and 16-tap Johnston filter are shown. However, too many taps can lead to problems with the accumulation of errors due to coarse quantisation. As opposed to human hearing, because the human visual system is more sensitive to phase changes [8], only linear phase (symmetric) QMFs are used in subband image coding.

Vetterli [192] extended the QMF idea to two or more dimensions, which was necessary for the subband coding of images. The idea was to use *separable filters*, where a two-dimensional QMF can be separated into two one dimensional filters [20]:

$$h(m,n) = h_1(m)h_2(n) \tag{4.9}$$

This simplifies the implementation of two dimensional subband coding where the row and columns can be filtered separately, similar in operation to the separable transform. Figure 4.11 shows one 'level' of a subband decomposition which produces four subbands, labelled

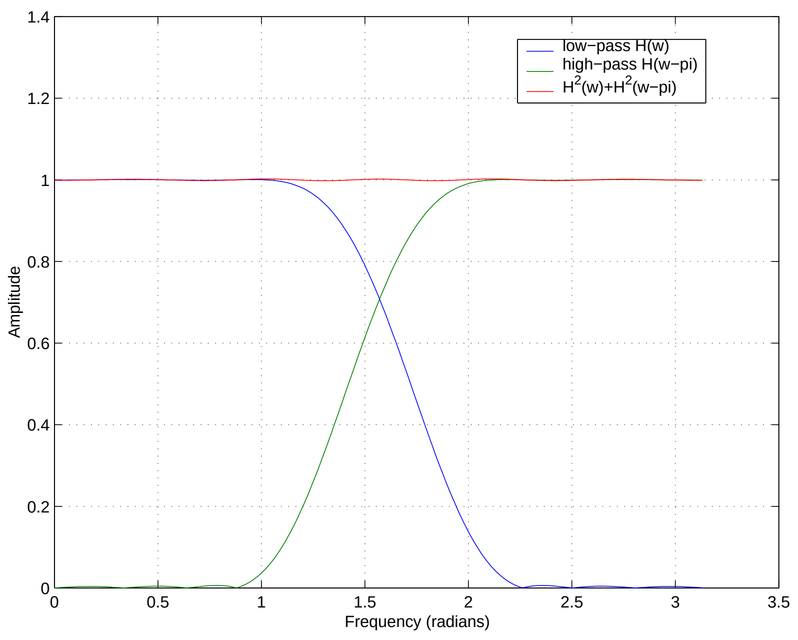


Figure 4.10: Frequency response of the Johnston QMF (16-taps)

LL (low-low), LH (low-high), HL (high-low), and HH (high-high).

Subband coding can be considered a more general case of transform coding [142]. In order to compare and contrast the two coding schemes, consider the case of transform coders (at least, Fourier-based ones). The frequency spectrum is divided uniformly into spectral lines or bins and these are processed (ie. quantised) in the frequency domain. Whereas in subband coding, the frequency spectrum may be split uniformly or non-uniformly into bands of frequencies. The outputs of the filters are time domain signals, hence the quantisation is performed in the time domain.

There are many different configurations of subband decompositions. The two most common are shown in Figure 4.12. The octave subband decomposition, also known as dyadic subband decomposition, is typically used for the wavelet transform, which will be discussed later.

Subband coding was first applied to speech coding by Crochiere *et al.* [32]. Since then, it has made inroads in speech coding standards such as the ITU-T G.722 (at 48/56/64 kbps) and G.726, and in audio coding standards such as the MPEG 1 layer 1/2/3 coding [152]. Its direct application to image coding was first investigated by Woods and O'Neil [200]. In their study, the subband image coder achieved higher SNRs than the adaptive

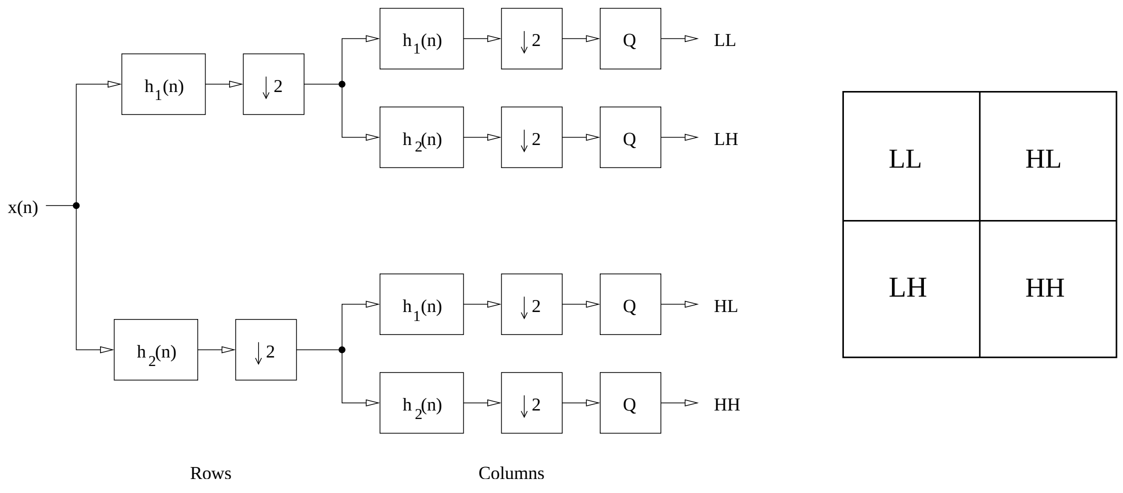


Figure 4.11: Two dimensional subband coding using separable filters

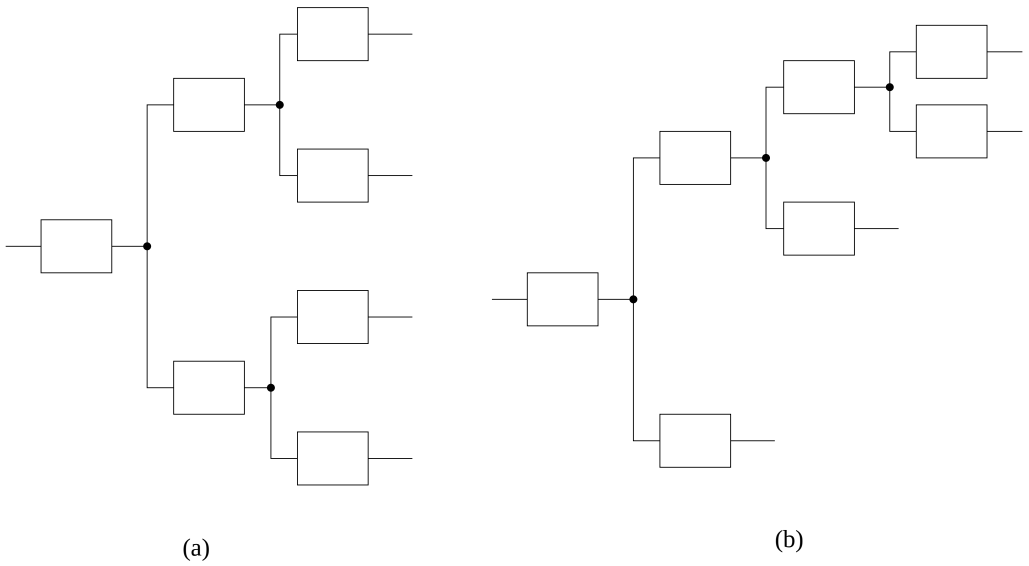


Figure 4.12: Subband decomposition configurations: (a) Uniform subband decomposition; (b) Octave or dyadic subband decomposition

DCT transform coder of Chen and Smith [28], at bitrates below 1.5 bits/pixel. An adaptive subband coding scheme was also presented, where subbands were classified as either busy, non-busy, or quiet, and this coder outperformed all other quantisation schemes considered in the study, including the vector quantiser and differential vector quantiser [200].

The most significant advantage of subband/wavelet image coding over transform coding is the latter operates on blocks of the image while the former operates on the entire image. Consequently, subband/wavelet image coders do not suffer from the block artifacts that are commonly encountered in low bitrate transform coding. Typical artifacts in subband image coding include 'ringing' occurring around the edges [142], which are mostly due to coarse quantisation of the high frequency subbands which contain the edge information.

4.4.4 Wavelet-Based Image Coding

Wavelets and their Properties

With the short-time Fourier transform (STFT), a signal is time windowed[5] into segments and the frequency spectra is determined for each segment via a discrete Fourier transform [112]. This results in a two-dimensional data representation known as the time-frequency representation[6]. While this representation is adequate for signal analysis, it has inherent disadvantages. To obtain high resolution in the frequency domain (narrowband), the time window has to be made wider, in order to capture longer data lags, but this results in low resolution in the time domain. On the other hand, to obtain high resolution in the time domain, a shorter time window is required but this in turn degrades the frequency resolution to wideband [158]. This trade-off between time and frequency is intrinsic and was first discovered by Gabor [50], popularly known as the *Gabor uncertainty principle*:

$$\Delta\omega\Delta t \geq \frac{1}{2} \tag{4.10}$$

Since the basis functions of the STFT are of fixed size in time and frequency, the time-frequency resolution is fixed as well [58]. Relating this to DCT-based transform image coding, because the basis functions (cosines) have fixed spatial area and frequency bandwidth, smooth and edge regions are represented by transform coefficients at the same

[5]For the case of images, the signals span space rather than time.

[6]The time-frequency diagram was first introduced by Gabor [50], where the time-frequency window is a special Gaussian function (Gabor function) and the transform later called a *Gabor transform*

spatial-frequency resolution. Consequently, edge information tends to result in energy being dispersed to many transform coefficients. A higher bitrate is therefore required to reconstruct the edge information accurately [158].

Wavelets allow a signal to be analysed at different scales or support widths. The wavelet basis set consists of functions with different support widths to trade-off time and frequency resolution [68]. Wavelets with wide support examine large regions of the signal and hence are suitable for low frequency content or analysing ‘trends’ while those with short support examine small regions of the signal and hence are suitable for high frequency content or analysing ‘anomalies’ [68, 158]. Because wavelets often have compact support, the decomposition of a signal at a time instant consists of only those wavelets that are located in (or, translated to) that region. Other wavelets that are outside the vicinity do not contribute to the reconstruction. This contrasts to the Fourier transform, whose complex exponential basis functions have global support and exist for all time, or in the case of the STFT, exist throughout the analysis window [38]. The property of compact support is of benefit to the image coding problem, as wavelets with short support have excellent spatial resolution and this allows edge information to be represented by *sparser* wavelet coefficients. Therefore, encoding edge information will not require as high a bitrate as one would need in a DCT-based transform coder.

Wavelets are a set of basis functions that are generated through dilations and translations of a single function, $\psi(t)$, called the *mother wavelet* [7].

$$\psi_{a,b}(t) = |a|^{-\frac{1}{2}}\,\psi\left(\frac{t-b}{a}\right) \tag{4.11}$$

where a specifies the dilation factor (scale) and b the amount of translation. High frequency resolution (or, small scale) wavelets which have a narrow width, correspond to $a < 1$ while low frequency (or, large scale) wavelets which have a wider width, correspond to $a > 1$ [7]. The mother wavelet also satisfies the following property

$$\int_{-\infty}^{\infty} \psi(x)dx = 0$$

which implies the function is oscillatory [7].

The continuous wavelet transform (CWT), $c(a, b)$, of a signal, $f(t)$, is defined as:

$$c(a, b) = \int f(t)\psi^{*}_{a,b}(t)dt \tag{4.12}$$

where a and b are the scale and translation of the wavelet, respectively. As opposed to the time-frequency representation of the STFT, the CWT produces a time-scale representation [191].

It should be noted that the scale, a, and translation, b, in (4.12) are continuous variables. In order to limit the number of wavelet basis functions, the scale and translations are discretised, with $c(a, b)$ becoming a set of wavelet coefficients. The resulting wavelet is termed a *discrete wavelet* [191]. In order for the wavelet transform to be non-redundant and easily computed, the basis functions need to be orthogonal. A dyadic discretisation of the scale and translation, $a = 2^m$ and $b = 2^m n$, is popular and results in the following wavelet functions [7]:

$$\psi_{m,n}(t) = 2^{-\frac{m}{2}}\psi\left(2^{-m}t - n\right) \tag{4.13}$$

which form an orthogonal basis in $L^2(R)$[7]. A wavelet series decomposition can then be performed on a continuous signal, $f(t)$:

$$f(t) = \sum_{m,n} c(m, n)\psi_{m,n}(t) \tag{4.14}$$

where:

$$c(m, n) = \langle \psi_{m,n}(t), f(t) \rangle \tag{4.15}$$

Multiresolution Analysis and the Discrete Wavelet Transform

Burt and Adelson [25] proposed the Laplacian pyramidal coder, shown in Figure 4.13, which introduced the concept of approximating an image at different resolutions. An image is firstly low-pass filtered to reduce the bandwidth and downsampled by two to produce a lower resolution approximation of the image. This lower resolution image is upsampled and passed through an interpolation filter to give a predicted image, which is subtracted from the original to produce a residual image. The pixel values of the residual image are sparser (ie. have lower entropy) and therefore, are more amenable to coding

[7]This is the space of square integrable functions.

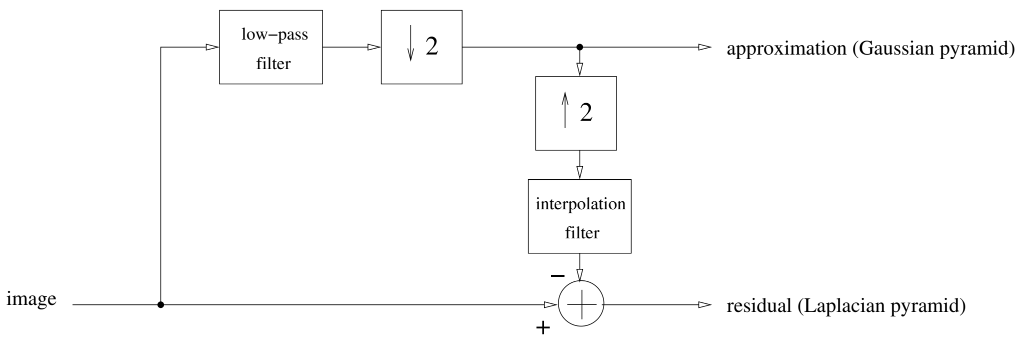

low−pass filter
↓2
approximation (Gaussian pyramid)
↑2
interpolation filter
−
+
image
residual (Laplacian pyramid)

Figure 4.13: Laplacian pyramidal image coder

[25]. The above process can be recursively applied on the approximated image and as a result, smaller and smaller approximated and residual images are produced, forming the Gaussian and Laplacian pyramid, respectively. The disadvantage of this method is that residual image is the same size as the original, and together with the smaller approximated image, there is actually more data produced as a result of the process [34].

Mallat [109] introduced the concept of multiresolution analysis (MRA) using wavelets, which is similar to the Laplacian pyramid idea. That is, a lower resolution approximation of an image is found, effectively via a low-pass and downsampling operation. However, the loss of information (residual), as a result of going to the coarser approximation, is represented more efficiently than in the Laplacian pyramidal coder and does not lead to an increase in data. This is the basic operation of the discrete wavelet transform (DWT).

In the DWT, two sets of basis functions are required: the wavelet function, $\psi_{m,n}(t)$, and its associated scaling function, $\phi_{m,n}(t)$. Like the wavelet function, the scaling function can be dilated and translated to form an orthogonal basis set:

$$\phi_{m,n}(t) = 2^{-\frac{m}{2}}\phi(2^{-m}t - n) \tag{4.16}$$

The scaling basis functions are used to approximate the image at different scales. In other words, they span successive approximation spaces [7]:

$$\ldots \subset V_1 \subset V_0 \subset V_{-1} \subset \ldots \tag{4.17}$$

where at resolution 2^m:

$$V_m = \text{span}\,\{\phi_{m,n}(t) : m, n \in \mathcal{I}\} \tag{4.18}$$

The wavelets, $\psi_{m,n}(t)$, span an orthogonal complementary space:

$$W_m = \operatorname{span}\{\psi_{m,n}(t) : m, n \in \mathcal{I}\} \tag{4.19}$$

The two function spaces V_m and W_m constitute the resolution of 2^m and together, complement to form a function space of finer resolution of 2^{m-1} [7]:

$$V_{m-1} = V_m \oplus W_m \tag{4.20}$$

One interpretation of this is that when a signal needs to be represented in a coarser resolution space, V_m, there will be some information loss, which is represented by the complementary wavelet space, W_m [7, 112]. The coarser resolution of the signal is decomposed into scaling basis functions, $\phi_{m,n}(t)$, while the information loss is decomposed into wavelet basis functions, $\psi_{m,n}(t)$. An alternative way of interpreting the DWT is viewing the various dilated wavelet basis functions as a set of constant Q-factor band-pass filters[8] centred at octave frequencies. The scaling functions, which are essentially low-pass filters, cover the rest (lower part) of the spectrum [191].

Implementing the Discrete Wavelet Transform

Mallat [109] showed that the DWT can be implemented using a subband coding algorithm. This simplifies the implementation as appropriate filterbanks can be derived from the wavelets and the subband coding algorithm can be applied.

Equation (4.20) and its interpretation lends itself to the idea of subband filtering. The function space, V_m, represents the low-frequency information of the signal, L_m. While the function space, W_m, represents the high-frequency information, H_m, of the signal that is removed. Or quite simply:

$$\underbrace{V_{m-1} = V_m \oplus W_m}_{\text{wavelet interpretation}} \quad \Leftrightarrow \quad \underbrace{L_{m-1} = L_m + H_m}_{\text{subband interpretation}} \tag{4.21}$$

The subband algorithm fits in well with finding the discrete wavelet transform. When $a(m,n)$ and $c(m,n)$ are the wavelet coefficients in the V_m (or low subband) and W_m (or

[8]The Q factor of a band-pass filter is defined as the product of its centre frequency with its bandwidth.

high subband), respectively, decomposition of the DWT can be expressed as [7]:

$$c(m,n) = \sum_k h_2(2n-k)a(m-1,k) \tag{4.22}$$

$$a(m,n) = \sum_k h_1(2n-k)a(m-1,k) \tag{4.23}$$

$$h_2(n) = (-1)^n h_1(1-n) \tag{4.24}$$

$$h_1(n) = \sqrt{2}\int \phi(x-n)\phi(2x)dx \tag{4.25}$$

where h_1 and h_2 are low-pass and high-pass filters respectively. The original image is set to be finest resolution, $m = 0$.

With regards to performance, there are some characteristics of the DWT that make it suitable for image coding. Wavelets tend to produce smooth, regular, and short filters which are appropriate for images. Most areas in an image are smooth and the scaling functions approximate them well [7]. Short DWT filters reduce the accumulation of quantisation errors in high frequency subbands. Subband QMFs used in audio are often too long (eg. 32 taps or more) and these are not suitable for image coding [193]. Also, the step response of short filters tends to have less oscillations and these have a major influence on the coding of edges [192].

It must be noted that the wavelet and scaling functions are assumed to be orthonormal. This results in only one filter that is needed to be specified and both analysis and synthesis filter banks can be derived from it. However, there are no non-trivial orthonormal linear phase FIR wavelet filters with the perfect reconstruction property except for the Haar wavelets [7]. Rather than orthonormality, a less restricting condition can be used called *biorthogonality.* Biorthogonal wavelet filters define an extra set of basis functions, bringing the total to four: $\phi(t), \tilde{\phi}(t), \psi(t), \tilde{\psi}(t)$. The 'tilde' functions become the synthesis basis functions while the other two form the analysis basis functions. For perfect reconstruction, given the biorthogonal low-pass filter pairs, h_1 and g_1 [7]:

$$h_2(n) = (-1)^n g_1(1-n) \tag{4.26}$$

$$g_2(n) = (-1)^n h_1(1-n) \tag{4.27}$$

Table 4.1 shows the filter coefficients for the biorthogonal 9/7-tap wavelet filter given

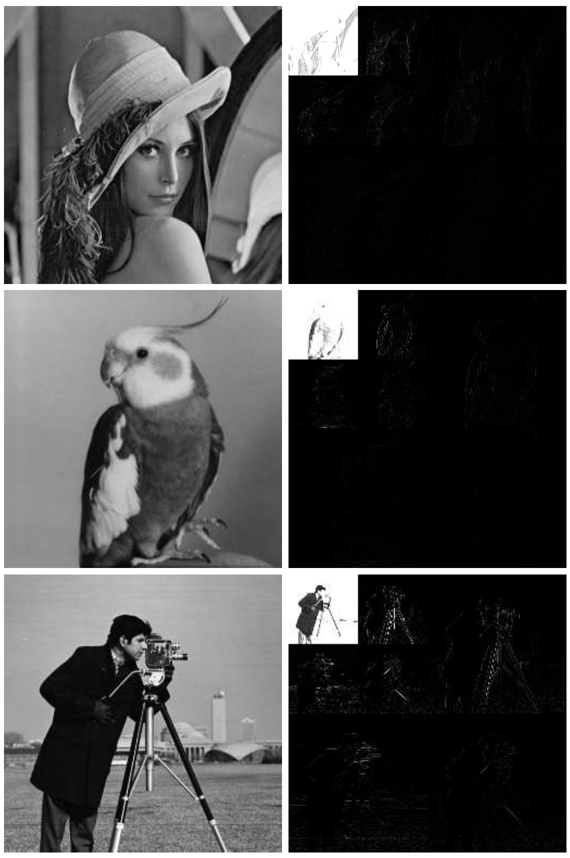


Figure 4.14: Discrete wavelet transform of greyscale test images: 'lena', 'bird', 'cameraman' (using 9/7-tap biorthogonal wavelet filters)

in [7]. Figure 4.14 shows the original greyscale test images 'lena', 'bird', and 'camera man' as well as their wavelet transform (using the 9/7-tap wavelet filter). It can be observed that most of the energy of the image has been compacted into the LL subband, where the low frequency information or 'trends' of the image have been captured by the scaling functions. The coefficients in the other subbands are very sparse except for some edge information, which have been captured by the local wavelet functions. In fact, wavelets act as 'singularity detectors' and it is because of the sparseness of the wavelet coefficients that makes the DWT a more efficient transform than the DCT on edge data [194]. Also, since this wavelet is biorthogonal rather than orthonormal, energy has not been conserved and the LL subband contains more energy.

Table 4.1: Coefficients of the spline variant 9/7-tap wavelet filter (after [7])

n	1	±1	±2	±3	±4
$h_1(n)\sqrt{2}$	0.602 949	0.266 864	-0.078 223	-0.016 864	0.026 749
$g_1(n)\sqrt{2}$	0.557 543	0.295 636	-0.028 772	-0.045 636	0

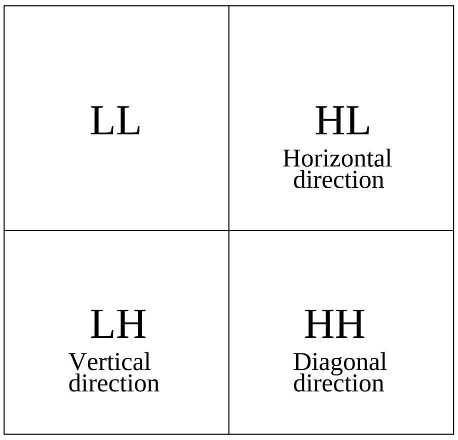


Figure 4.15: Image subbands and their respective directions (after [7])

Vector Quantisation of Wavelet Coefficients

Early subband and wavelet coders utilised scalar quantisation-based schemes. The first application of vector quantisers for wavelet image coding was investigated by Antonini *et al.* [6]. They noted that global vector quantisers, which were designed to quantise all subbands, tended to smooth edge information which degraded the final reconstruction [6]. As shown in Figure 4.15, each of the high frequency wavelet subimages capture edges of a certain direction. Therefore, they adopted a multiresolution codebook approach, where individual codebooks were designed for each resolution and each directional subimage using the LBG algorithm. This resulted in a better preservation of edge information. The lowest LL subband was coded using scalar quantisers. A bit allocation formula, based on Lagrangian optimisation and the high resolution rate-distortion performance of vector quantisers, was derived to distribute bits to each of the subimage vector quantisers. A PSNR of 31.5 dB was achieved on the image 'lena' (256×256) at 0.69 bits/pixel [6]. It was report in [7] that, when using a weighted MSE distance measure, the subjective quality of the reconstructed image ('lena' of dimensions 512×512) from the wavelet coder (PSNR=30.85 dB at 0.37 bits/pixel) was better than that of vector quantiser-based image

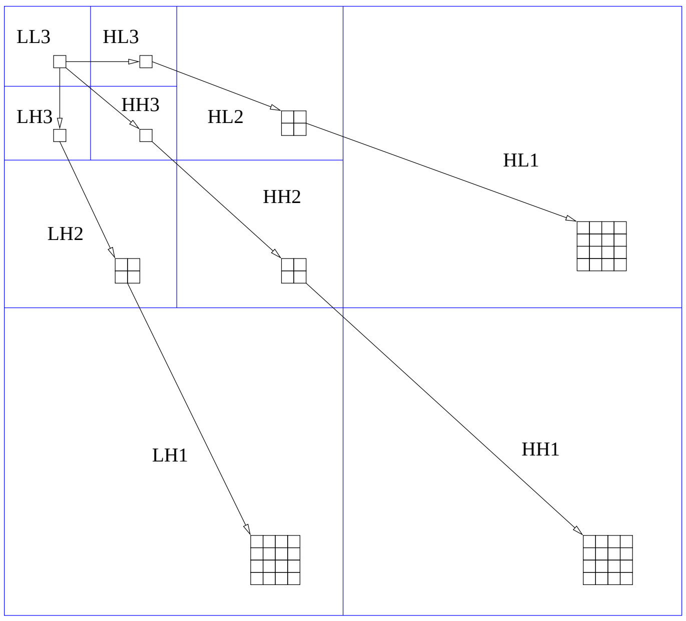


Figure 4.16: Zerotrees used in the embedded zerotree wavelet coder (after [158])

coders which, in comparison, achieved a higher PSNR. The wavelet coded image also did not suffer from the block artifacts of vector quantisation.

The Embedded Zerotree Wavelet Coder

Shapiro [158] introduced an advanced method of coding wavelet coefficients that utilised correlation between subimages. While this algorithm was designed without the use of rate-distortion optimising methods, Ortega and Ramchandran [119] showed that the embedded zerotree wavelet (EZW) models the subbands more effectively than previous wavelet coding techniques based on greedy bit-allocation algorithms, and hence achieves an operational rate-distortion function that outperforms all previous methods. Shapiro stated that the significance map coding rate achieved in EZW is even lower than the first-order entropy [158, 159]. Indeed EZW and later on an extended version, SPIHT, still remain one of the best performing image compression techniques in the literature [118].

Shapiro made a few observations of the wavelet transform of an image. Firstly he noted that areas of insignificance[9] or areas where the coefficient values are low, appear in the same spatial location in other subbands of differing orientation at the same level as well

[9]In the context of EZW coding, a coefficient is considered significant if its magnitude is greater than a threshold.

as in finer levels [158]. In other words, when there is an 'insignificant' coefficient in one subband, there is a very good chance that its *descendants* in finer levels are insignificant as well [158]. Secondly, he noted that it is easier to predict insignificance across subbands rather than significance [158]. In this scheme, a coefficient is termed the *parent* while the set of four coefficients in the next finer level are the *children* (See Figure 4.16). Coefficients at finer levels form the *descendants* of coefficients at coarser levels. When a parent is not a descendant of any previously found zerotree root, is insignificant, and all of its descendants are known to be insignificant with respect to a threshold, a *zerotree* is formed and the insignificant parent is termed the *zerotree root* or *ZTR*. An insignificant coefficient whose descendants are not all insignificant is called an *isolated zero* or *IZ*. Significant coefficients are classified as either *positive significant* (*POS*) or *negative significant* (*NEG*) [158].

A *significance map* shows the classification of each coefficient, whether it is significant or insignificant. Zerotrees allow these significance maps to be represented efficiently by utilising the correlation of insignificant coefficients in different subbands. For a certain threshold, the wavelet coefficients are scanned and classified into the categories of *POS, NEG, ZTR, IZ* to produce the significance map [158]. Like the end-of-block (EOB) symbol in JPEG for representing a runlength of zeros, zerotrees allow the EZW coder to efficiently code exponentially large regions of insignificant wavelet coefficients [158].

One distinct feature of the EZW coder is that it produces an embedded code [158]. That is, a low resolution version of the image is embedded at the start of the bitstream and further bits progressively improve the reconstructed image. This allows EZW coding to stop when the desired bitrate has been reached. Also, the receiver can stop receiving more bits from the encoder and still be able to reconstruct a decent version of the image from the data already received. Using *successive approximation quantisation* (SAQ), the threshold, which determines whether a wavelet coefficient is significant, is initially set to half the magnitude of the largest coefficient. The significance map is calculated based on this threshold and transmitted in an efficient way. The threshold is then halved and the process continues. Basically, as the threshold is successively decreased, more and more wavelet coefficients are deemed significant. Also, the significance map is coded using a scalar quantiser whose levels are successively refined by both the encoder and decoder during each stage. For more details on SAQ and EZW coding in general, the reader is referred to [158].

EZW coding highlights an interesting contrast between the type of approximation used in optimal wavelet transform coding and block-based transform coding. In block-based transform coding, where the KLT is applied, we have shown previously that linear approximation (truncating transform coefficients in ascending order) results in the least MSE distortion and generally, reconstructed image quality. However, in optimal wavelet transform coding, the best approximation is based on keeping those coefficients with the largest magnitudes rather than based on their ordering. This is termed as *non-linear approximation* and it can be shown that the incurred MSE distortion as a result of this is lower than that of linear approximation in Fourier-based transform methods [194].

The Set Partitioning in Hierarchical Trees Coder

Amir and Said [150] extended the ideas of EZW to produce an embedded algorithm which was more superior in terms of bitrate and complexity. EZW relies on using an adaptive arithmetic coder for entropy encoding the zerotree and subordinate information [158]. The set partitioning in hierarchical trees (SPIHT) algorithm surpasses the EZW coder when using an adaptive arithmetic coder. But surprisingly, the algorithm without an entropy coder, which is termed as binary-uncoded SPIHT [150], outperforms the more complex EZW coder by about 0.5 dB for the same bitrate.

The SPIHT coder is based on the concept of *subset partitioning* to encode significance. The algorithm also differs with EZW coding in that it does not explicitly transmit the ordering information. The EZW coder transmits this ordering information via significance maps which need to be coded using zerotree encoding in order to allow the decoder to know where significant coefficients occur. SPIHT coding removes the need to transmit a significance map by ensuring both the encoder and decoder use the same set partitioning rules and that they both follow the same execution path determined via decisions sent by the encoder [150]. That is, both the encoder and decoder synchronise their operations via decisions.

In much the same way as defining zerotrees in EZW coding, SPIHT coding defines a tree structure that spans the subbands of the wavelet transform called a *spatial orientation tree* (SOT). SOTs are different to zerotrees in that they are not formed on the basis of whether coefficients are insignificant or not. Each *node* of the tree is a pixel and is identified by its co-ordinate. The pixels in the same SOT are defined as the *descendants* of the *parent node*

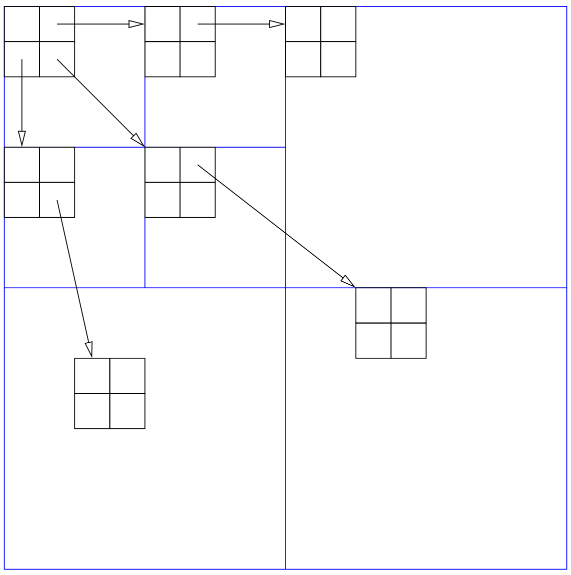


Figure 4.17: Spatial orientation trees used in SPIHT coding (after [150])

The *offspring* are the *direct* descendants of the parent node. The nature of the tree is that all nodes either have no offspring (hence they are leaves) or have exactly four offspring which is represented by a 2×2 block. *Tree roots* are defined as those pixels in the coarsest level (LL subband) and each have their own offspring except for the one in the top left corner [150].

SOTs are used to partition pixels into different sets. Like zerotrees, they exploit spatial self-similarity in the different levels of the wavelet transform. These different sets are defined as follows [150]:

- $O(i, j)$ is the set of co-ordinates of all offspring of node (i, j);
- $D(i, j)$ is the set of co-ordinates of all descendants of the node (i, j);
- H is the set of co-ordinates of all spatial orientation tree roots or nodes in the finest wavelet subband; and
- $L(i, j) = D(i, j) - O(i, j)$ which is the set of co-ordinates of all descendants from the node (i, j) but not including direct offspring from this node.

Each SOT can be partitioned into different partition sets. The procedures for doing this partitioning, which are called the set partitioning rules, are [150]:

1. Initial partition is formed from the sets $\{(i, j)\}$ and $D(i, j)$ for all $(i, j) \in H$;

2. if the set $D(i, j)$ is significant, it is further partitioned into five sets: $L(i, j)$ and four single-element sets $\{(k, l)\}$ where each $(k, l) \in O(i, j)$;
3. if $L(i, j)$ is significant then it is partitioned into the four sets $D(k, l)$ with $(k, l) \in O(i, j)$.

These rules are explained below:

- Rule 1 suggests forming the initial partition from all SOT roots (pixels in the LL subband) and their descendants;
- Rule 2 suggests that if the descendant set of root (i, j) are significant, then this partition is split into five different sets: a set consisting of the descendants of root (i, j) but not including the offspring; and four sets that each contain the pixels of the offspring; and
- Rule 3 suggests that if the set of descendants (not including offspring) from root (i, j) are significant, then it should split into four different sets which are descendant sets of each of the offspring of root (i, j).

The basic idea is that all the pixels of the wavelet transform are divided into partitioning subsets, T_m, through the use of spatial orientation trees. Then a *decision* is made of whether a set is significant or not. The concept of bitplane encoding is also used in SPIHT coding and is done via the use of thresholds that are determined by n and the intervals are $2^n \leq |c_{i,j}| < 2^{n+1}$ where n is decremented after each pass [150]. The significance S, at the precision n, of a partition set T can be expressed as:

$$S_n(T) = \begin{cases} 1, & \max_{(i,j)\in T_m} \{|c_{i,j}|\} \geq 2^n \\ 0, & \text{otherwise} \end{cases} \tag{4.28}$$

These decisions are sent to the decoder in order to allow it and the encoder to synchronise their partition sets. When the encoder reports that a particular partition set T_m is insignificant, the decoder knows that all wavelet coefficients belonging to that set are insignificant. When instead the encoder reports that a particular partition set T_m is significant, then the encoder will proceed to partition this into new subsets $T_{m,l}$ based on the set partitioning rules described in the previous section. Since these rules are known to the decoder as well, it can also partition the set in the same way. Therefore, the execution path is synchronised between the encoder and decoder via these significance tests. The set division process stops when all single co-ordinate significant subsets have been tested [150]. For a more detailed description of SPIHT coding, the reader is referred to [150].

4.5 Non-Expansive Filtering for the Subband Decomposition of Images

In contrast to speech and audio coding, any expansion of image data as a result of the linear convolution of a row or column with an FIR filter, is undesirable and should be avoided. Also, both edges of a finite length signal need to be handled in a special way as the impulse response of the analysis and synthesis filters need to overlap correctly for perfect reconstruction [142]. The simplest method is to add redundancy to both ends of the signal, via zero or constant value padding. The former presents a problem since there will generally be a sharp discontinuity, if the edge value has a large magnitude. This causes a large amount of energy to appear in the high frequency subband. Because the high frequency subbands are generally coarsely quantised, ringing artifacts will appear on the edges of the reconstructed image [142]. Also, both methods of padding will lead to an expansion of data [142].

Smith and Eddins [164] gave two methods of adding redundancy without any expansion of data: *circular extension* and *symmetric extension*. In the circular extension method, the signal is wrapped around itself to form a periodic signal of period, N. In the symmetric extension method, both edges are reflected to form a periodic signal of period, $2N$. The idea is that when we filter a periodic signal, where there are only N unique samples, the output signal will also be periodic and have at least N unique samples as well, which circumvents the expansion issue.

Kiya *et al.* [84] investigated a larger class of extension methods, which included symmetric and anti-symmetric filters with an even and odd number of taps. They noted that the methods outlined by Smith and Eddins [164] assumed the filters to have an even number of taps only. Martucci [106] gave a detailed treatment of filter causality and signal extension methods (both circular and symmetric) as well as their effects on perfect reconstruction in subband image coding. A general procedure was also presented that guarantees both non-expansive filtering and perfect reconstruction for all possible cases.

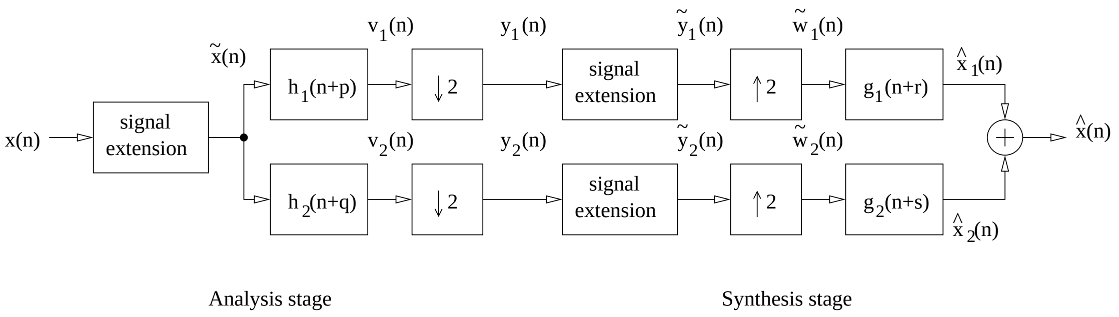


Figure 4.18: Subband coding with non-causal filtering and signal extension (after [106])

This procedure, as given in [106], is presented below.

4.5.1 Non-Causal Filtering with Minimal Delay

A causal filter is one in which the output response is calculated based on the present and past input samples only. That is, the impulse response of a causal filter, $h(n)$, is zero for $n < 0$. Generally, the processing of real-time signals requires the use of causal filters only since we do not have access to future input values. Assuming we are applying a causal, linear phase filter of length, M, on a signal of finite length N, the output signal will have a length of $N + M - 1$ samples and a delay of $\frac{M-1}{2}$. This is mostly due to the linear convolution of a finite length signal.

Because images are finite and span space rather than time, the causality constraint of the filters can be relaxed. Non-causal filters are known to add minimal delay to the output, hence they are suitable for use in image subband coding [106]. A non-causal filter is obtained by shifting the impulse response of the causal filter by $\frac{M-1}{2}$ samples. Shifting a linear phase filter, with an odd number of taps, by this amount results in no delay (Figure 4.19(a)). For the case of a filter with an even number of taps and depending on the shift, either a half sample delay or advance is possible (Figure 4.19(b)).

In subband coding, because the downsampler and upsampler are shift-invariant, then extra care needs to be made with regards to the delays caused by the filtering, in order to ensure perfect reconstruction [106]. Figure 4.18 shows the subband coder with non-causal filters, which are formed by shifting their causal versions by appropriate values, denoted as p, q, r, s. The signal extension blocks are ignored for now [106].

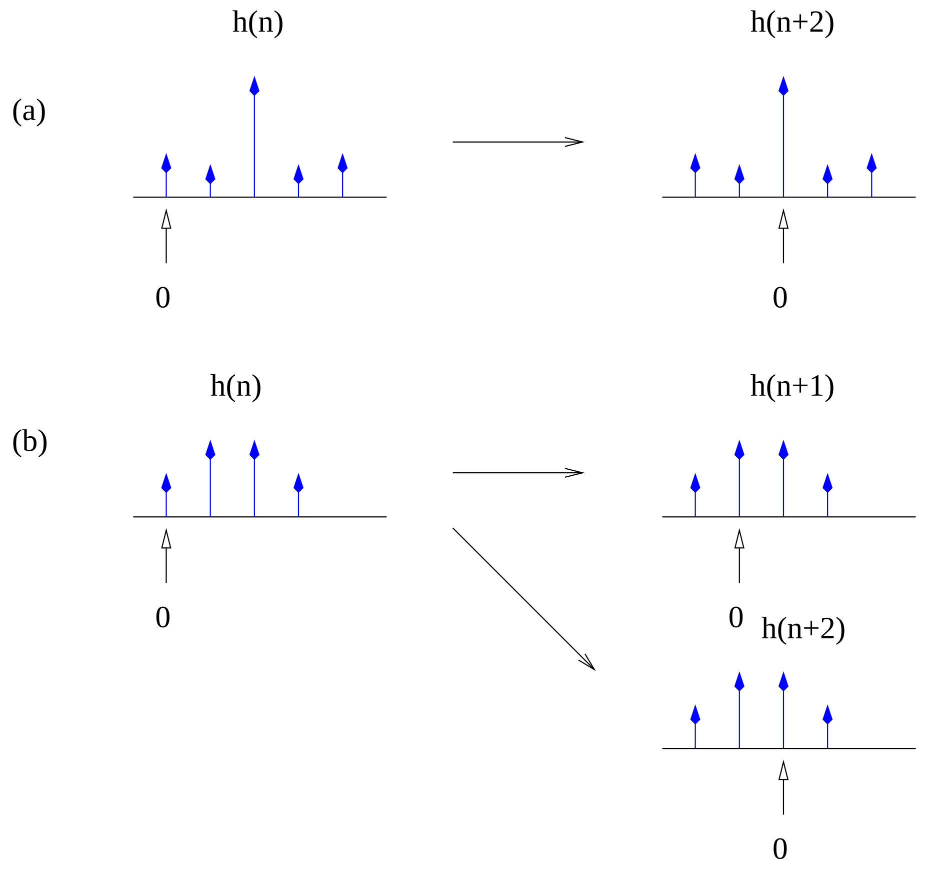


Figure 4.19: Impulse responses of causal linear phase filters and their non-causal versions with minimal delay (after [106]): (a) Causal filter, $h(n)$, with an odd number of taps and its non-causal version, $h(n+2)$, which imparts no delay; (b) Causal filter, $h(n)$, with an even number of taps and its non-causal versions, $h(n+1)$, which imparts a half sample delay and $h(n+2)$, which imparts a half sample advance.

In the Fourier domain, the output in Figure 4.18 can be expressed as [106]:

$$\hat{X}(e^{j\omega}) = T(e^{j\omega})X(e^{j\omega}) + A(e^{j\omega})X(e^{j(\omega-\pi)}) \tag{4.29}$$

where the transmission factor, $T(e^{j\omega})$, is given by [106]:

$$T(e^{j\omega}) = \frac{1}{2}\left[H_1(e^{j\omega})G_1(e^{j\omega})e^{j\omega(p+r)} + H_2(e^{j\omega})G_2(e^{j\omega})e^{j\omega(q+s)}\right] \tag{4.30}$$

and the aliasing factor, $A(e^{j\omega})$, is given by [106]:

$$A(e^{j\omega}) = \frac{1}{2}\left[H_1(e^{j(\omega-\pi)})G_1(e^{j\omega})e^{j\omega(p+r)}e^{-j\pi p} + H_2(e^{j(\omega-\pi)})G_2(e^{j\omega})e^{j\omega(q+s)}e^{-j\omega q}\right] \tag{4.31}$$

Two solutions exist for perfect reconstruction, where the aliasing factor, $A(e^{j\omega}) = 0$, and the transmission factor, $T(e^{j\omega}) = 1$. With the first solution [106]:

$$\begin{aligned} G_1(e^{j\omega}) &= H_2(e^{j(\omega-\pi)}) && (4.32)\\ G_2(e^{j\omega}) &= -H_1(e^{j(\omega-\pi)}) && (4.33) \end{aligned}$$

we have the following constraints on the filter shifts [106]:

$$\begin{aligned} p + r &= q + s && (4.34)\\ |p - q| \bmod 2 &= 0 && (4.35) \end{aligned}$$

Constraint (4.34) ensures the delays or advances in both bands are aligned, when the bands are added to form the output, while constraint (4.35), where p and q must be both even or both odd, ensures proper alignment prior to downsampling [106]. With this solution, the output is given by [106]:

$$\hat{X}(e^{j\omega}) = \frac{1}{2}\left[H_1(e^{j\omega})H_2(e^{j(\omega-\pi)}) - H_2(e^{j\omega})H_1(e^{j(\omega-\pi)})\right]e^{j\omega(p+r)}X(e^{j\omega}) \tag{4.36}$$

Therefore, the output is delayed by $(p+r)$ samples and hence can be controlled by picking appropriate values for p and r. As well as affecting the delay, these filter shifts will need to be further constrained, if the symmetric extension method is used. No constraints to p and q are needed when using the circular extension method [106].

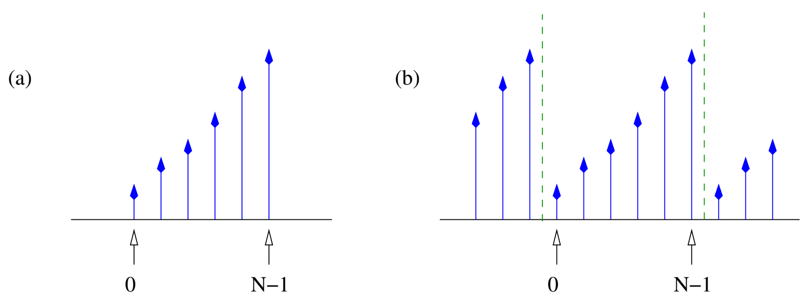


Figure 4.20: The circular extension method (after [106]): (a) Original signal of N samples; (b) periodically extended signal using circular extension (dotted line shows axis of symmetry).

With the second solution for perfect reconstruction, which applies to odd-tapped, biorthogonal wavelet filters [106]:

$$G_1(e^{j\omega}) = H_2(e^{j(\omega-\pi)}) \tag{4.37}$$

$$G_2(e^{j\omega}) = H_1(e^{j(\omega-\pi)}) \tag{4.38}$$

we have the following constraints on the filter shifts [106]:

$$p + r = q + s \tag{4.39}$$

$$|p - q| \bmod 2 = 1 \tag{4.40}$$

Here, either one of p and q is odd, with the other even.

4.5.2 Signal Extension Methods

As we mentioned before, any expansion of data as a result of filtering, is undesirable and can be prevented by forming a periodically extended version of the signal. The convolution of the impulse response of a filter with a periodic signal results in another periodic signal. As noted by Martucci [106], an extension of $M-1$ is enough to generate unique samples after filtering[10]. Two types of extensions have been investigated [84, 106, 164] and these are discussed in the following sections.

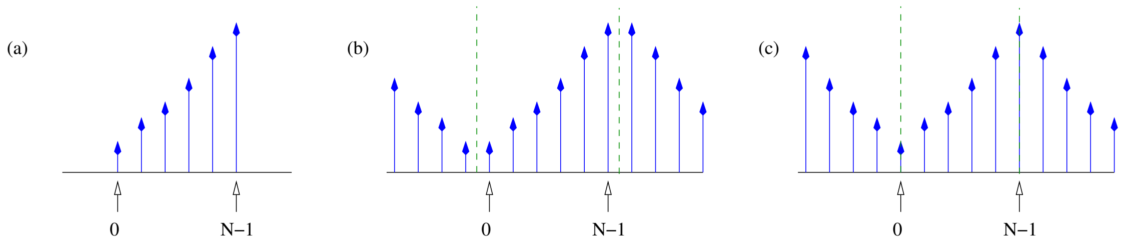


Figure 4.21: The symmetric extension method (after [106]): (a) Original signal of N samples; (b) periodically extended signal using symmetric extension with half-sample symmetry on both ends (dotted line shows axis of symmetry); (c) periodically extended signal using symmetric extension with whole-sample symmetry on both ends (dotted line shows axis of symmetry).

The Circular Extension Method

The circular extension method is the easiest of the two methods to implement since the convolution becomes cyclic, which is equivalent to multiplying the discrete Fourier transforms of the signal and filter. Both linear and non-linear phase filters can be used with this method as well [142]. The disadvantage with the circular convolution method is the discontinuity introduced by wrapping one end of the signal onto the other [142, 106]. Generally, there will be a difference in the pixel values on the edges, which results in a large amount of energy being captured in the high frequency subimage. Coarse quantisation of this subimages will cause ringing artifacts on the edge of the image.

The Symmetric Extension Method

The symmetric extension method produces better quality reconstructed images in subband coding than the circular extension method because of the smooth transition of the edge extension [142]. In fact, the periodic sequence formed from symmetric extension and the subsequent convolution is equivalent to the DCT [84]. Therefore, the reasons for why symmetric extension is better than circular extension are intimately related to the energy compaction characteristics of the DCT (or, DFT of a signal with a period of $2N$) as compared with those of the DFT (of a signal with a period of N). However, symmetric extension involves more constraints and is thus more difficult to implement than circular extension, though both require the same amount of computations [106]. Note that the input signal is assumed to have an even number of samples.

[10]If filters of different lengths, M_1 and M_2 are used, the larger of the two is used [106].

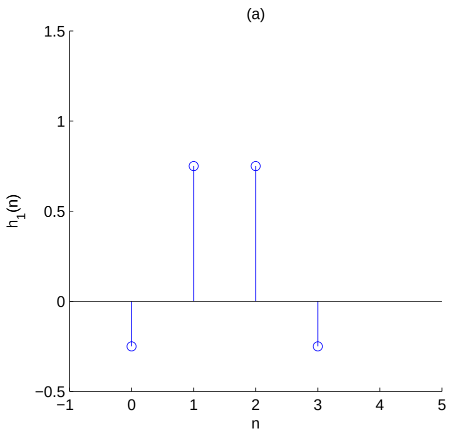
Table 4.2: Symmetry properties of convolved signals (after [106]), where WS – whole-sample, HS – half-sample, $y(n) = x(n) * h(n)$

$x(n)$	$h(n)$	$y(n)$
WS symmetric	WS symmetric	WS symmetric
HS symmetric	HS symmetric	WS symmetric
HS symmetric	WS symmetric	HS symmetric
symmetric	symmetric	symmetric
anti-symmetric	anti-symmetric	symmetric
symmetric	anti-symmetric	anti-symmetric

Firstly, it is important to distinguish between two different types of symmetry that are possible, as described in [106]. Figure 4.21(b) shows an example of what is termed, *half-sample symmetry*. Here, the edge value is replicated and the axis of symmetry falls in between these two samples. Figure 4.21(c) shows an example of *whole-sample symmetry*, where the axis of symmetry coincides with the edge sample. Also, because linear phase filters are always symmetric (or anti-symmetric), then it is apparent that a whole-sample symmetric filter has an odd number of taps while a half-sample symmetric filter has an even number of taps. Secondly, it is well known that the convolution of a symmetric signal with a symmetric filter will result in a symmetric output signal. Because there are two forms of symmetry possible, Martucci [106] tabulated all possible cases of the convolution of two symmetric sequences, $x(n)$ and $h(n)$, and the output, $y(n)$, which is reproduced in Table 4.2.

Referring to the analysis stage of Figure 4.18, the input signal, $x(n)$, is symmetrically extended to produce, $\tilde{x}(n)$. The type of symmetric extension to be used is dependent on the filter we are using. We should always aim for the output of the filter, $v(n)$, *to have whole-sample symmetry*. Therefore, according to Table 4.2, if we are using an even-tapped filter (half-sample symmetric), then the correct extension of the signal would be half-sample symmetry. Likewise, if we are using an odd-tapped filter (whole-sample symmetric), then the correct extension of the signal would be whole-sample symmetry.

Figures 4.23 and 4.24 show a series of graphs, which appear in [106], that depict symmetric extension, filtering, and downsampling for even and odd-tapped filters, which are shown in 4.22.



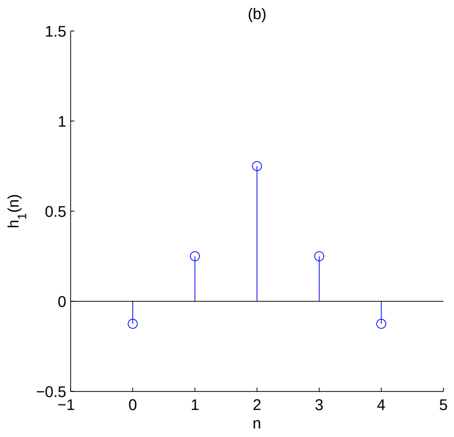


Figure 4.22: Impulse response of an even and odd-tapped symmetric filter: (a) $H_1(z) = \frac{1}{4}(-1 + 3z^{-1} + 3z^{-2} - z^{-3})$; (b) $H_1(z) = \frac{1}{8}(-1 + 2z^{-1} + 6z^{-2} - 2z^{-3} - z^{-4})$ (after [106])

Using Filters with an Even Number of Taps

The filtering of the symmetrically extended input signal, $\tilde{x}(n)$, will produce a periodic output signal, $v(n)$, that possesses whole-sample symmetry on both ends. If an even-tapped filter (Figure 4.22(a)) is shifted to have a half sample advance, $h_1(n + 2)$, then the output, $v(n)$, will have $N + 1$ unique samples, as shown in Figure 4.23(c). If we are downsampling by two in the subband coder, then there will be two different ways of doing it since there is an odd number of samples (assuming N is even, $N + 1$ will be odd), with one resulting in $\frac{N}{2}$ samples (top set of ticks in Figure 4.23(c)) and the other resulting in $\frac{N}{2}+1$ samples in $y(n)$ (bottom set of ticks in Figure 4.23(c)) [106]. Because any expansion of data should be avoided, the first method of downsampling, which results in $\frac{N}{2}$, is the preferred one.

At the synthesis stage, the signal, $y(n)$, is symmetrically extended to form, $\tilde{y}(n)$. The type of symmetric extension required at both ends is apparent in Figure 4.23(d), which shows the upsampled version of $\tilde{y}(n)$. We note that based on the way we have performed the downsampling, the synthesis stage has enough unique samples to extend and recreate the periodic structure of $v(n)$, albeit with half the number of the samples, via half-sample symmetry on both ends. Also, if the filter is anti-symmetric, as is the case with high pass QMFs with an even number of taps, then anti-symmetric extension will need to applied instead.

Finally, in order to determine the shifts required in the synthesis filters, we refer to

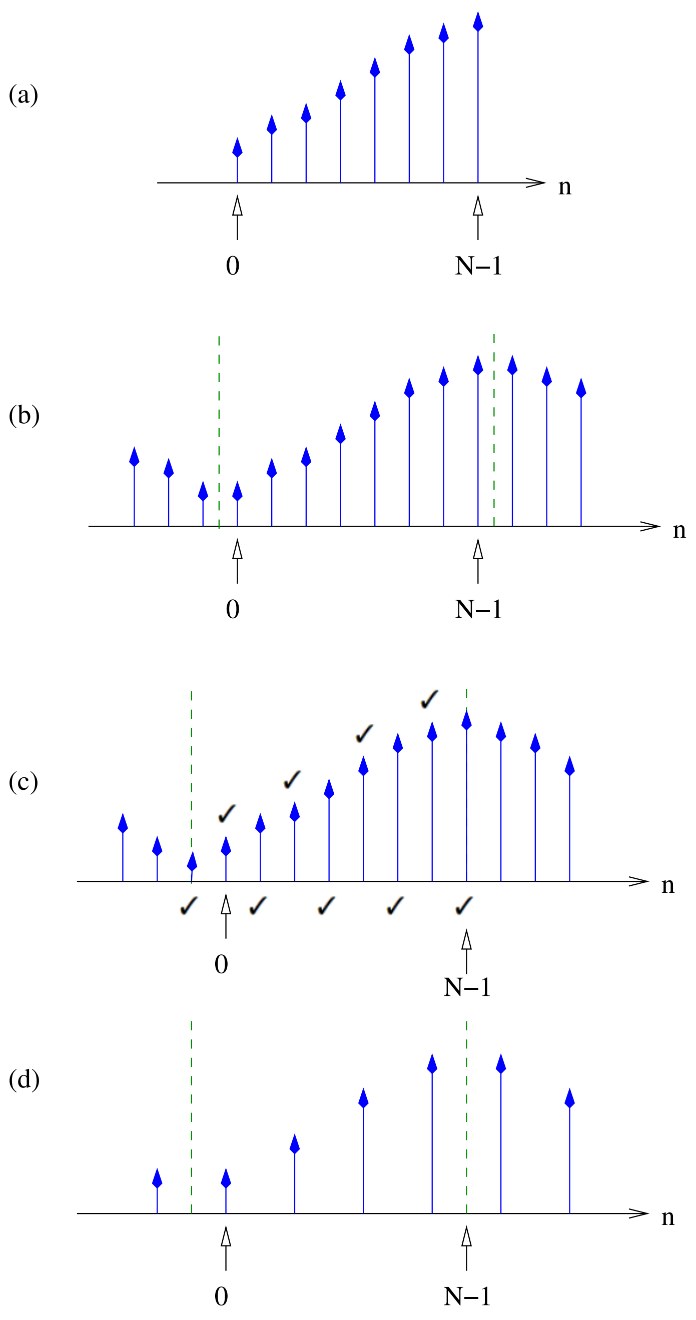


Figure 4.23: Symmetric extension for filters with an even number of taps (after [106]): (a) Original signal, $x(n)$; (b) symmetrically extended signal, $\tilde{x}(n)$ (both ends with half-sample symmetry); (c) filtered signal, $v(n)$. The ticks above and below the graph show two different ways of downsampling; (d) symmetrically extended, upsampled signal, $w(n)$.

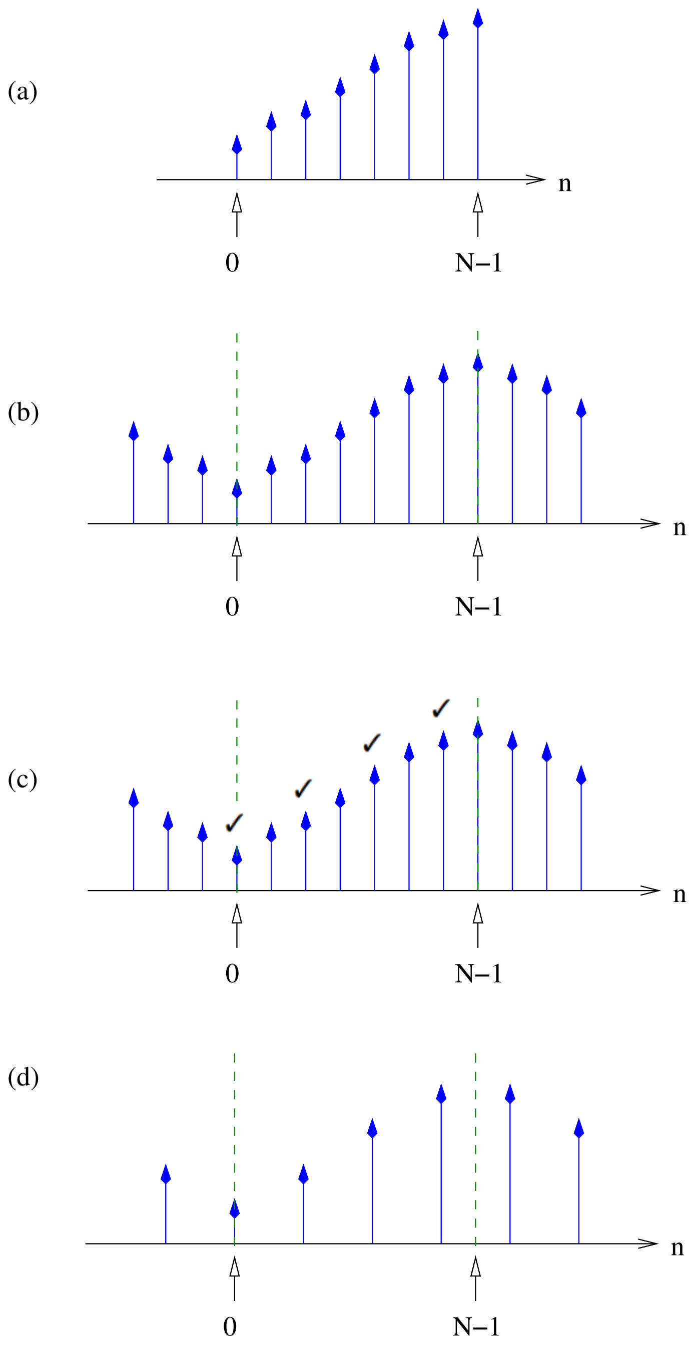


Figure 4.24: Symmetric extension for filters with an odd number of taps (after [106]): (a) Original signal, $x(n)$; (b) symmetrically extended signal, $\tilde{x}(n)$ (both ends with whole-sample symmetry); (c) filtered signal, $v(n)$. The ticks above the graph show the samples that are retrained after downsampling; (d) symmetrically extended, upsampled signal, $w(n)$.

constraint (4.34). With $p = 2$ and $q = 2$ which imparts a half-sample advance, then in order for there to be no delay on the output, we choose $r = 1$ and $s = 1$ which impart a half-sample delay [106].

Using Filters with an Odd Number of Taps

For an odd-tapped filter (Figure 4.22(b)), which has whole-sample symmetry, then according to Table 4.2, we need to use whole-sample symmetry at both ends, as shown in Figure 4.24(b). If the odd-tapped filter is shifted to impart no delay, $h(n + 2)$, then the output, $v(n)$, will have N unique samples, as shown in Figure 4.24(c). Because there are an even number of samples in this case, then there is only one way of downsampling by two to produce $\frac{N}{2}$ samples in $y(n)$.

At the synthesis stage, the signal, $y(n)$, is symmetrically extended to form, $\tilde{y}(n)$. The type of symmetric extension required at both ends is apparent in Figure 4.24(d). We note that the synthesis stage has enough unique samples to recreate the periodic structure of $v(n)$ via whole-sample symmetry on the left and half-sample symmetry on the right. If our filters are shifted to impart a one sample advance, then the reverse symmetry should be used. That is, half-sample symmetry on the left and whole-sample symmetry on the right [106].

4.5.3 An Example of Symmetric Extension for Subband Decomposition of Finite Length Signals

In this section, we discuss the implementation, via a MATLAB demonstration, of the analysis and synthesis stages of a subband coder using the symmetric extension method. No quantisation will be performed, hence perfect reconstruction is to be sought, as measured by the signal-to-noise ratio (SNR) of the output. The QMFs used are the 32-tap Johnston filters [79], whose coefficients are given in Table 4.3. In order to simplify the implementation, causal filtering will be used along with appropriate truncation.

The input signal that we will be considering is shown is a ramp, as shown in Figure 4.25, where $N = 100$. Because our filters have an even number of taps, then it possesses half-sample symmetry. We aim for whole-sample symmetry on the filter outputs, thus according to Table 4.2, we extend our input signal by $M - 1 = 15$ samples on both ends

Table 4.3: Coefficients of the 32-tap Johnston symmetric QMFs (after [79])

n	$h(n)$
0	2.245139e-3
1	-3.971152e-3
2	-1.969672e-3
3	8.181941e-3
4	8.426833e-4
5	-1.422899e-2
6	2.069470e-3
7	2.270415e-2
8	-7.961731e-3
9	-3.496440e-2
10	1.947218e-2
11	5.481213e-2
12	-4.452423e-2
13	-9.933859e-2
14	1.329725e-1
15	4.636741e-1
16	4.636741e-1
17	1.329725e-1
18	-9.933859e-2
19	-4.452423e-2
20	5.481213e-2
21	1.947218e-2
22	-3.496440e-2
23	-7.961731e-3
24	2.270415e-2
25	2.069470e-3
26	-1.422899e-2
27	8.426833e-4
28	8.181941e-3
29	-1.969672e-3
30	-3.971152e-3
31	2.245139e-3

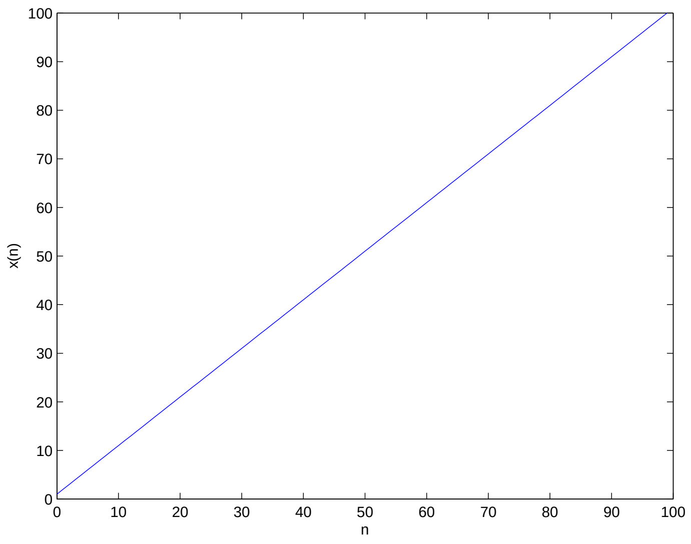


Figure 4.25: Ramp signal, $x(n) = n + 1$ where $n = 0, 1, \ldots, N$

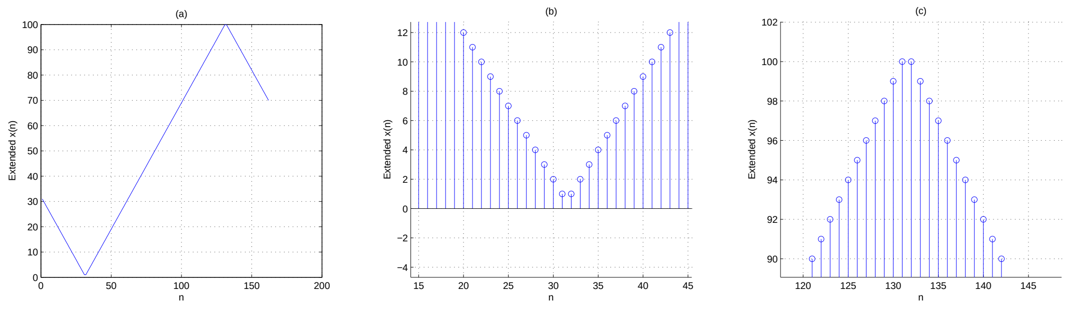


Figure 4.26: Symmetrically extended ramp signal, $\tilde{x}(n)$: (a) Plot of $\tilde{x}(n)$; (b) zoomed-in section of the left edge, showing half-sample symmetry; (c) zoomed-in section of the right edge, showing half-sample symmetry

using half-sample symmetry. This extended signal is shown in Figure 4.26.

We apply the low-pass and high-pass analysis QMFs to the symmetrically extended input to produce a low-pass signal, $v_1(n)$ and $v_2(n)$, which are shown in Figure 4.27. We can see in Figures 4.27(b) and (d) that the symmetry of both signals has become whole-sample symmetry and centred at $n = 47$. Also, because the high-pass QMF is anti-symmetric, the high-pass filtered signal in Figure 4.27(d) possesses anti-symmetry as well.

In order to obtain $\frac{N}{2} = 50$ samples, the downsampling process should start by retaining the sample at $n = 48$ and onwards. At the synthesis stage, the filtered signals are extended

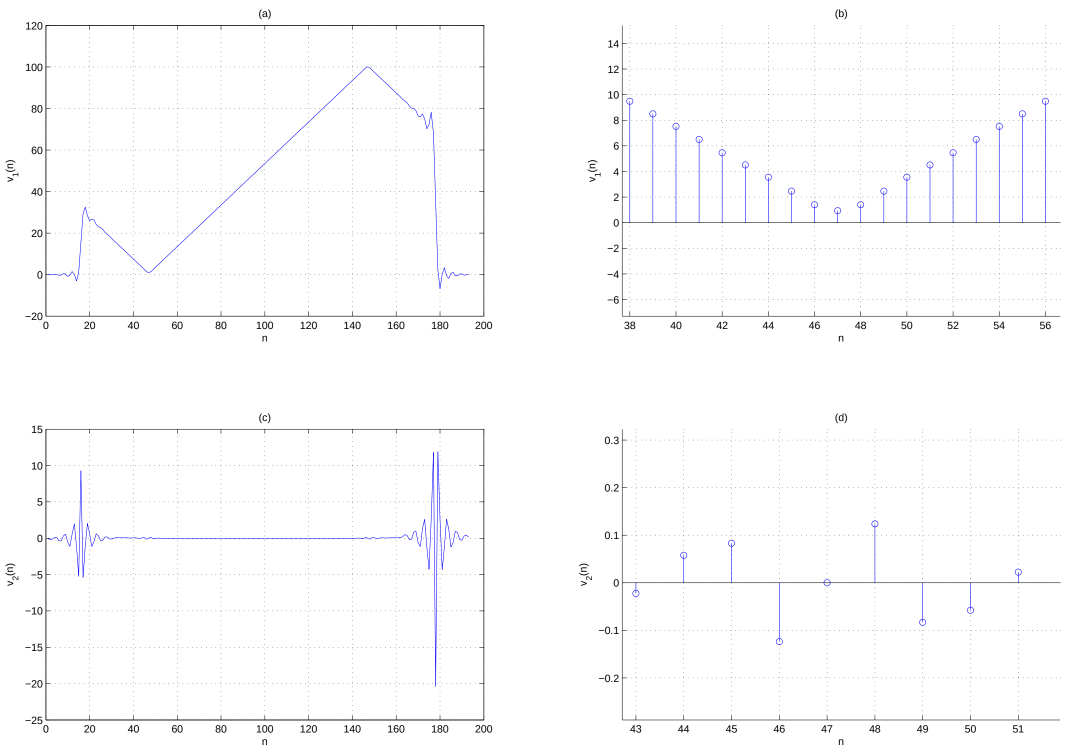


Figure 4.27: (a) Low-pass filtered signal, $v_1(n)$; (b) zoomed-in section of left edge of $v_1(n)$; (c) high-pass filtered signal, $v_2(n)$; (d) zoomed-in section of the left edge of $v_2(n)$

using half-sample symmetry (symmetric on low-pass, anti-symmetric on high-pass) on both ends and then upsampled. These signals are shown in Figure 4.28, where the axis of symmetry is $n = 62$.

Figure 4.29 shows the reconstructed signals, $\hat{x}_1(n)$ and $\hat{x}_2(n)$, formed by filtering the extended and upsampled signals, $\tilde{w}_1(n)$ and $\tilde{w}_2(n)$. Both exhibit half-sample symmetry around $n = 77.5$. The final reconstructed signal, $\hat{x}(n)$, is formed by summing $\hat{x}_1(n)$ and $\hat{x}_2(n)$ and truncating the samples at $n < 78$. A gain of two needs to be applied to the final signal for perfect reconstruction with an SNR of 84.82 dB.

4.5.4 An Example of Symmetric Extension for the Discrete Wavelet Transform of Finite Length Signals

In this section, we discuss the implementation, via a MATLAB demonstration, of the forward and inverse discrete wavelet transform using the symmetric extension method. No quantisation will be performed, hence perfect reconstruction is to be sought, as measured

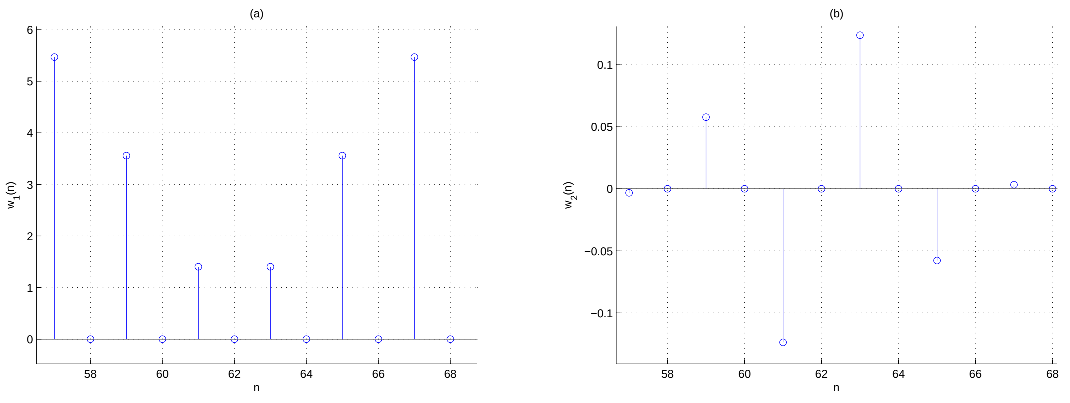


Figure 4.28: (a) Zoomed-in section of left edge of $\tilde{w}_1(n)$; (b) zoomed-in section of the left edge of $\tilde{w}_2(n)$

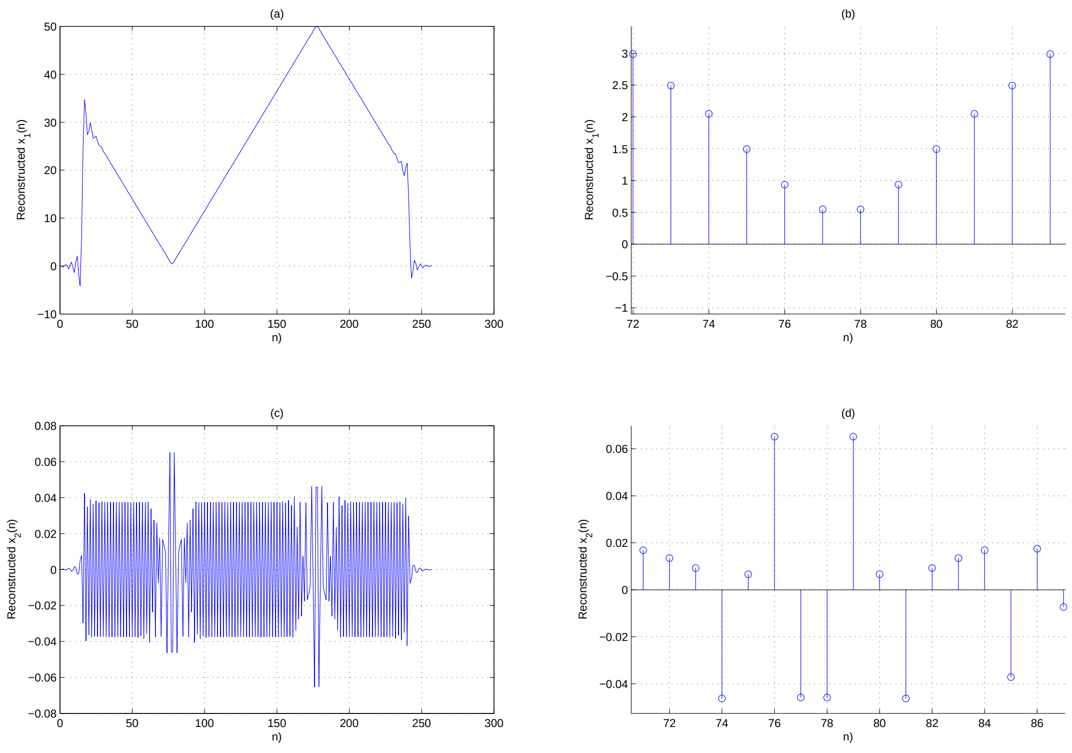


Figure 4.29: (a) Low-pass reconstructed signal, $\hat{x}_1(n)$; (b) zoomed-in section of $\hat{x}_1(n)$; (c) High-pass reconstructed signal, $\hat{x}_2(n)$; (d) zoomed-in section of $\hat{x}_2(n)$.

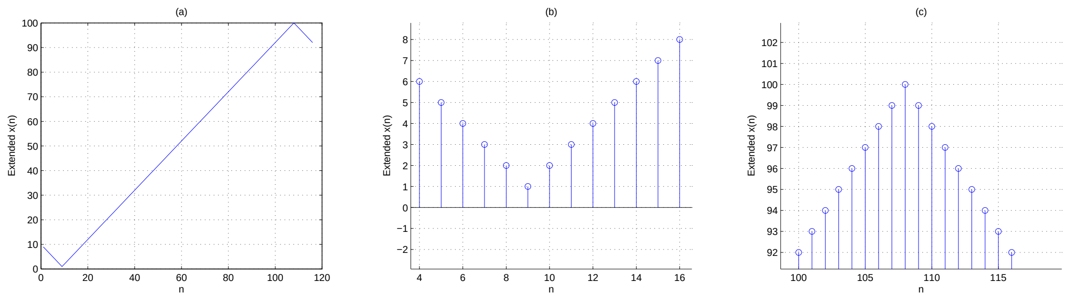


Figure 4.30: Symmetrically extended ramp signal, $\tilde{x}(n)$: (a) Plot of $\tilde{x}(n)$; (b) zoomed-in section of the left edge, showing whole-sample symmetry; (c) zoomed-in section of the right edge, showing whole-sample symmetry

by the signal-to-noise ratio (SNR) of the output. The wavelets used are the 9/7-tap biorthogonal wavelets of Antonini *et al.* [7] whose coefficients are given in Table 4.1. In order to simplify the implementation, causal filtering will be used along with appropriate truncation.

The input signal that we will be considering is the ramp of the previous section (Figure 4.25), where $N = 100$. Because our filters have an odd number of taps, then they possess whole-sample symmetry. We aim for whole-sample symmetry on the filter outputs, thus according to Table 4.2, we extend our input signal by 8 samples[11] on both ends using whole-sample symmetry. This extended signal is shown in Figure 4.30.

We apply the low-pass and high-pass analysis wavelet filters to the symmetrically extended input to produce a low-pass signal, $v_1(n)$ and $v_2(n)$, which are shown in Figure 4.31. We can see in Figures 4.31(b) and (d) that the symmetry of low and high-pass signals have become whole-sample symmetry and centred at $n = 13$ and $n = 12$, respectively. The difference in the symmetry centres are mostly due to the difference in lengths of the filters.

In order to obtain $\frac{N}{2} = 50$ samples, the downsampling process should start by retaining the sample at $n = 13$ and onwards. Note that by retaining sample 13 and onwards in the high-pass signal, the symmetry of the subsequent upsampled version will become half-sample. Hence, at the synthesis stage, the low-pass, downsampled signals are extended using whole-sample symmetry on the left end and half-sample symmetry on the right end, while for the high-pass signal, it is extended using half-sample symmetry on the left and whole-sample on the right.

[11]We choose to use the length of the longest filter, minus one.

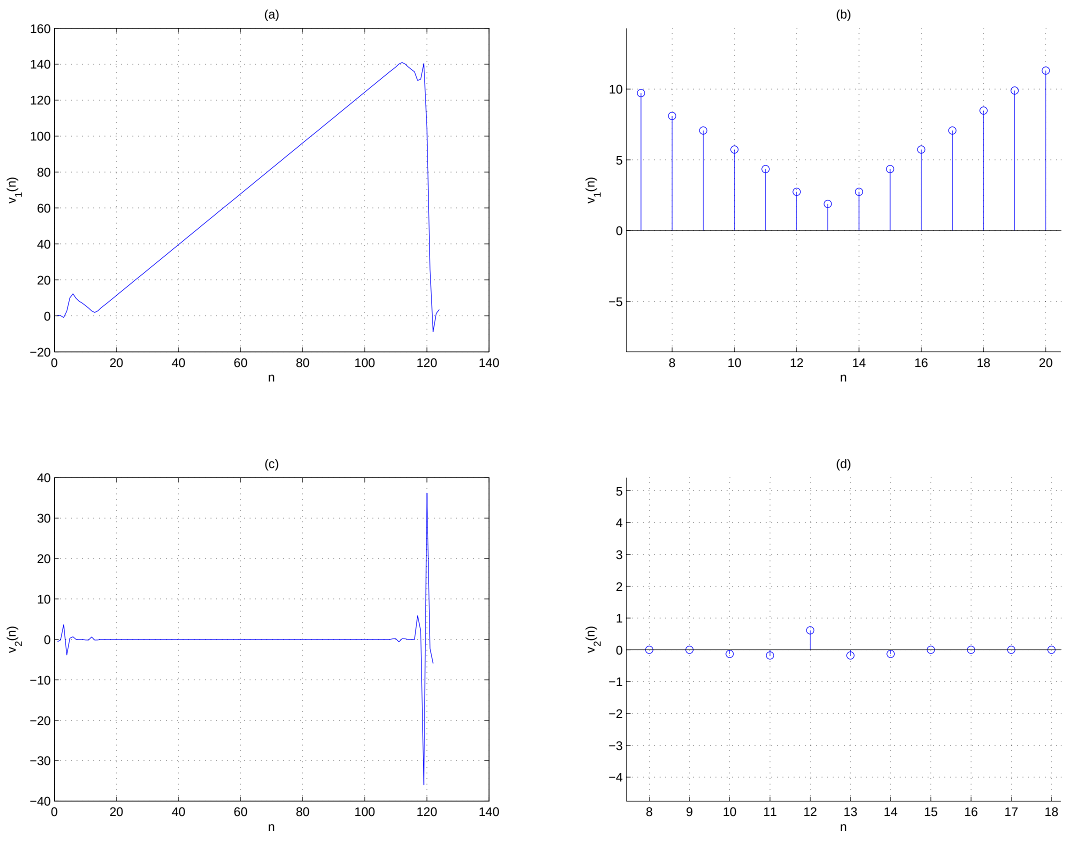


Figure 4.31: (a) Low-pass filtered signal, $v_1(n)$; (b) zoomed-in section of left edge of $v_1(n)$; (c) high-pass filtered signal, $v_2(n)$; (d) zoomed-in section of the left edge of $v_2(n)$

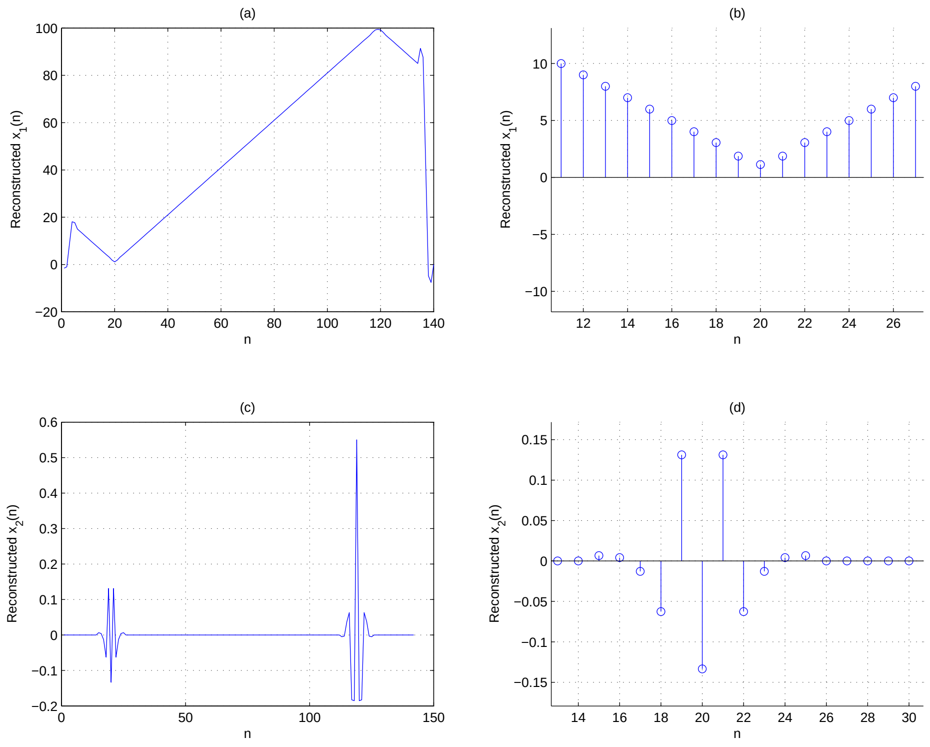


Figure 4.32: (a) Low-pass reconstructed signal, $\hat{x}_1(n)$; (b) zoomed-in section of $\hat{x}_1(n)$; (c) High-pass reconstructed signal, $\hat{x}_2(n)$; (d) zoomed-in section of $\hat{x}_2(n)$.

Figure 4.32 shows the reconstructed signals, $\hat{x}_1(n)$ and $\hat{x}_2(n)$, formed by filtering the extended and upsampled signals, $\tilde{w}_1(n)$ and $\tilde{w}_2(n)$. Both exhibit whole-sample symmetry around $n = 20$. The final reconstructed signal, $\hat{x}(n)$, is formed by summing $\hat{x}_1(n)$ and $\hat{x}_2(n)$ and truncating the samples at $n < 20$. The final signal has an SNR of 275.18 dB. It is particularly interesting to note that the 9/7-tap biorthogonal wavelet filters achieves a significantly higher SNR than the 32-tap Johnston QMF.

4.6 Experimental Setup for Image Coding Simulations

The rest of this chapter is dedicated to evaluating and discussing the application of block quantisation schemes in image coding. In particular, we present PSNR results and reconstructed images for each quantisation scheme and do a comparison. Hence, we have collected a training and testing set of images to be used for the evaluation of each scheme.

The training image set consists of 18 images and the testing image set consists of 6 images, which are not part of the training set. They are all 8-bit greyscale images, hence the luminance values are within the range of 0 and 255. The image names and their dimensions are shown in Table 4.4. The train and test image sets are also shown in Figures 4.33 and 4.34.

For the block quantisation experiments, blocks of 8×8 pixels are used. Therefore, the training image set contains 73728 blocks or vectors. Peak signal-to-noise ratios (PSNR) are used to objectively judge image quality. Gaussian optimised Lloyd-Max scalar quantisation levels up to 256 levels (8 bits) were generated and used in the block quantisation experiments.

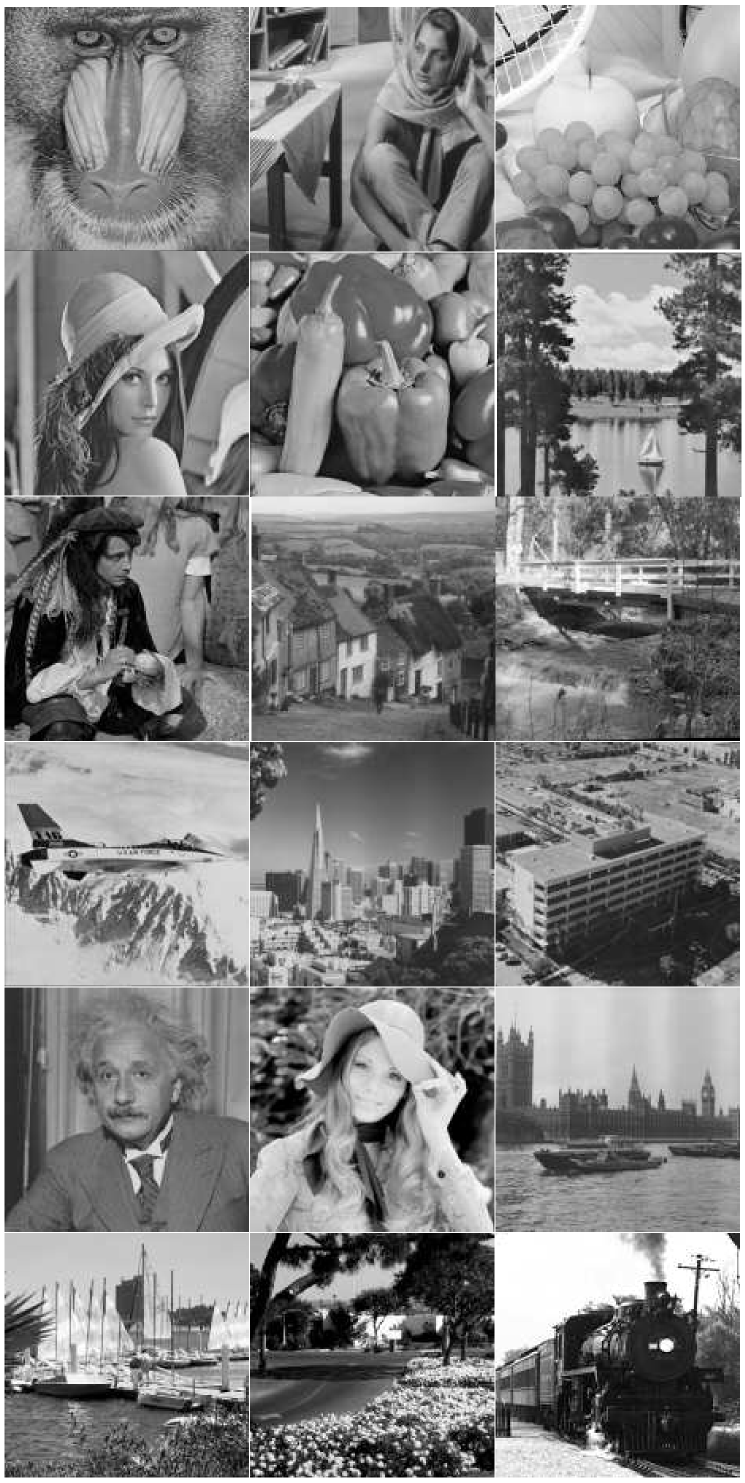


Figure 4.33: Training set of images

Table 4.4: Training and testing image set for image coding experiments

Training Set		Testing Set	
Image Name	Dimensions	Image Name	Dimensions
baboon	512×512	boat	512×512
barbara	512×512	kids	512×512
fruits	512×512	crowd	512×512
lena	512×512	couple	512×512
peppers	512×512	mill	512×512
sailboat	512×512	vegas	512×512
man	512×512		
goldhill	512×512		
bridge	512×512		
jet	512×512		
pyramid	512×512		
aero	512×512		
einstein	512×512		
hat	512×512		
london	512×512		
tekboat	512×512		
tekrose	512×512		
loco	512×512		

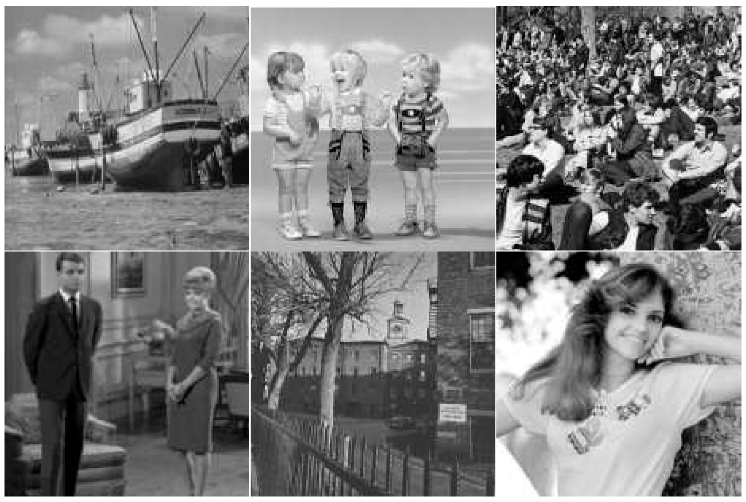


Figure 4.34: Testing set of images

4.7 Application of the Block Quantiser

In this section, we evaluate the traditional block quantiser in an image coding scenario using two different transforms. The KLT-based scheme is expected to perform the best while the DCT should achieve slightly lower performance. With the block quantiser, the image source is assumed to be Gaussian and the performance of the DCT-based scheme will be highly dependent on how similar the image source is to a highly correlated Gauss-Markov process. The aim of this experiment is to provide a baseline for comparison with the GMM-based block quantiser and the improvements, in terms of quantisation performance as measured by PSNR and subjective image quality, that result from accurate PDF modelling and multiple transforms.

4.7.1 The KLT-Based Block Quantiser

Table 4.5 shows the PSNRs of different images (from both the training and testing set) at different bitrates, when using the KLT-based block quantiser. We can see that as the bitrate is increased, the PSNR improves for all images, even for those that were not part of the training set.

Tables 4.6, 4.7, and 4.8 show the allocation of quantiser levels to each of the 64 transform coefficients at different bitrates. Because the bit allocation is based on the variance of the component, more bits are assigned to the first coefficient in the top-left. As the bitrate is increased, more bits are gradually assigned to the higher transform components which should lead to better visual quality. Note that at each bitrate, all image blocks are quantised using the same bit allocation, which is based on the assumption that all blocks possess the same statistics.

Figure 4.35 shows the original and reconstructed images at various bitrates of 'goldhill', which was part of the training set. Looking at Figure 4.35(b), it can be seen that there is a high degree of block artifacts and graininess, especially in the sky, at a bitrate of 0.25 bits/pixel. At 0.5 bits/pixel (Figure 4.35(c)), the granular noise and block artifacts have been reduced, particularly in the sky and the walls of the white houses. At 1 bit/pixel (Figure 4.35(d)), the block artifacts have been considerably reduced though there is a slight amount of granular noise.

Table 4.5: PSNR as a function of bitrate for the KLT-based block quantiser

Image Name	PSNR (in dB)			
	0.25 bits/pixel	0.5 bits/pixel	0.75 bits/pixel	1.0 bits/pixel
baboon	21.50	22.89	23.80	24.60
barbara	22.60	23.75	24.51	25.25
fruits	24.48	26.08	26.90	28.06
lena	25.46	27.31	28.18	29.66
peppers	25.31	27.18	28.05	29.48
sailboat	23.73	25.80	26.77	28.32
man	23.02	24.67	25.55	26.79
goldhill	26.17	28.08	29.18	30.54
bridge	22.88	24.69	25.78	26.97
jet	23.74	25.72	26.56	27.92
pyramid	25.17	27.33	28.23	29.53
aero	25.00	27.47	28.45	30.00
einstein	27.12	29.00	29.87	31.15
hat	25.30	27.07	27.77	29.21
london	25.92	27.94	28.84	30.18
tekboat	19.43	20.82	21.56	22.49
tekrose	17.59	18.90	19.56	20.52
loco	20.73	22.18	22.92	23.95
boat	24.24	26.02	26.94	28.26
kids	22.00	23.14	23.80	24.60
crowd	17.34	18.80	19.45	20.55
couple	27.68	29.04	29.61	30.60
mill	21.27	22.77	23.65	24.67
vegas	25.60	27.62	28.53	29.99

Table 4.6: Levels allocation table for KLT-based block quantiser at 0.25 bits/pixel

32	4	4	2	2	2	2	2
2	2	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1

Table 4.7: Levels allocation table for KLT-based block quantiser at 0.5 bits/pixel

32	8	8	4	4	4	4	2
2	2	2	2	2	2	2	2
2	2	2	2	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1

Table 4.8: Levels allocation table for KLT-based block quantiser at 1.0 bits/pixel

64	16	16	8	8	8	4	4
4	4	4	4	4	4	4	2
2	2	2	2	2	2	2	2
2	2	2	2	2	2	2	2
2	2	2	2	2	2	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1

Figure 4.36 shows the original and reconstructed images at various bitrates of ‘boat’, which was not part of the training set. Like ‘goldhill’, there is a large amount of block artifacts and granular distortion throughout the entire image at 0.25 bits/pixel and these gradually reduce as the bitrate is increased. At 1 bit/pixel (Figure 4.36(d)), the block artifacts are less noticeable but the smooth regions like the sky have a slight amount of granular distortion.

The graininess is due to the coarse quantisation of the high frequency KLT coefficients since the quantisation noise in each coefficient will be spatially spread to all pixels within the block. As the bitrate is increased, more bits are allocated to these transform coefficients, reducing the quantisation noise as well as the subsequent granular noise in the spatial domain.

4.7.2 The DCT-Based Block Quantiser

Table 4.9 shows the PSNRs of different images (from both the training and testing set) at different bitrates, when using the DCT-based block quantiser. We can see that as the bitrate is increased, the PSNR improves for all images, even for those that were not part

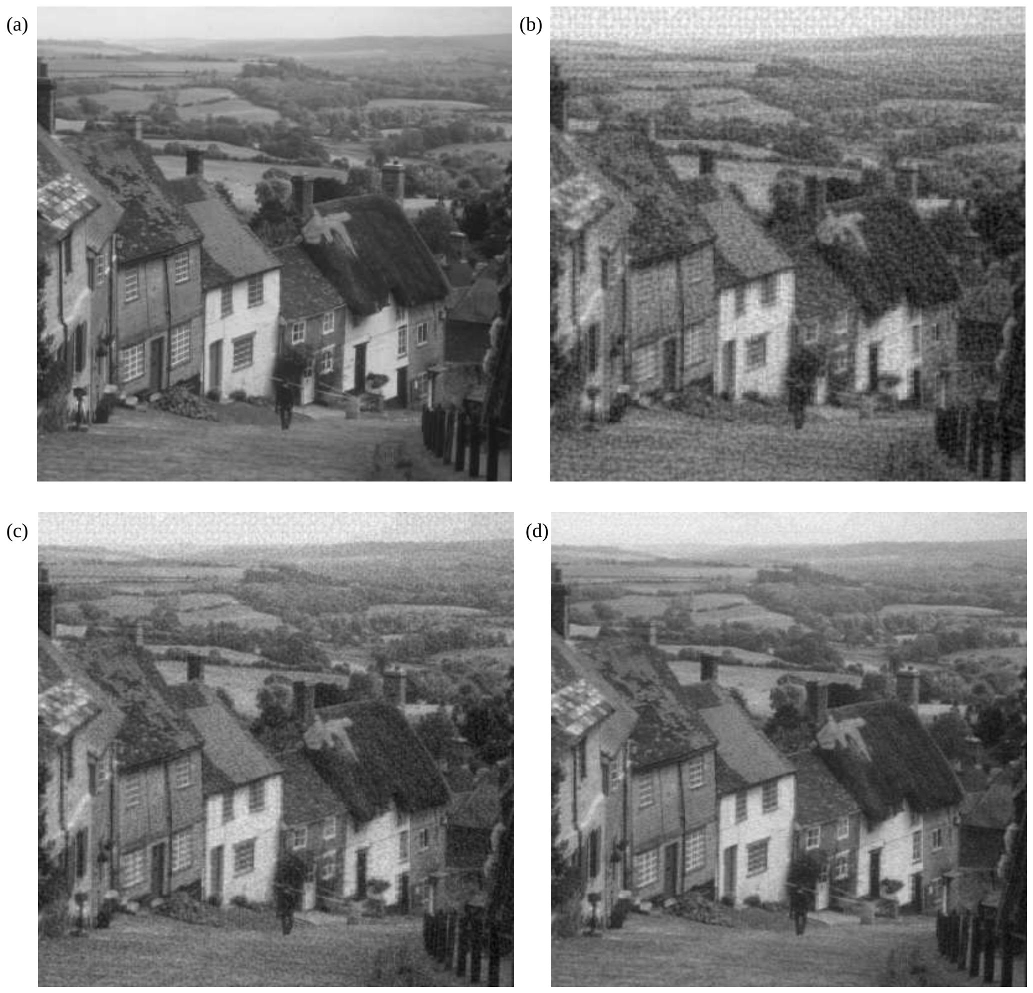


Figure 4.35: Results of the ‘goldhill’ image at various bitrates (part of training set) using the KLT-based block quantiser: (a) original 8-bit image; (b) 0.25 bits/pixel (PSNR=26.17 dB); (c) 0.5 bits/pixel (PSNR=28.08 dB); (d) 1.0 bits/pixel (PSNR=30.54 dB)

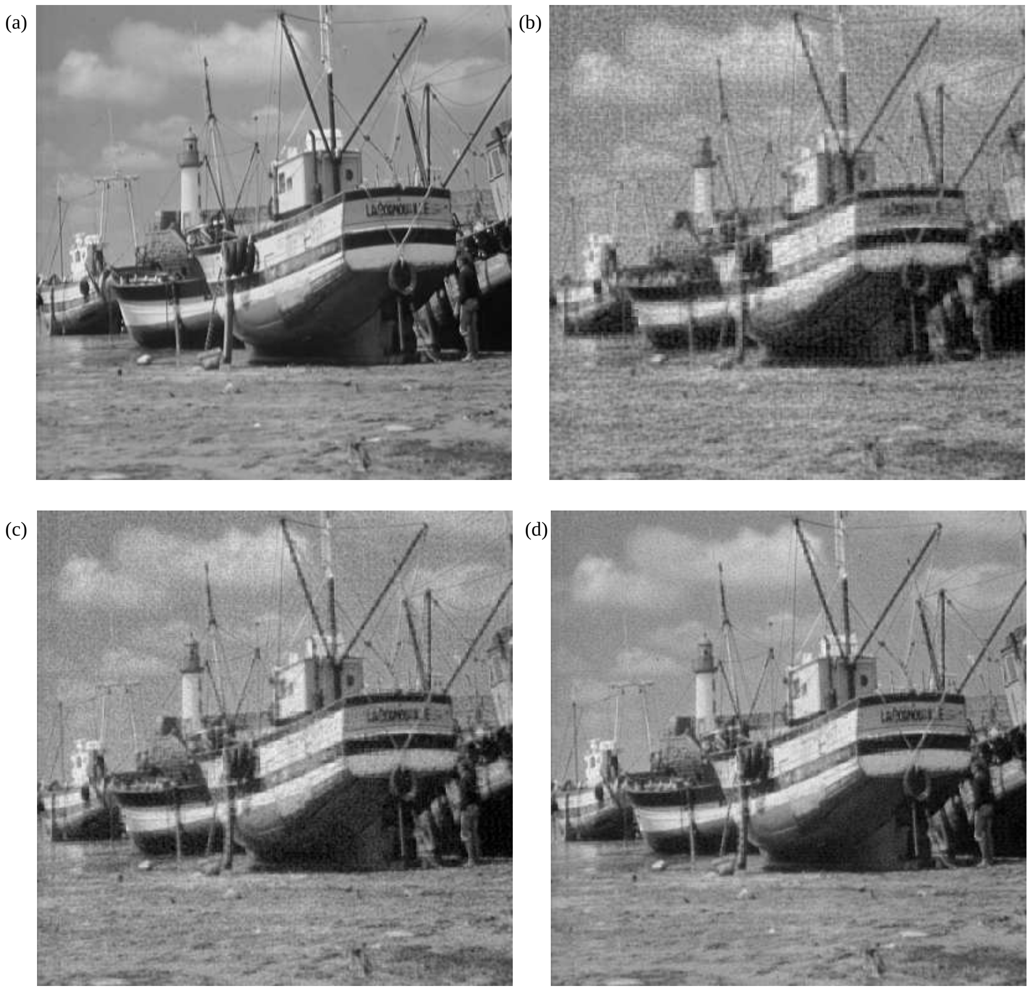


Figure 4.36: Results of the 'boat' image at various bitrates (not part of training set) using the KLT-based block quantiser: (a) original 8-bit image; (b) 0.25 bits/pixel (PSNR=24.24 dB); (c) 0.5 bits/pixel (PSNR=26.02 dB); (d) 1.0 bits/pixel (PSNR=28.26 dB)

Table 4.9: PSNR as a function of bitrate for the DCT-based block quantiser

Image Name	PSNR (in dB)			
	0.25 bits/pixel	0.5 bits/pixel	0.75 bits/pixel	1.0 bits/pixel
baboon	21.41	22.75	23.57	24.41
barbara	22.46	23.55	24.19	24.97
fruits	24.36	25.93	26.64	27.88
lena	25.25	27.04	27.75	29.34
peppers	25.04	26.80	27.48	28.99
sailboat	23.54	25.54	26.39	27.99
man	22.87	24.47	25.27	26.53
goldhill	25.91	27.78	28.73	30.21
bridge	22.70	24.46	25.46	26.66
jet	23.50	25.38	26.08	27.51
pyramid	24.94	26.99	27.65	29.12
aero	24.79	27.13	27.97	29.59
einstein	26.90	28.77	29.44	30.92
hat	25.17	26.91	27.50	29.03
london	25.59	27.54	28.26	29.74
tekboat	19.26	20.57	21.21	22.14
tekrose	17.51	18.79	19.42	20.37
loco	20.55	21.92	22.58	23.61
boat	24.06	25.76	26.55	27.88
kids	21.87	22.99	23.60	24.43
crowd	17.24	18.64	19.24	20.32
couple	27.31	28.73	29.09	30.36
mill	21.08	22.50	23.25	24.28
vegas	25.35	27.35	28.11	29.68

Table 4.10: Levels allocation table for DCT-based block quantiser at 0.25 bits/pixel

32	4	2	2	1	1	1	1
4	2	2	1	1	1	1	1
2	2	1	1	1	1	1	1
2	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1

Table 4.11: Levels allocation table for DCT-based block quantiser at 0.5 bits/pixel

32	8	4	2	2	1	1	1
8	4	2	2	2	1	1	1
4	2	2	2	1	1	1	1
4	2	2	1	1	1	1	1
2	2	1	1	1	1	1	1
2	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1

of the training set. Comparing with the PSNRs of the KLT-based block quantiser in Table 4.5, we can see that the DCT-based block quantiser is performs slightly worse, which is to be expected. However, it can be concluded that the coding efficiency of the DCT is quite close to that of the KLT for image data, which suggests that image sources are highly correlated and are similar to Gauss-Markov processes. This confirms past studies such as [116] that the DCT is a good alternative to the KLT in image coding.

Tables 4.10, 4.11, and 4.12 show the allocation of quantiser levels to each of the 64 transform coefficients at different bitrates. Because the bit allocation is based on the variance of the component, more bits are assigned to the first coefficient in the top-left, which represents the DC frequency. As the bitrate is increased, more bits are gradually assigned to the higher frequency components which should lead to better visual quality. Comparing the bit allocation to those in Tables 4.6, 4.7, and 4.8, we notice that the larger variance coefficients are linearly ordered row-wise (starting from the top) for the KLT-based block quantiser while in the DCT-based block quantiser, they are linearly ordered row-wise and column-wise. This is because the KLT essentially operates on one dimensional vectors while the DCT is two-dimensional, transforming both rows and columns.

Figure 4.37 shows the original and reconstructed images at various bitrates of 'goldhill',

Table 4.12: Levels allocation table for DCT-based block quantiser at 1.0 bits/pixel

64	16	8	4	4	2	2	1
16	8	4	4	2	2	2	1
8	4	4	2	2	2	2	1
4	4	2	2	2	2	1	1
4	2	2	2	2	1	1	1
2	2	2	1	1	1	1	1
2	2	1	1	1	1	1	1
2	1	1	1	1	1	1	1

which was part of the training set. Like the KLT-based block quantiser, the reconstructed images exhibit a high degree of block artifacts and graininess at the low bitrates and these gradually reduce as the bitrate is increased. Though the PSNRs of the DCT-based block quantiser are slightly lower than those of the KLT-based block quantiser, subjective comparison of the reconstructed images shows minimal visual differences.

Looking at Figure 4.37(b), it can be seen that there is a high degree of block artifacts and graininess, especially in the sky, at a bitrate of 0.25 bits/pixel. At 0.5 bits/pixel (Figure 4.35(c)), the granular noise and block artifacts have been reduced, particularly in the sky and the walls of the white houses. At 1 bit/pixel (Figure 4.35(d)), the block artifacts have been considerably reduced though there is a slight amount of granular noise. As mentioned previously, this granular noise that spatially occurs within the block is due to the noise introduced by coarse coefficient quantisation in the transform domain. As the bitrate is increased, and more and more bits are allocated to these high frequency coefficients, the noise in the spatial domain is reduced.

4.7.3 The K-Means-Based Multiple Transform Block Quantiser

This is the quantisation scheme described in Section 2.4.3, where the vector space is partitioned into m disjoint regions using the K-means clustering algorithm and multiple block quantisers are designed for the vectors belonging to each region. This scheme is expected to perform better than the single transform block quantiser of the previous section because it does not assume correlation to be uniform throughout the vector space. The clustering process is unsupervised and is done via the K-means algorithm, which minimises the MSE between the vectors and the closest centroid.

Table 4.13 shows the PSNRs of all images as a function of the number of clusters

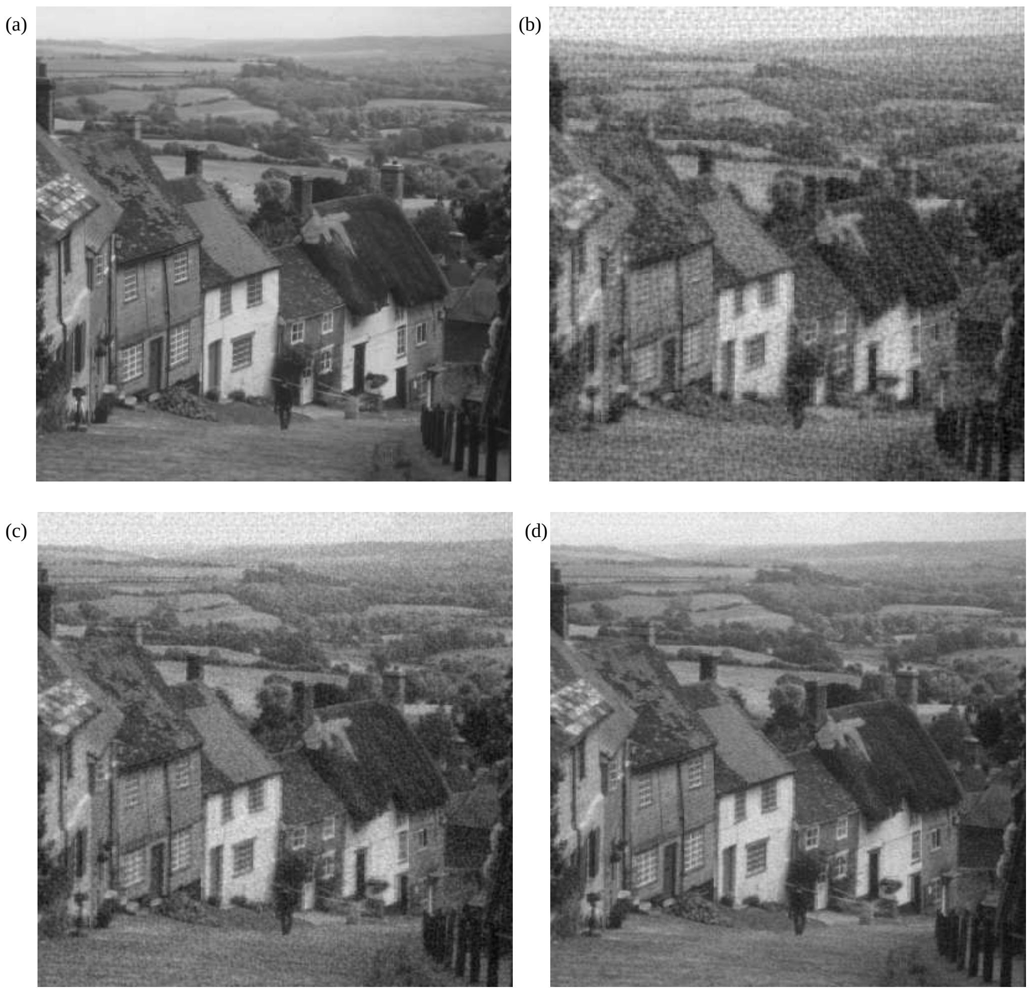


Figure 4.37: Results of the 'goldhill' image at various bitrates (part of training set) using the DCT-based block quantiser: (a) original 8-bit image; (b) 0.25 bits/pixel (PSNR=25.91 dB); (c) 0.5 bits/pixel (PSNR=27.78 dB); (d) 1.0 bits/pixel (PSNR=30.21 dB)

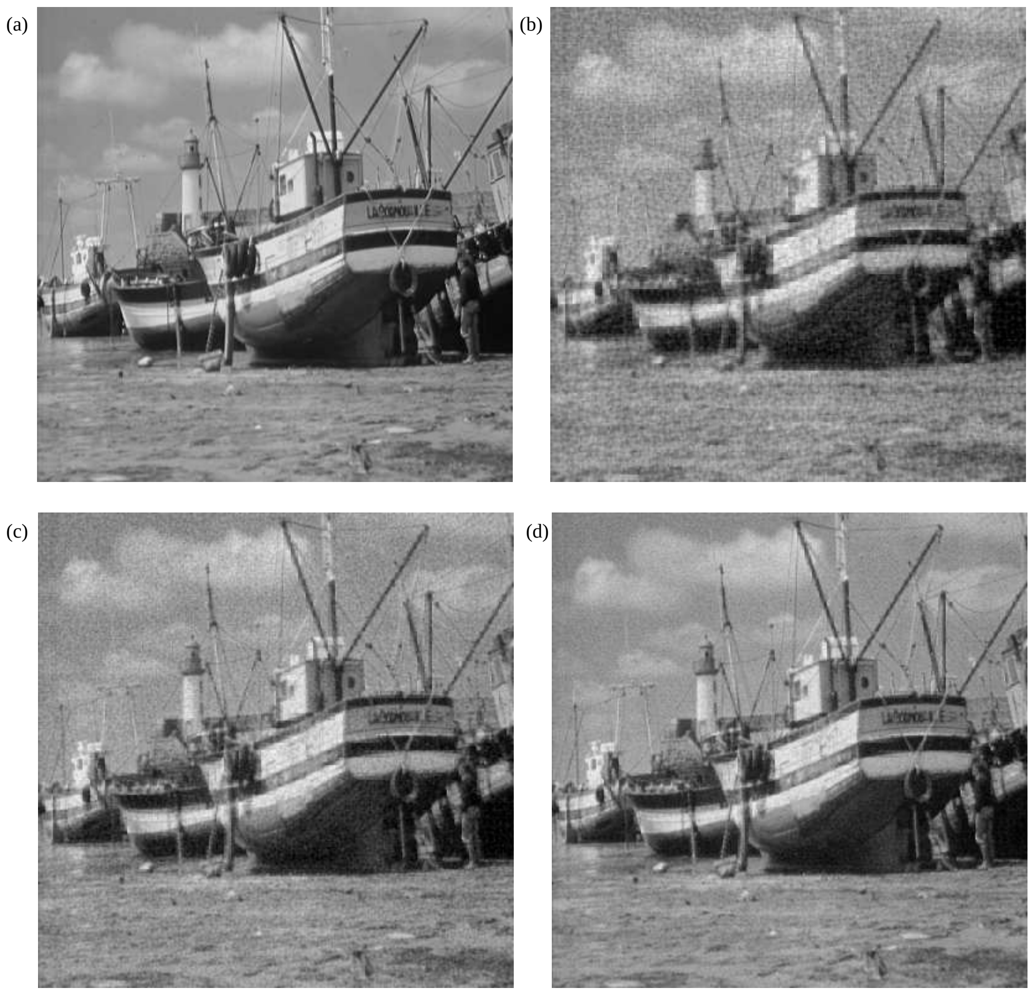


Figure 4.38: Results of the 'boat' image at various bitrates (not part of training set) using the DCT-based block quantiser: (a) original 8-bit image; (b) 0.25 bits/pixel (PSNR=24.06 dB); (c) 0.5 bits/pixel (PSNR=25.76 dB); (d) 1.0 bits/pixel (PSNR=27.88 dB)

Table 4.13: PSNR as a function of number of clusters for the K-means-based multiple transform block quantiser at 0.5 bits/pixel

Image Name	PSNR (in dB)			
	2 clusters	4 clusters	8 clusters	16 clusters
baboon	23.05	23.10	23.13	23.45
barbara	24.15	24.18	24.25	24.55
fruits	27.39	27.43	27.43	28.52
lena	29.01	28.83	28.72	29.34
peppers	28.75	28.51	28.49	29.14
sailboat	26.83	27.09	27.19	27.79
man	25.17	25.18	25.23	25.66
goldhill	29.07	28.96	28.94	29.18
bridge	25.04	25.06	25.04	25.42
jet	27.06	27.28	27.54	28.65
pyramid	28.95	28.76	28.70	29.14
aero	28.53	28.77	28.82	29.12
einstein	30.82	30.30	30.04	30.62
hat	28.38	28.39	28.41	29.24
london	29.85	29.71	29.91	30.00
tekboat	21.19	21.28	21.27	22.10
tekrose	19.14	19.28	19.29	20.06
loco	22.69	22.75	22.77	23.69
boat	27.00	26.95	26.88	27.40
kids	23.77	23.76	23.76	24.12
crowd	19.11	19.32	19.33	20.35
couple	31.34	30.78	30.72	31.11
mill	23.10	23.19	23.22	23.77
vegas	29.34	29.36	29.52	29.96

Table 4.14: Levels allocation table for cluster 1 of 4 cluster K-means-based multiple transform block quantiser at 0.5 bits/pixel

15	7	7	4	4	4	3	3
2	2	2	2	2	2	2	2
2	2	2	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1

at 0.5 bits/pixel. Comparing with the 0.5 bits/pixel column of Table 4.5, we can say that using multiple transforms has resulted in an improvement in PSNR. However, there is no consistent trend as the number of clusters are increased. For most of the images, the PSNR increases as we use more clusters. However, for some images (such as 'lena', 'peppers', 'goldhill', etc.), the PSNR decreases slightly as we use more clusters and then increases again. Thus we can infer that there are some regions produced by the K-means algorithm that are not quantised optimally by the local KLT-based block quantisers. An important factor that affects the performance of the KLT-based block quantiser is the PDF of the input vectors. As we have previously mentioned, the KLT always produces independent coefficients when the input vectors are produced by a correlated Gaussian source. These independent coefficients will also be Gaussian and hence the Lloyd-Max scalar quantisers will code them efficiently. Because the K-means algorithm classifies vector space based on Euclidean distance only, then there will be no guarantee that the vectors within each Voronoi region are Gaussian. This explains why for a certain number of clusters, the performance of the block quantisers decreases, because the new Voronoi regions are not well suited to be quantised. Hence, there arises the need for the joint design of the transform and quantiser in the adaptive transform coding [13, 40, 43]. However, as we have mentioned before, the advantage of the K-means-based multiple transform block quantiser is bitrate scalability, thus it is favourable to use predefined quantisers such as Lloyd-Max that are known by both the encoder and decoder.

Figures 4.39 and 4.40 show the original and compressed versions of the images 'goldhill' and 'boat', respectively. In both Figures, there is a noticeable improvement in visual quality as we use more clusters. That is, the images become cleaner and have less granular noise in the smooth regions. Tables 4.14, 4.15, 4.16 and 4.17 show the level allocations

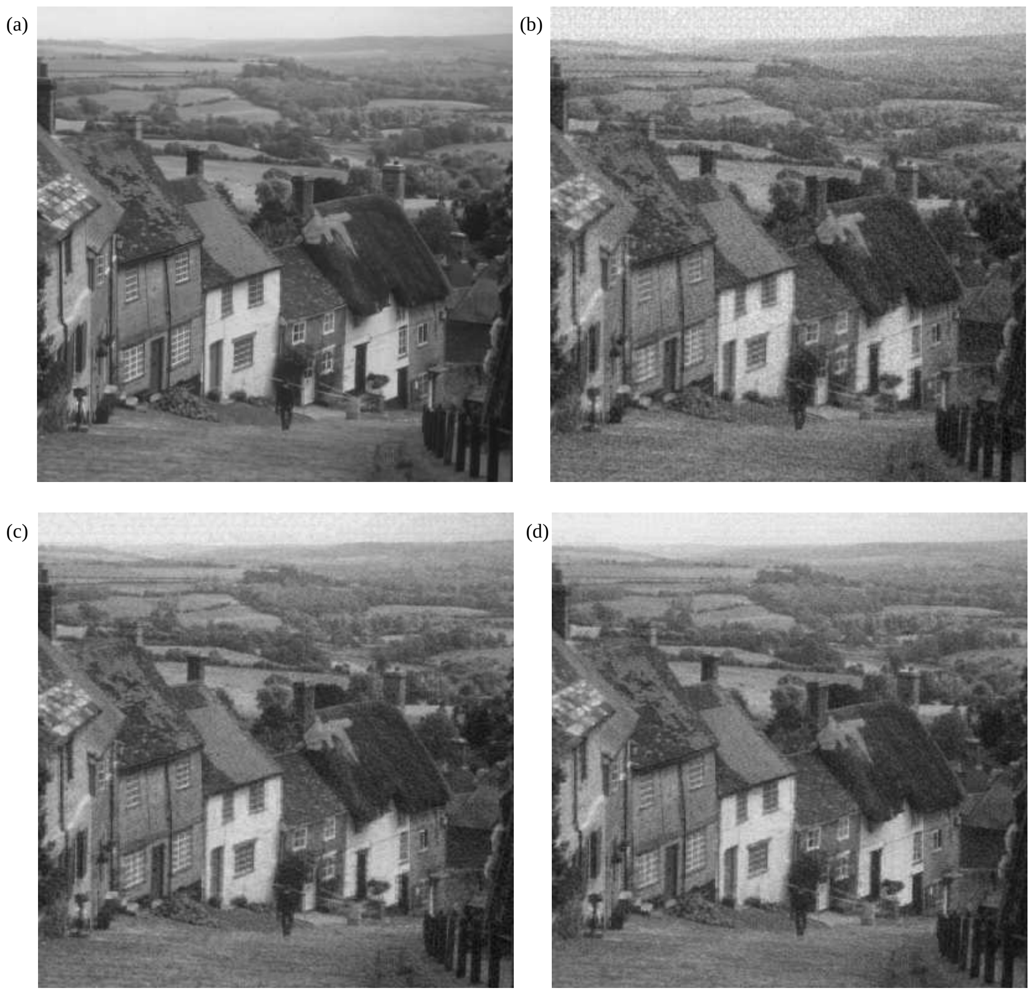


Figure 4.39: Results of the ‘goldhill’ image (part of training set) using the K-means-based multiple transform block quantiser with varying number of clusters at 0.5 bits/pixel: (a) original 8-bit image; (b) 1 cluster (PSNR=28.08 dB); (c) 4 clusters (PSNR=28.96 dB); (d) 16 clusters (PSNR=29.18 dB)

Table 4.15: Levels allocation table for cluster 2 of 4 cluster K-means-based multiple transform block quantiser at 0.5 bits/pixel

11	10	9	5	4	4	3	3
3	3	3	2	2	2	2	2
2	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1

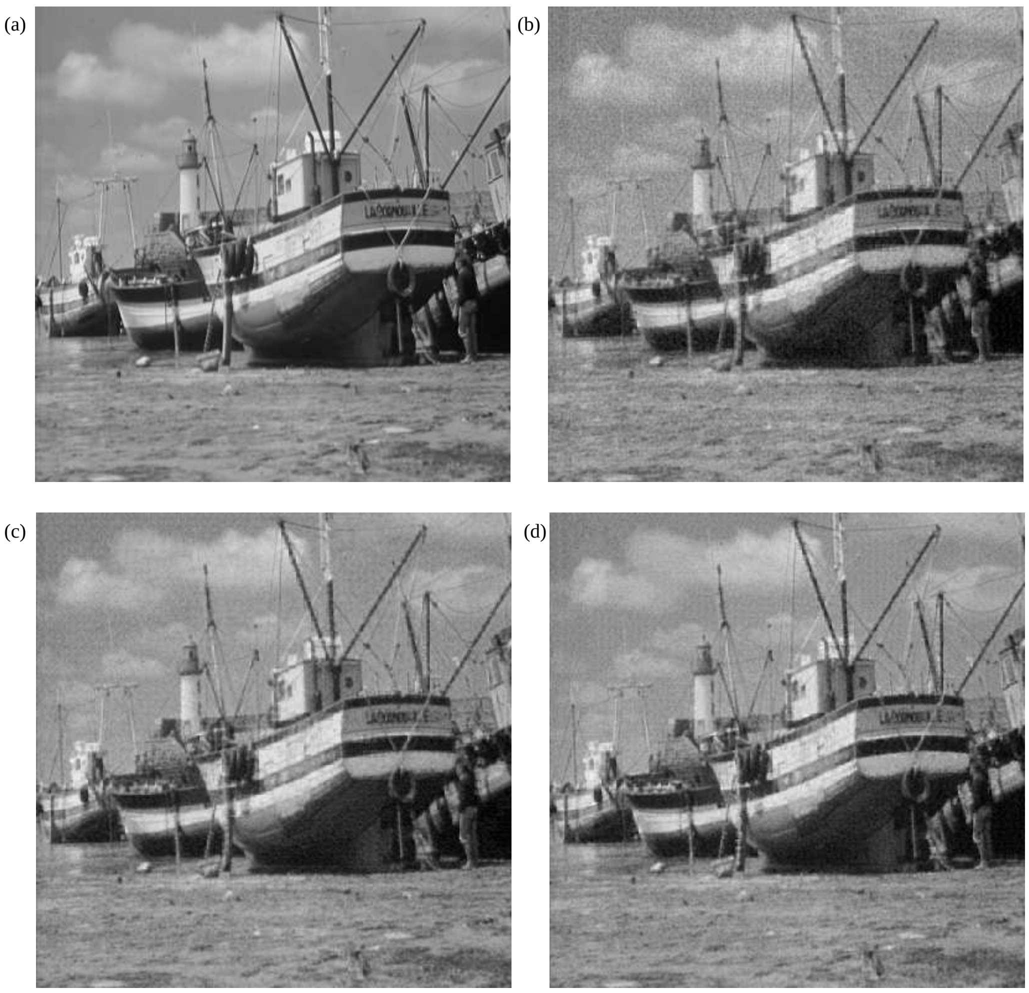


Figure 4.40: Results of the ‘boat’ image at various bitrates (not part of training set) using the K-means-based multiple transform block quantiser at 0.5 bits/pixel: (a) original 8-bit image; (b) 1 cluster (PSNR=26.02 dB); (c) 4 clusters (PSNR=26.95 dB); (d) 16 clusters (PSNR=27.40 dB)

Table 4.16: Levels allocation table for cluster 3 of 4 cluster K-means-based multiple transform block quantiser at 0.5 bits/pixel

10	10	9	5	4	4	3	3
3	3	2	2	2	2	2	2
2	2	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1

Table 4.17: Levels allocation table for cluster 4 of 4 cluster K-means-based multiple transform block quantiser at 0.5 bits/pixel

12	6	6	5	5	4	3	3
3	3	2	2	2	2	2	2
2	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1

for the block quantiser of each cluster. Because the allocation is performed on the basis of variance, we can see that the vectors belonging to each cluster have slightly different statistics. Notice, however, that there is not that much variation in the quantiser level allocations. That is, the low and high frequency transform coefficients for each cluster are allocated a similar proportion of the bit budget.

4.8 Application of the GMM-Based Block Quantiser

In this section, we evaluate the GMM-based block quantiser in an image coding scenario using two different transforms. The KLT-based scheme is expected to perform the best while the DCT-based one should achieve slightly worse performance, though the benefits of source modelling using a GMM should be present. The GMM-DCT-based block quantiser has a lower computational complexity and this will be informally compared via computation times. The GMM-based block quantiser will be compared with the traditional block quantiser and the K-means-based multiple transform block quantiser. Compared with the former, the differences are the use of a GMM for representing the source PDF in addition to multiple KLTs. Compared with the latter, the differences are the *a priori* assumption of Gaussian cluster PDFs and the use of the EM algorithm after the initial K-means clustering process.

4.8.1 The GMM-Based Block Quantiser

Table 4.18 shows the PSNRs as a function of the number of clusters for the GMM-based block quantiser at 0.5 bits/pixel. Compared to the performance of the traditional block

Table 4.18: PSNR as a function of number of clusters for the GMM-based block quantiser at 0.5 bits/pixel

Image Name	PSNR (in dB)			
	2 clusters	4 clusters	8 clusters	16 clusters
baboon	23.31	23.45	23.64	23.97
barbara	24.39	24.68	25.24	26.06
fruits	28.73	29.22	29.66	30.17
lena	30.35	30.90	31.27	31.86
peppers	30.19	30.58	31.19	31.89
sailboat	27.89	28.11	28.30	28.70
man	25.81	26.00	26.33	26.66
goldhill	29.66	29.96	30.15	30.53
bridge	25.34	25.39	25.59	25.82
jet	28.50	28.78	29.14	29.92
pyramid	30.28	30.61	30.89	31.32
aero	29.48	29.78	29.99	30.32
einstein	32.92	33.58	33.87	34.21
hat	29.67	30.25	30.66	31.26
london	30.64	31.11	31.47	31.95
tekboat	21.99	22.28	22.56	22.99
tekrose	20.03	20.32	20.55	20.75
loco	23.68	24.05	24.44	24.86
boat	27.81	28.11	28.54	29.03
kids	24.28	24.45	24.72	24.98
crowd	20.19	20.50	20.82	21.15
couple	34.55	36.52	36.89	37.47
mill	23.64	23.85	24.23	24.64
vegas	30.19	30.70	30.96	31.34

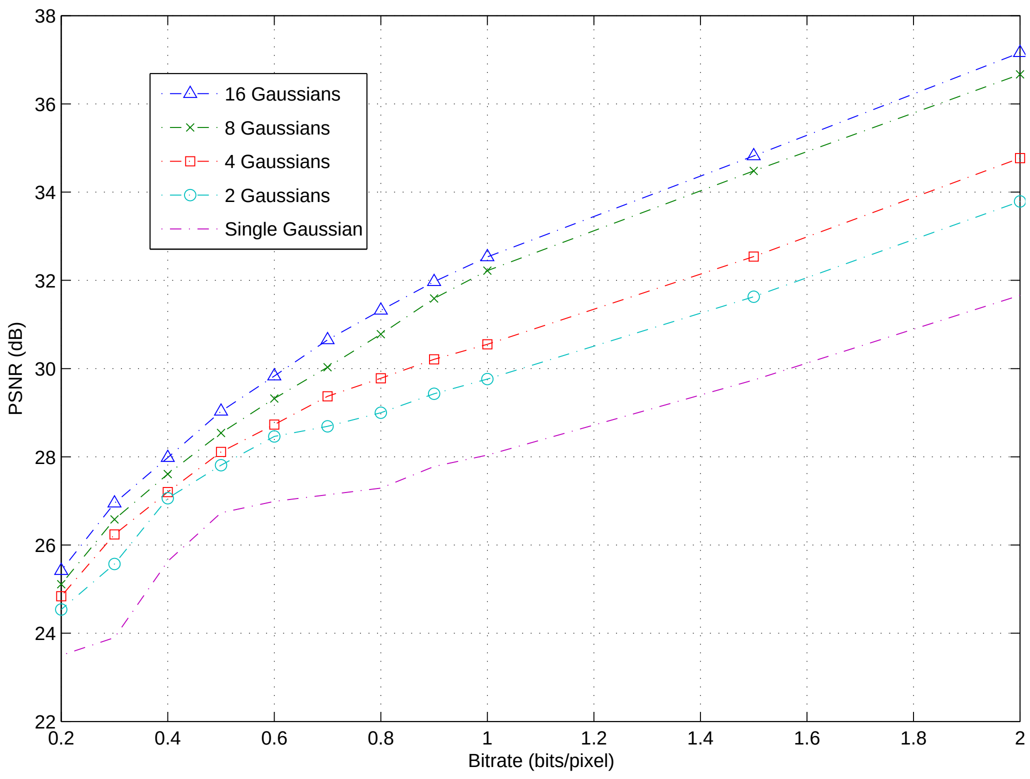


Figure 4.41: Plot of PSNR as a function of bitrate of GMM-based block quantisation of 'boat' with varying number of clusters

quantiser in Table 4.5, which uses a single KLT and assumes the source PDF to be Gaussian, we can see that the GMM-based block quantiser achieves higher PSNRs—by as much as 4 dB on the 'vegas' image[12]. Furthermore, as we increase the number of clusters, the PSNR always improves, unlike the inconsistent K-means-based multiple transform block quantiser of the previous section. That is, going from 2 clusters to 16 clusters results in a PSNR increase of 0.6 dB to as much as 2 dB. This may be attributed to the better estimation of the source PDF by a GMM that has more clusters. Figure 4.41 shows the PSNR as a function of bitrate and number of clusters for the image 'boat'. As we can see, increasing the number of clusters improves the PSNR for all bitrates.

Comparing these results with those of the K-means multiple transform block quantiser in Table 4.13, the GMM-based block quantiser always achieves higher PSNRs. This is quite an interesting observation because both quantisation schemes are essentially similar, apart from the *a priori* assumption of Gaussian distributed clusters, which leads to the application of the EM algorithm, as well as the resultant GMM clusters with soft

[12]Note that the 'couple' image has shown a much larger 8 dB improvement in PSNR. It would appear that this is an outlier since the other PSNR improvements are not of the same magnitude.

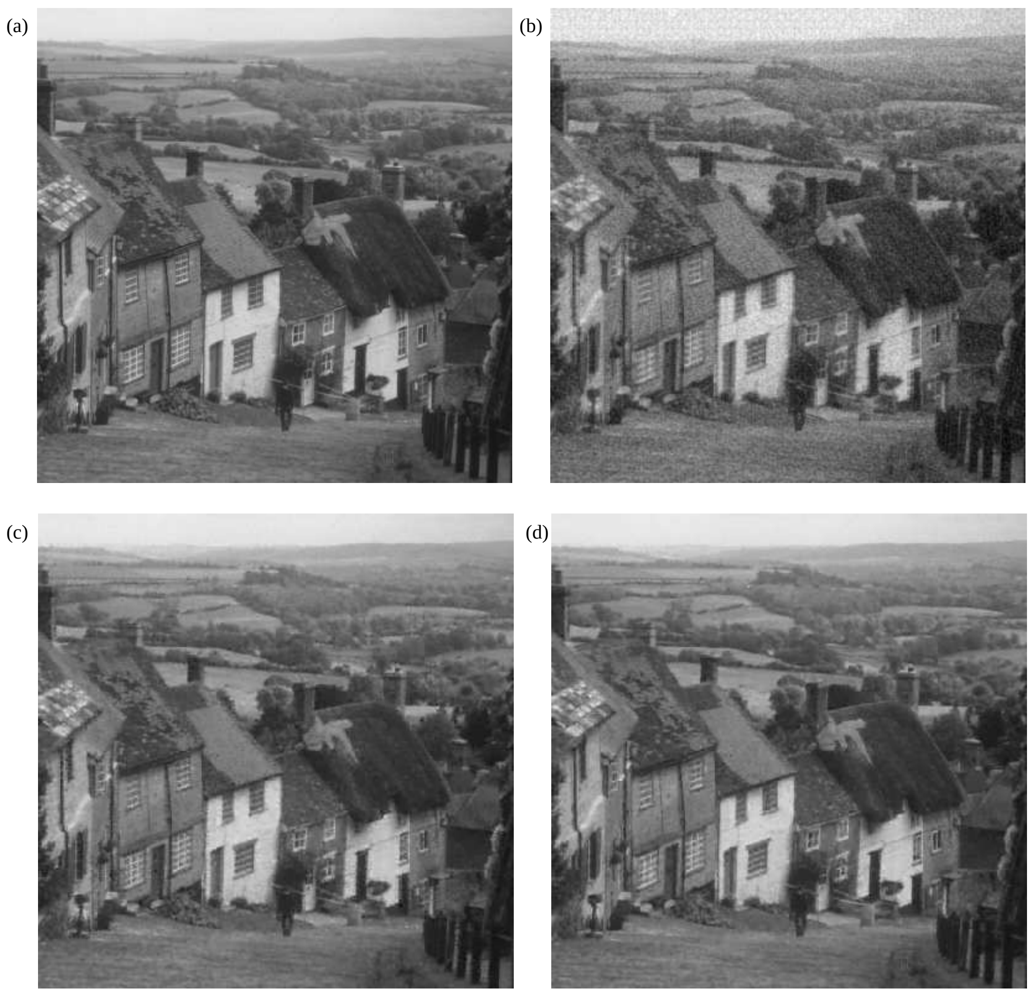


Figure 4.42: Results of the 'goldhill' image (part of training set) using the GMM-based block quantiser with varying number of clusters at 0.5 bits/pixel: (a) original 8-bit image; (b) 1 cluster (PSNR=28.08 dB); (c) 4 clusters (PSNR=29.96 dB); (d) 16 clusters (PSNR=30.53 dB)

boundaries, based on minimum quantiser distortion. It would appear that the individual cluster block quantisers inside the GMM framework are operating more efficiently, in a rate-distortion sense, than those in the K-means framework. As we have discussed previously, in the GMM framework (with hidden data), each vector is assumed to be generated by one of the Gaussian processes, governed by the mixture probabilities. Based on this view, each cluster block quantiser is operating on vectors that are generated by a specific Gaussian source, which satisfies the MSE optimality condition of the KLT. On the other hand, the block quantisers in the K-means framework operate on vectors that are under no assumption of having been generated by a Gaussian source but instead, are classified together based on the minimum unweighted Euclidean distance.

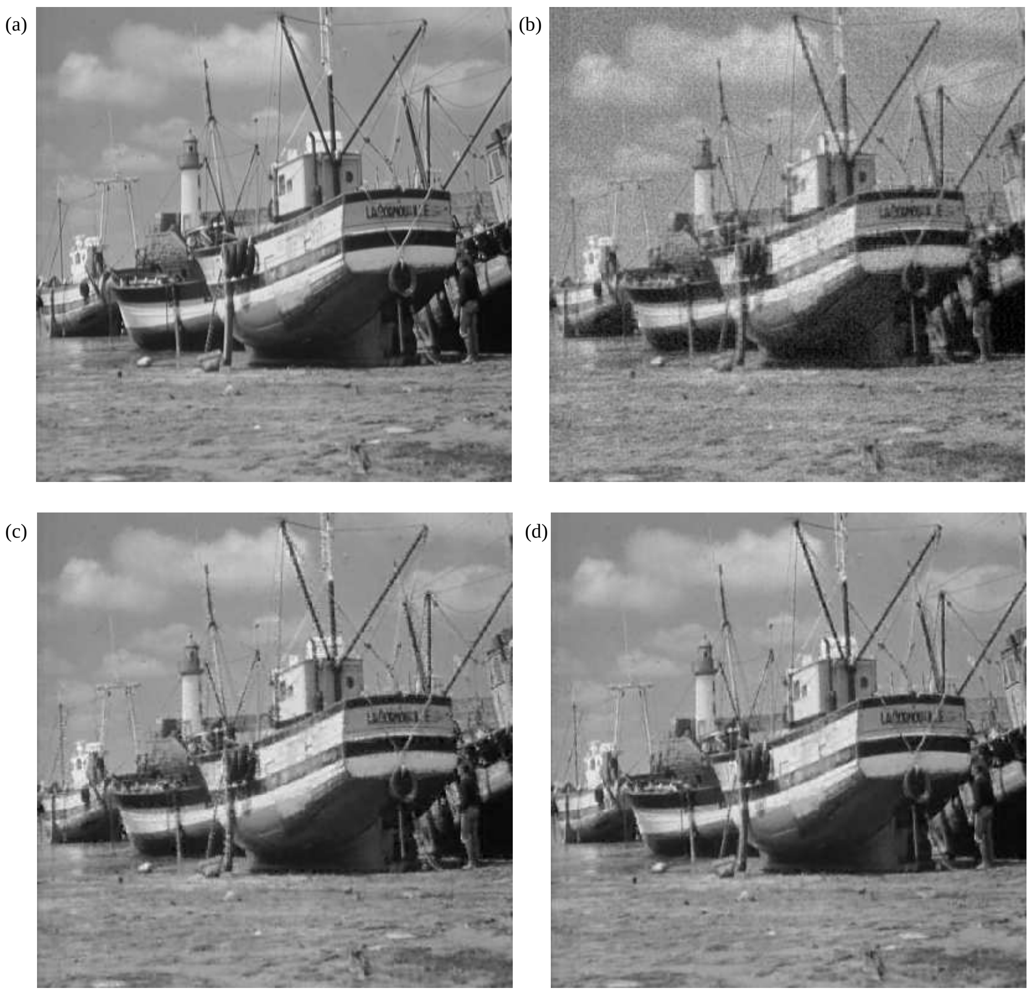


Figure 4.43: Results of the 'boat' image at various bitrates (not part of training set) using the GMM-based block quantiser at 0.5 bits/pixel: (a) original 8-bit image; (b) 1 cluster (PSNR=26.02 dB); (c) 4 clusters (PSNR=28.11 dB); (d) 16 clusters (PSNR=29.03 dB)

Figures 4.42 and 4.43 show the original and compressed versions of 'goldhill' and 'boat' using the GMM-based block quantiser at 0.5 bits/pixel, respectively. The 1 cluster GMM-based block quantiser is essentially the same as the traditional block quantiser. Comparing Figures 4.42(c) and (d) with (b), we can see a noticeable improvement in visual quality over the traditional block quantiser. There is considerably reduced graininess and less block artifacts. Comparing the 4 cluster and 16 cluster GMM-based block quantiser in Figures 4.42(c) and (d), respectively, there is some noticeable granular noise on the edge between the white walled house in the centre of the picture and the grey walled house to its left, which does not appear in the 16 cluster version.

Looking at Figure 4.43, the granular noise observed with traditional block quantisation is considerably reduced in the GMM-based block quantised versions, especially in the sky. In the 4 cluster scheme (Figure 4.43(c)), there are some block artifacts on the diagonal masts of the boat which are reduced in the 16 cluster scheme. These improvements can be explained by looking at the quantiser level allocation tables (Tables 4.19, 4.20, 4.21 and 4.22) of the 4 cluster GMM-based block quantiser. In cluster 4, the first component has been given 256 levels or 8 bits while the other clusters have been allocated much less. Assuming each component to be related to frequency, we can say that cluster 4 has allocated more of the bit budget to low frequencies (smooth regions) while in clusters 1, 2, and 3, the bit budget is spread to higher frequency components. Because edges cause energy to be spread throughout the transform domain, then images with a large amount of edges are expected to require more bits in the high frequencies. In order to see this effect, Table 4.23 shows the number of vectors per cluster block quantiser, for three images. The 'fruits' image contains a lot of smooth regions while the images, 'barbara' and 'baboon' tend to contain more edges. We can see that more blocks were quantised using cluster 4 for the 'fruits' image than the other clusters while in the other two images, a higher proportion of blocks were quantised by clusters 1, 2 and 3. Therefore, we can say that the GMM-based block quantiser has effectively classified the image blocks based on image activity and designed appropriate KLTs and bit allocations for each class.

We mentioned previously that the cluster allocations for the K-means-based scheme showed little variation or adaptivity among the clusters. For example, the levels allocated to the first transform coefficient of each cluster were very similar (15, 11, 10, 12). For the GMM-based scheme, there is quite a lot of variation and adaptivity in the cluster

Table 4.19: Levels allocation table for cluster 1 of 4 cluster GMM-based block quantiser at 0.5 bits/pixel

81	12	11	5	4	4	3	3
2	2	2	2	2	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1

Table 4.20: Levels allocation table for cluster 2 of 4 cluster GMM-based block quantiser at 0.5 bits/pixel

17	7	7	4	4	4	3	3
3	3	3	2	2	2	2	2
2	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1

allocations (81, 17, 21, 256). If we view the clustering process as a classification of image blocks, then it would appear that the K-means-based scheme produces classes which are 'close' to each other and have similar statistics, whereas the GMM-based scheme produces classes that are 'far' apart and have different statistics. Because images are non-stationary, it is desirable for an adaptive transform coding scheme to capture most of the variability. If we compare Figures 4.42 and 4.43 with 4.39 and 4.40, respectively, we can see that the quality of the 16 cluster GMM-based block quantised images is better than the 16 cluster K-means multiple transform block quantiser. There is less granular noise in the GMM-based scheme because there are clusters which have more quantiser levels assigned in the high frequency components compared with those of the K-means-based multiple transform block quantiser, which have less variation in their quantiser level allocations.

Table 4.21: Levels allocation table for cluster 3 of 4 cluster GMM-based block quantiser at 0.5 bits/pixel

21	7	6	4	4	4	3	3
3	2	2	2	2	2	2	2
2	2	2	2	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1

Table 4.22: Levels allocation table for cluster 4 of 4 cluster GMM-based block quantiser at 0.5 bits/pixel

256	8	7	3	2	2	2	2
2	2	2	2	2	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1

Table 4.23: Cluster densities for the 4 cluster GMM-based block quantiser at 0.5 bits/pixel

Cluster number	Number of vectors per cluster		
	'fruits'	'barbara'	'baboon'
1	1499	1487	1393
2	327	857	1195
3	256	618	1215
4	2014	1134	193

Table 4.24: PSNR performance of the fixed-rate (2 bits/pixel), 4 cluster GMM-based block quantiser using integer and fractional bit-based cluster block quantisers (underallocation) on images that were part of the training set

Image Name	PSNR (dB)	
	Integer	Fractional
lena	38.32	38.38
Einstein	41.06	41.12
jet	36.37	36.50
goldhill	37.25	37.32
hat	37.38	37.44

Table 4.25: PSNR performance of the fixed-rate (2 bits/pixel), 4 cluster GMM-based block quantiser using integer and fractional bit-based cluster block quantisers (underallocation) on images that were not part of the training set

Image Name	PSNR (dB)	
	Integer	Fractional
boat	34.76	34.81
kids	29.55	29.57
crowd	26.96	27.02
mill	30.48	30.52
vegas	38.49	38.58

Table 4.26: PSNR performance of the fixed-rate (0.15 bits/pixel), 4 cluster GMM-based block quantiser using integer and fractional bit-based cluster block quantisers (overallocation) on images that were part of the training set

Image Name	PSNR (dB)	
	Integer	Fractional
lena	24.81	25.50
Einstein	26.33	27.39
jet	22.60	23.67
goldhill	24.83	25.46
hat	24.08	24.60

Table 4.27: PSNR performance of the fixed-rate (0.15 bits/pixel), 4 cluster GMM-based block quantiser using integer and fractional bit-based cluster block quantisers (overallocation) on images that were not part of the training set

Image Name	PSNR (dB)	
	Integer	Fractional
boat	23.32	23.95
kids	21.52	21.92
crowd	17.09	17.16
mill	20.61	20.71
vegas	24.06	24.35

Table 4.28: Effective bitrates for each cluster at 2 bits/pixel

Cluster	**Target**		**Integer**		**Fractional**	
	Bitrate	Total Bits	Bitrate	Total Bits	Bit Rate	Total Bits
1	1.943308	124.371726	1.9375	124	1.943224	124.366336
2	1.966590	125.861752	1.953125	125	1.966462	125.853568
3	1.991375	127.44803	1.984375	127	1.991121	127.431744
4	1.895964	121.341682	1.890625	121	1.895655	121.32192

4.8.2 Comparison Between Integer and Fractional Bit-Based Cluster Block Quantisers in the GMM-Based Block Quantiser

In this section, we investigate the performance of the GMM-based block quantiser which uses integer and fractional bit-based cluster block quantisers. In the integer bit-based scheme, the number of bits allocated to each cluster is truncated and then further allocated to each of the transform coefficients, where further truncation is performed. If the bitrate is high, then the bit allocation formula will tend to underallocate, due to the truncation. If the bitrate is low, then the bit allocation formula will tend to overallocate, due to the zeroing of negative values.

Tables 4.24 and 4.26 show the PSNR results for training images of both cases of underallocation and overallocation, respectively. It can be seen that a small increase in PSNR has been achieved through better utilisation of the bit budget when using fractional bits rather than with integer bits. The benefits of using fractional bits are more evident at low bitrates. At 0.15 bits/pixel, the results for the images 'Einstein' and 'jet' show a 1 dB improvement in PSNR while other images gain about 0.6 dB which is more significant than those observed at 2 bits/pixel. Tables 4.25 and 4.27 show the PSNR results for images that were not part of the training set. As before, better utilisation of the bit budget has led to a slightly higher PSNR.

These performance improvements may be explained by Table 4.29 which shows the effective bitrate compared with the target bitrate of each cluster block quantiser. In Table 4.29, the total fractional bits for cluster 1 is roughly 5.97 bits. For the integer bit-based cluster block quantiser, this total is truncated to 5 bits (32 levels), hence 0.97 bits are not utilised. While for the fractional bit-based cluster block quantiser, the total equivalent bits is approximately 5.9 bits (60 levels). The increase in the performance may also be due

Table 4.29: Effective bitrates for each cluster at 0.15 bits/pixel

Cluster	Target		Integer		Fractional	
	Bitrate	Total Bits	Bit Rate	Total Bits	Bitrate	Total Bits
1	0.093308	5.971726	0.078125	5	0.092295	5.90688
2	0.116590	7.461752	0.109375	7	0.116554	7.459456
3	0.141375	9.048003	0.140625	9	0.141375	9
4	0.045964	2.941682	0.03125	2	0.043865	2.80736

to the fact that the optimality of the constrained minimisation formula of (2.66) is better preserved in fractional bit encoding. For the integer bit case, fractional parts calculated from the formula are discarded and a heuristic algorithm is then used to compensate the truncated bits. The heuristic algorithm is itself dependent on various assumptions and approximations and may only produce a suboptimal solution. On the other hand, most of the fractions from (2.66) are preserved when converting from bits to the levels since the truncation occurs after the conversion. For example, consider a bitrate of 5.9 bits calculated from (2.66). In the integer bit case, this bitrate would truncated to 5 bits or 32 levels. In the fractional bit case, converting to levels gives $2^{5.9} = 59.714$ and after truncation results in 59 levels. By changing the point at which truncation is performed, an extra 27 quantiser levels are preserved. Therefore, it is expected that fractional bit-based coding allows us to get closer to the values specified by the Lagrangian-minimised formula and have less dependence on the use of heuristics and high resolution approximations.

4.8.3 The GMM-DCT-Based Block Quantiser

In this section, we present the results of our modified scheme which uses a GMM to model the distribution of DCT coefficients. The advantage of this method is speed, due to less transform operations, while maintaining similar quantiser distortion performance as the KLT-based scheme.

Table 4.30 shows the PSNRs as a function of the number of clusters for the GMM-DCT-based block quantiser at 0.5 bits/pixel. Compared to the performance of the traditional block quantiser using the KLT and DCT in Tables 4.5 and 4.9, respectively, we can see that the GMM-DCT-based block quantiser achieves higher PSNRs. Furthermore, as we increase the number of clusters, the PSNR always improves, which is consistent with the notion that more accurate modelling of the PDF using a GMM leads to better quantisation

Table 4.30: PSNR as a function of number of clusters for the GMM-DCT-based block quantiser at 0.5 bits/pixel

Image Name	PSNR (in dB)			
	2 clusters	4 clusters	8 clusters	16 clusters
baboon	23.18	23.25	23.60	23.84
barbara	24.29	24.36	24.80	25.46
fruits	28.61	28.98	29.50	29.92
lena	30.11	30.38	31.03	31.43
peppers	29.83	29.96	31.01	31.60
sailboat	27.68	27.73	28.17	28.46
man	25.67	25.77	26.26	26.48
goldhill	29.47	29.68	30.02	30.40
bridge	25.17	25.18	25.53	25.71
jet	28.15	28.28	29.07	29.53
pyramid	29.99	30.15	30.72	31.06
aero	29.25	29.44	29.74	29.98
einstein	32.77	33.21	33.57	33.93
hat	29.59	29.94	30.42	30.85
london	30.41	30.65	31.44	31.79
tekboat	21.75	21.93	22.50	22.88
tekrose	19.91	20.19	20.44	20.67
loco	23.48	23.75	24.33	24.77
boat	27.57	27.70	28.39	28.73
kids	24.20	24.32	24.67	24.90
crowd	20.00	20.29	20.73	21.04
couple	34.33	35.78	36.27	36.84
mill	23.41	23.53	24.17	24.52
vegas	30.07	30.39	30.68	31.00

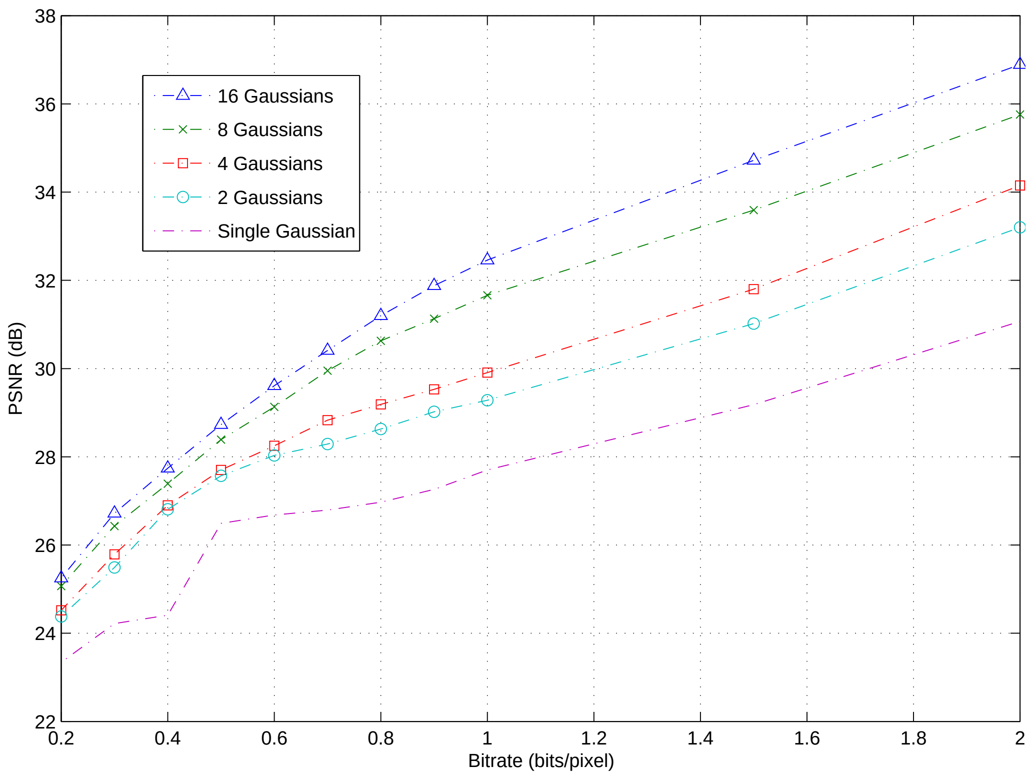


Figure 4.44: Plot of PSNR as a function of bitrate of GMM-DCT-based block quantisation of 'boat' with varying number of clusters

performance. As expected, the GMM-DCT-based block quantiser performs slightly worse though the difference in PSNR is often less than 0.5 dB. Given the fact that using the same fixed transform for each cluster results in such minor degradation in PSNR, we may conclude that, for image coding, the gains in quantisation performance, achieved through GMM-based block quantisation, are mostly due to the source modelling aspect, rather than using locally adaptive decorrelating transforms. This is further demonstrated in Figure 4.44, where we have plotted the PSNR as a function of bitrate and varying number of clusters. We can observe that the PSNR increases as we use more clusters. It is apparent that these gains are mostly due to the accurate PDF modelling, rather than the transform which is fixed.

The improvements to image quality resulting from more accurate modelling of the PDF can be seen in Figure 4.45 where the image 'goldhill' was compressed using traditional DCT block quantisation (single Gaussian) as well as using the improved GMM-DCT-based block quantiser at a bitrate of 0.5 bits/pixel. In Figure 4.45(b), it can be seen that the sky as well as the ground looks very grainy. Also, there is a large degree of blocked distortion

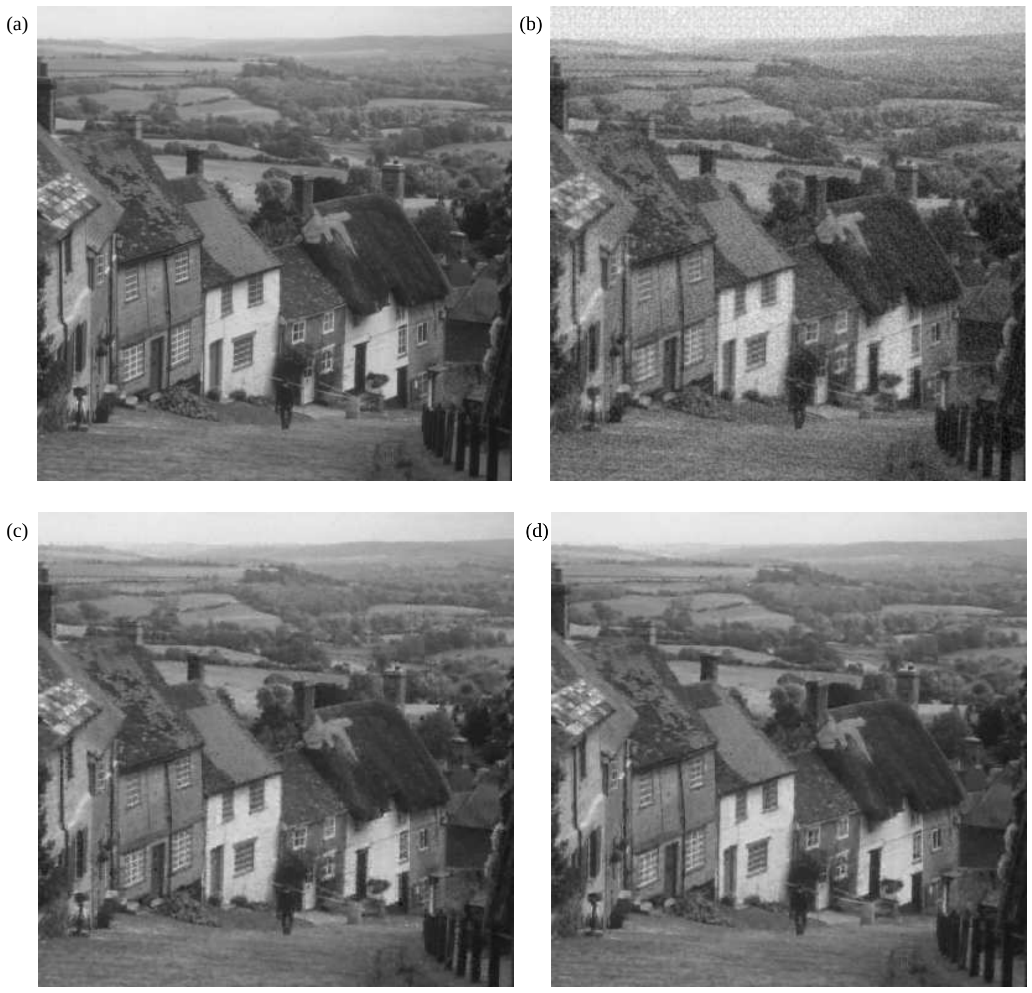


Figure 4.45: Results of the 'goldhill' image (part of training set) using the GMM-DCT-based block quantiser with varying number of clusters at 0.5 bits/pixel: (a) original 8-bit image; (b) 1 cluster (PSNR=27.78 dB); (c) 4 clusters (PSNR=29.68 dB); (d) 16 clusters (PSNR=30.40 dB)

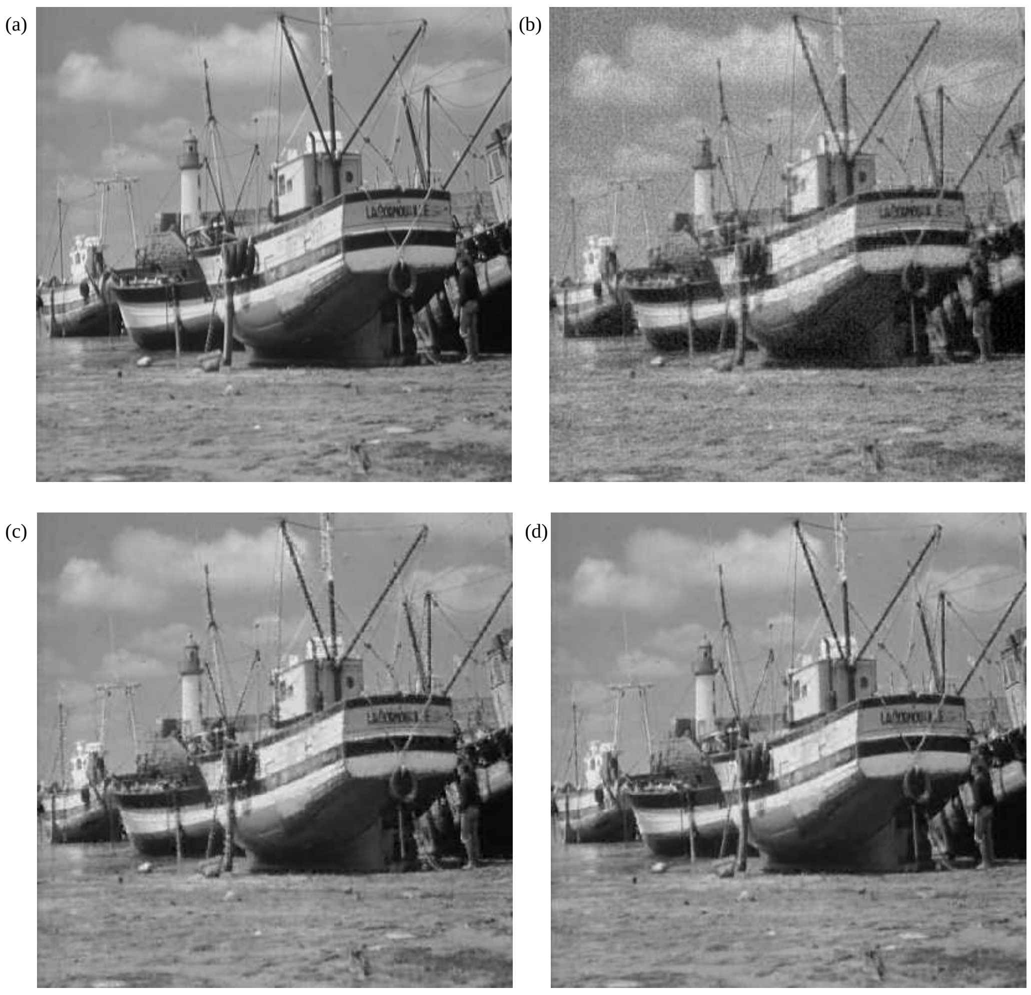


Figure 4.46: Results of the ‘boat’ image at various bitrates (not part of training set) using the GMM-based block quantiser at 0.5 bits/pixel: (a) original 8-bit image; (b) 1 cluster (PSNR=25.76 dB); (c) 4 clusters (PSNR=27.70 dB); (d) 16 clusters (PSNR=28.73 dB)

Table 4.31: Levels allocation table for cluster 1 of 4 cluster GMM-DCT-based block quantiser at 0.5 bits/pixel

86	11	4	2	1	1	1	1
11	4	2	1	1	1	1	1
5	3	2	1	1	1	1	1
3	2	1	1	1	1	1	1
2	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1

Table 4.32: Levels allocation table for cluster 2 of 4 cluster GMM-DCT-based block quantiser at 0.5 bits/pixel

17	7	4	3	2	1	1	1
7	4	3	2	1	1	1	1
4	3	2	1	1	1	1	1
3	2	2	1	1	1	1	1
3	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1

observable in region of the houses, especially on the edges of the white house in the centre of the picture. By using a 4 cluster GMM to model the PDF, one can see in Figure 4.45(c) that the sky and ground area appear much smoother. There is some granular distortion near the edge of the white house in the centre of the picture and the grey house to its left. In Figure 4.45(d), where a 16 cluster GMM is used, the sky and ground are considerably smoother and less noisy and distortions in the fields has been considerably reduced. Similarly, Figure 4.46 shows the image ‘boat’ which is not part of the training set. In Figure 4.46(b), where a traditional DCT-based block quantiser was used, the smooth areas such as the sky and the black bottom of the boat are very grainy. There is also block distortion on the white sides of the boat. It can be observed that in Figures 4.46(c) and (d), as more clusters are used, the graininess of the smooth regions has been reduced as well as the block distortions. Also, the block artifacts on the diagonal masts of the boat are reduced as we increase the number of clusters.

Tables 4.31, 4.32, 4.33 and 4.34 show the quantiser level allocations of a 4 cluster GMM-DCT-based block quantiser. We can see that the DC coefficient of cluster 4 has been allocated 256 levels (8 bits), thus it is adapted for smooth, slow changing regions, while

Table 4.33: Levels allocation table for cluster 3 of 4 cluster GMM-DCT-based block quantiser at 0.5 bits/pixel

21	6	4	2	2	1	1	1
7	4	3	2	2	1	1	1
4	3	2	2	1	1	1	1
3	2	2	1	1	1	1	1
2	2	1	1	1	1	1	1
2	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1

Table 4.34: Levels allocation table for cluster 4 of 4 cluster GMM-DCT-based block quantiser at 0.5 bits/pixel

256	7	2	2	1	1	1	1
7	2	2	1	1	1	1	1
3	2	2	1	1	1	1	1
2	2	1	1	1	1	1	1
2	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1

the other clusters (particularly, cluster 2 and 3) have more quantiser levels allocated to the higher frequency coefficients, which capture most of the energy of fast changing regions such as edges and textures. To see the distributions of these clusters, Table 4.35 shows the cluster densities of the 4 cluster GMM-DCT-based block quantiser at 0.5 bits/pixel. For the image 'fruits', where there are comparatively more smooth regions, a large proportion of image blocks have been quantised by cluster 1 and 4, which have more levels allocated to the DC coefficient. For the other images, especially 'baboon', which contains a large amount of edge activity, a large proportion of image blocks have been quantised using clusters 1, 2, and 3 with cluster 4 accounting for less than 5% of the total image blocks.

At the expense of slightly degraded performance, using the DCT results in less computations. Table 4.36 shows the average time, in seconds, it takes to quantise 19 test images at a bitrate of 1 bit per pixel. In total, 77824 image blocks were quantised on an Intel Pentium 4 system running at 2.4 GHz. The number of clusters was increased (2, 4, 8, 16) and the performance times are averaged. It can be seen that the DCT version of the coder is considerably faster than the KLT-based one. It is expected that the differences in time would be even more if better optimised DCT algorithms were used.

Table 4.35: Cluster densities for the 4 cluster GMM-DCT-based block quantiser at 0.5 bits/pixel

Cluster number	Number of vectors per cluster		
	'fruits'	'barbara'	'baboon'
1	1534	1529	1376
2	343	1054	1285
3	257	441	1256
4	1962	1081	179

Table 4.36: Comparison of average processing times between the GMM-KLT and GMM-DCT-based block quantiser (in seconds)

No. of clusters	GMM-DCT	GMM-KLT
2	6.5	14.8
4	8.6	30.6
8	13.2	59.5
16	21.5	120

4.9 Reducing Block Artifacts in the GMM-Based Block Quantiser Using Wavelet Pre-Processing

One of the disadvantages of block-based coding schemes, such as transform coders and vector quantisers, is the presence of block artifacts in the reconstructed image. Popular methods for reducing block artifacts include using block overlapping, post-filtering [145], and the lapped orthogonal transform (LOT) [110]. Subband and wavelet transform-based image coders do not suffer from this problem since the transformation operates on the entire image rather than on individual blocks [20]. In this section, we investigate the application of the GMM-based block quantiser on image wavelet coefficients, as an alternative method for reducing the block artifacts in the reconstructed image. In this framework, the discrete wavelet transform may be viewed as a pre-processing step before quantisation.

The first issue is extracting the vectors or blocks from the wavelet transform of an image. Most subband decomposition and wavelet transform-based coding schemes employ different quantisers for each subimage. For example, in the wavelet coder of Antonini *et al.* [7], the lowest subimage is quantised using a scalar quantiser, while the higher subbands are vector quantised at varying bitrates with different resolution codebooks. In this respect, we

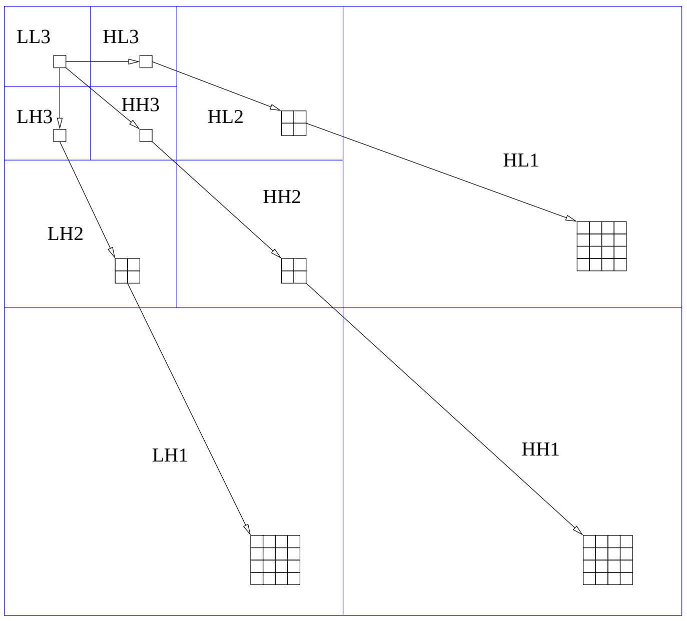


Figure 4.47: Wavelet tree structure for extracting vectors from the same spatial location (after [158]). The wavelet coefficients shown above are concatenated to form one block or vector for quantisation.

can quantise the lowest subimage using the GMM-based block quantiser, since it contains a large amount of correlation while the other higher subimages can be quantised using scalar quantisers that are PDF optimised for a Laplacian distribution [142]. However, the bit budget needs to be allocated (often arbitrarily) to each of the individual quantisation schemes which can complicate the design. Also, artifacts in the lowest (LL) subimage tend to have a much larger impact on the reconstructed image because it contains the most variance or energy, which explains why high bitrate scalar quantisation schemes are often used, such as the one in [7], since block-based quantisers introduce block artifacts.

Therefore, we have chosen to adopt the hierarchical tree structure, first introduced by Lewis and Knowles [98] and used by Shapiro [158] for representing zerotrees, where each coefficient in the low resolution subimages have descendants in the same spatial location in the upper (higher resolution) subimages. For simplicity, we concatenate the wavelet trees in all three directions into a single vector. Figure 4.47 shows the coefficients of three wavelet trees branching from a common node in the lowest subimage. Vectors are formed in the following way: [LL3 LH3 HL3 HH3 LH2 HL2 HH2 LH1 HL1 HH1]. We first make a few observations of the vectors derived from these wavelet trees.

Firstly, the discrete wavelet transform has an energy compaction property that is similar to the KLT. That is, most of the image energy is concentrated into a few coefficients in the lowest subimage while the higher subimages contain sparse coefficients. The wavelet tree vectors have their energy compacted into the first coefficient. Secondly, the quantisation distortion in the lowest subimage is limited to individual scalars while the block size is dyadically increased as we go to higher and higher subimages. This prevents block artifacts from appearing in the lowest subimage which may have a large impact on the final reconstructed image. Lastly, it has been proposed by Shapiro [158], that the coefficients within these wavelet trees, are correlated to a degree. That is, if a coefficient is 'insignificant' (less than a predetermined threshold), then there is a likelihood that its descendants will be insignificant as well. Because these wavelet trees tend to group together wavelet coefficients exhibiting self-similarity in the higher subimages, then they appear to be good candidates for block quantisation as the decorrelation allows the removal of redundancy.

It is therefore expected that block artifacts, as a result of the block quantisation, are reduced in this modified scheme since the most sensitive subimage (the lowest subimage) is effectively scalar quantised while the block size is gradually increased as we go up to finer resolutions. Also, the estimation of the PDF of coefficients in the lowest subimage by the GMM will increase the efficiency of scalar quantisation. Furthermore, the bit allocation to the components can be performed easily using closed-form expressions rather than arbitrarily or heuristically. The disadvantage of this scheme is that correlation within the lowest subimage is not exploited by the GMM-based block quantiser. It is expected that the predictive GMM-based block quantiser will do better, in this regard. Also, the higher subimages typically have Laplacian PDFs, which may not be well suited for GMM estimation. Finally, the wavelet transform (and its inverse) incurs further computational complexity on the GMM-based block quantiser. Therefore, it is not expected that this scheme will be competitive with state-of-the-art wavelet coders such as EZW and SPIHT. What it will highlight, however, is that a pre-processing of the image data using the discrete wavelet transform followed by a unique block extraction, that takes into account spatial locality [98] and interband correlation, leads to the reduction of block artifacts, which are typical of transform coding schemes. It is expected that the block artifacts will be more or less replaced by some edge ringing and smoothing of fine detail, which are typical of wavelet image coders.

Table 4.37: PSNR of the 16 cluster GMM-based block quantiser at 0.5 bits/pixel with and without the wavelet transform pre-processing

Image Name	PSNR (in dB) at 0.5 bits/pixel	
	With wavelet transform	Without wavelet transform
baboon	24.07	23.97
barbara	26.75	26.06
fruits	30.62	30.17
lena	32.62	31.86
peppers	32.54	31.89
sailboat	29.27	28.70
man	27.09	26.66
goldhill	31.06	30.53
bridge	26.13	25.82
jet	30.22	29.92
pyramid	31.78	31.32
aero	31.22	30.32
einstein	34.78	34.21
hat	32.67	31.26
london	32.38	31.95
tekboat	23.33	22.99
tekrose	21.00	20.75
loco	25.11	24.86
boat	29.59	29.03
kids	25.14	24.98
crowd	21.57	21.15
couple	39.11	37.47
mill	24.83	24.64
vegas	32.34	31.34

4.9.1 Experimental Setup

In our experiments, we perform a three level discrete wavelet transform using the 9/7-tap biorthogonal wavelets from [7]. Symmetric extension is used when filtering the boundaries of the image. The vectors formed from the wavelet trees have a dimension of 64, as shown in Figure 4.47. Therefore, the effective number of vectors or blocks is the same as the GMM-based block quantiser using 8×8 spatial blocks. We have used 20 iterations of the EM algorithm to estimate a 16 cluster GMM.

4.9.2 Results and Discussion

Table 4.37 shows the PSNRs of all images in the training and testing set at a bitrate of 0.5 bits/pixel for the GMM-based block quantiser with and without the wavelet transform. Note that the PSNRs of the latter case are the same as those in the last column of Table 4.18. We observe that there is a finite PSNR improvement of up to 1 dB[13], when we apply the wavelet transform pre-processing step.

In order to see the visual quality, Figures 4.48, 4.49, and 4.50 show a comparison between the reconstructed images from the GMM-based block quantiser with and without the wavelet transform. We can see that, particularly in the zoomed up regions, the modified scheme has reduced the block artifacts. However there is some degree of smoothing of detail and ringing, particularly in the image 'boat' on the diagonal masts of the boat. These are typical artifacts observed in subband and wavelet coders [142].

In order to see the effect of quantisation, Figure 4.51 shows the original wavelet transforms of three images, 'man', 'vegas', and 'barbara', as well as the quantised versions at 0.5 bits/pixel. In these images, we can see that our modified scheme preserves important detail that occurs in the same relative spatial location of the high frequency subimages. Also, it appears that details in the highest frequency subimages have been discarded by the scheme, which is similar to what is normally done in other wavelet coders such as [7].

In summary, using the wavelet transform as a pre-processing step has reduced block artifacts in the GMM-based block quantisation of images. Furthermore, at the same bitrate and vector dimensionality, the addition of wavelet transform pre-processing has allowed the GMM-based block quantiser to achieve higher PSNRs.

4.10 Chapter Summary

In this chapter, we have presented a comprehensive literature review of image coding techniques, which includes vector quantisation, transform coding, and subband and wavelet-based coding. Fundamental to image subband coding and image processing in general, is the non-expansive filtering of data with a finite length. Similar to the procedure given in [106], the symmetric extension method was examined in depth with examples provided for

[13]The image 'couple' has shown a much larger increase in PSNR than the other images.

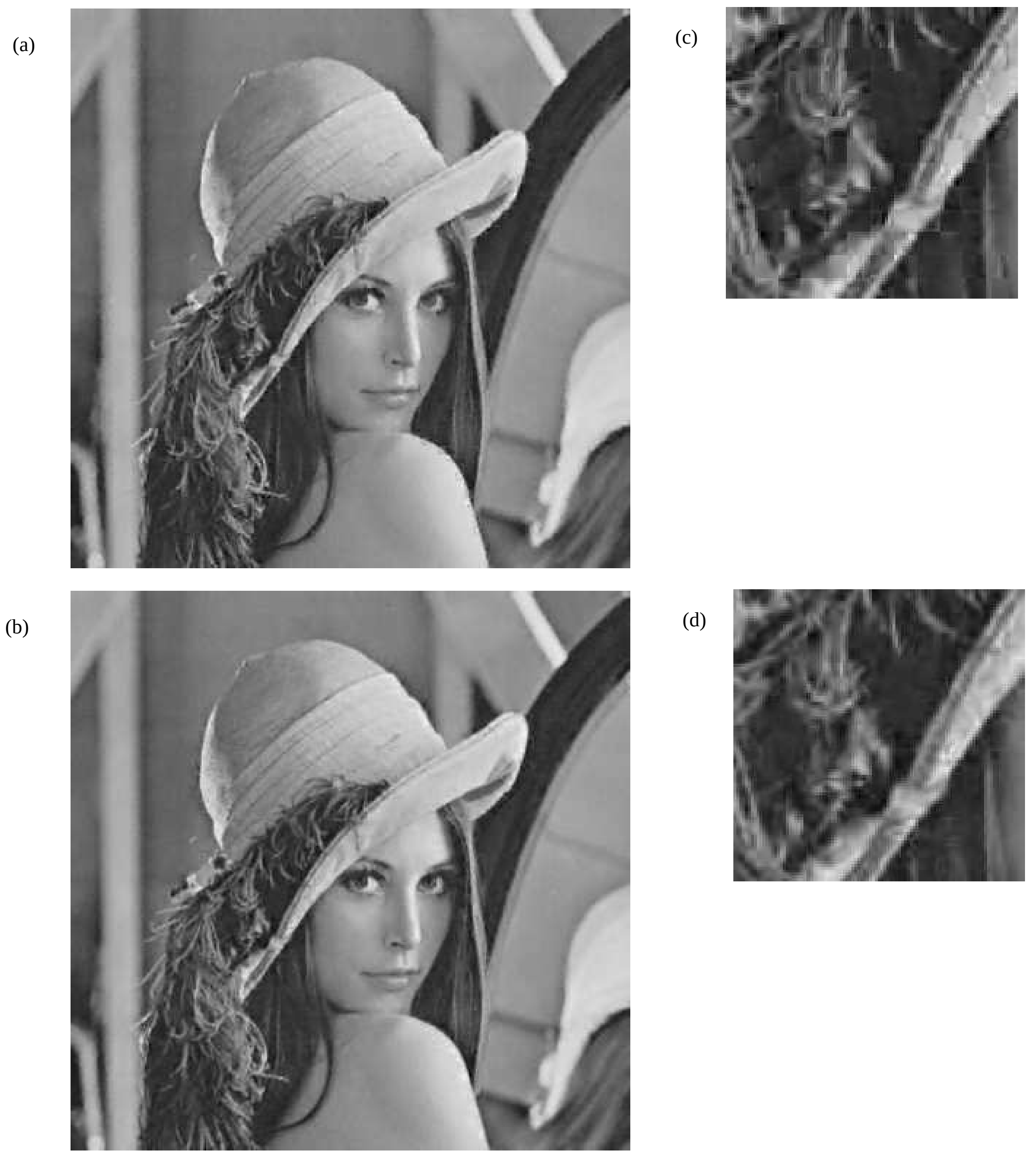
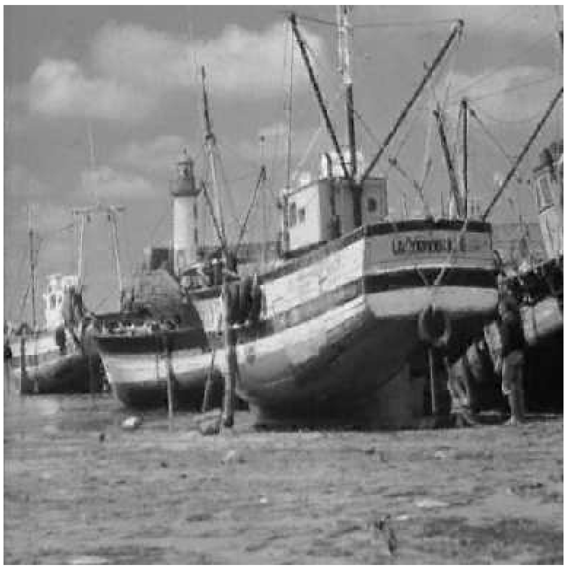
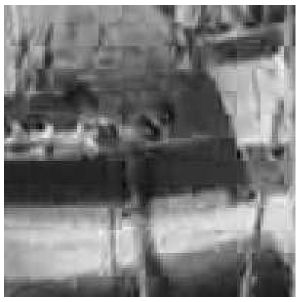


Figure 4.48: Quality comparison of the 'lena' image using the GMM-based block quantiser with and without the wavelet transform pre-processing at 0.5 bits/pixel: (a) Without wavelet transform pre-processing (PSNR=31.86 dB); (b) with wavelet transform pre-processing (PSNR=32.62 dB); (c) zoomed up region of image in (a), showing block artifacts; (d) zoomed up region of image in (b)

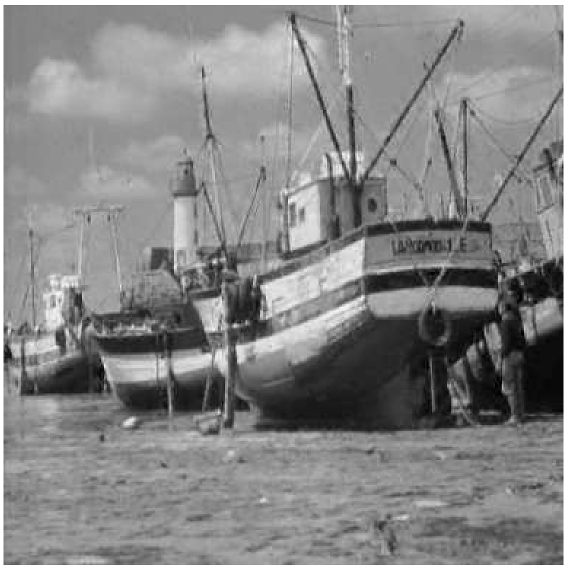
(a)



(c)



(b)



(d)

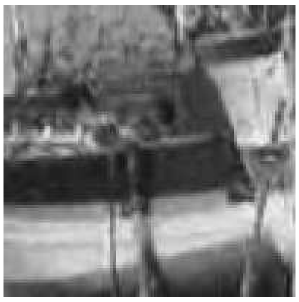


Figure 4.49: Quality comparison of the 'boat' image using the GMM-based block quantiser with and without the wavelet transform pre-processing at 0.5 bits/pixel: (a) Without wavelet transform (PSNR=29.03 dB); (b) with wavelet transform pre-processing (PSNR=29.59 dB); (c) zoomed up region of image in (a), showing block artifacts; (d) zoomed up region of image in (b)

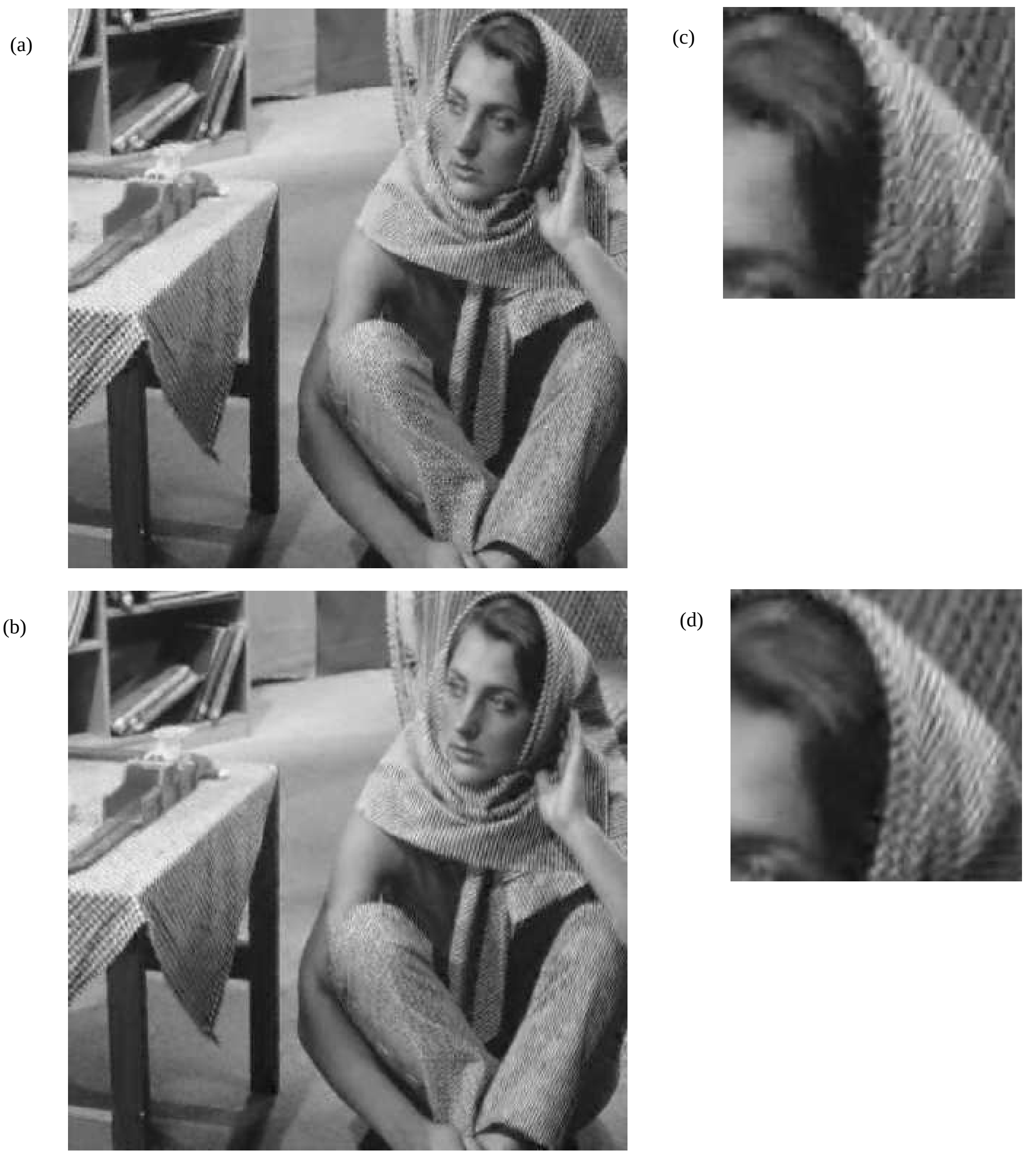


Figure 4.50: Quality comparison of the 'barbara' image using the GMM-based block quantiser with and without the wavelet transform pre-processing at 0.5 bits/pixel: (a) Without wavelet transform pre-processing (PSNR=26.06 dB); (b) with wavelet transform pre-processing (PSNR=26.75 dB); (c) zoomed up region of image in (a), showing block artifacts; (d) zoomed up region of image in (b)

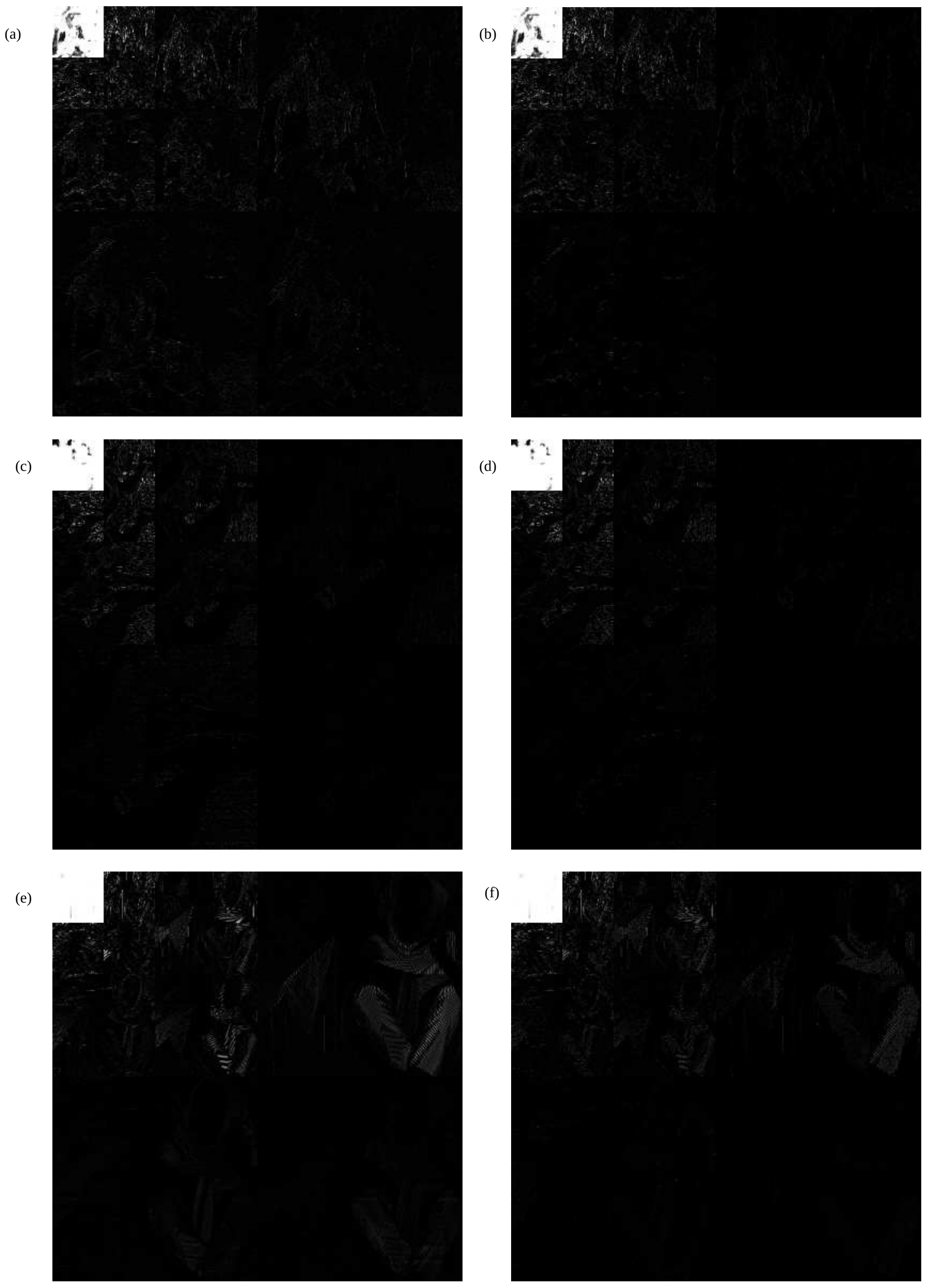


Figure 4.51: Comparing the wavelet transform with the quantised version at 0.5 bits/pixel, showing the preservation of important spatially-localised detail: (a) Wavelet transform of the image ‘man’; (b) quantised wavelet transform of the image ‘man’; (c) wavelet transform of the image ‘vegas’; (b) quantised wavelet transform of the image ‘vegas’; (d) wavelet transform of the image ‘barbara’; (e) quantised wavelet transform of the image ‘barbara’.

even and odd tapped filters as well as filters with unequal lengths. The remainder of the chapter was dedicated to the results and discussion of various quantisation schemes, such as the block quantiser based on the KLT and DCT and the GMM-based block quantiser. It was shown that the GMM-based block quantiser achieved higher PSNRs and better subjective quality than the traditional fixed-rate block quantiser/transform coder at a given bitrate, which demonstrates the advantages of accurate source PDF estimation and the use of multiple decorrelating transforms. Because images are mostly highly correlated and have Gauss-Markov properties, replacing the KLT with the data dependent DCT should result in comparable performance. Through PSNRs and visual inspection, we showed that the GMM-DCT-based block quantiser is comparable in quantisation performance, with only a fraction of the complexity. Next, a novel and low complexity method of encoding fractional bits in a fixed-rate framework and heuristic algorithms for compensating quantiser levels in bit allocation were evaluated and shown to improve the PSNR slightly. Finally, we presented a method of pre-processing an image using the wavelet transform before block quantisation that reduces block artifacts and improves the image quality. Table 4.38 presents a summary of the PSNR results for all the block quantisation schemes that were considered in this chapter.

Table 4.38: Summary of PSNRs for all quantisation schemes considered at 0.5 bits/pixel. These include: KLT-based block quantiser (KLT); DCT-based block quantiser (DCT); K-means multiple transform block quantiser (K-means); GMM-KLT-based block quantiser (GMM-KLT); GMM-KLT-based block quantiser with wavelet pre-processing (GMM-KLT-WT); GMM-DCT-based block quantiser (GMM-DCT). The number of clusters used is 16 for the K-means and GMM-based block quantisers.

Image Name	PSNR (in dB) at 0.5 bits/pixel					
	KLT	DCT	K-means	GMM-KLT	GMM-KLT-WT	GMM-DCT
baboon	22.89	22.75	23.45	23.97	24.07	23.84
barbara	23.75	23.55	24.55	26.06	26.75	25.46
fruits	26.08	25.93	28.52	30.17	30.62	29.92
lena	27.31	27.04	29.34	31.86	32.62	31.43
peppers	27.18	26.80	29.14	31.89	32.54	31.60
sailboat	25.80	25.54	27.79	28.70	29.27	28.46
man	24.67	24.47	25.66	26.66	27.09	26.48
goldhill	24.69	27.78	29.18	30.53	31.06	30.40
bridge	28.08	24.46	25.42	25.82	26.13	25.71
jet	24.69	25.38	28.65	29.92	30.22	29.53
pyramid	27.33	26.99	29.14	31.32	31.78	31.06
aero	27.47	27.13	29.12	30.32	31.22	29.98
einstein	29.00	28.77	30.62	34.21	34.78	33.93
hat	27.07	26.91	29.24	31.26	32.67	30.85
london	27.94	27.54	30.00	31.95	32.38	31.79
tekboat	20.82	20.57	22.10	22.99	23.33	22.88
tekrose	18.90	18.79	20.06	20.75	21.00	20.67
loco	22.18	21.92	23.69	24.86	25.11	24.77
boat	26.02	25.76	27.40	29.03	29.59	28.73
kids	23.14	22.99	24.12	24.98	25.14	24.90
crowd	18.80	18.64	20.35	21.15	21.57	21.04
couple	29.04	28.73	31.11	37.47	39.11	36.84
mill	22.77	22.50	23.77	24.64	24.83	24.52
vegas	27.62	27.35	29.96	31.34	32.34	31.00

Chapter 5

LPC Parameter Quantisation in Narrowband Speech Coding

5.1 Abstract

In this chapter, we report on the contributions to LPC parameter quantisation in the area of narrowband speech coding. We first provide a brief review of speech coding, such as speech production, autoregressive and linear prediction modelling, and LPC speech coders that have been standardised and adopted by industry. These include the LPC vocoder, RPE-LTP and the CELP speech coders, which form the basis of numerous low bitrate speech coding standards. The preservation of filter stability and accurate quantisation of LPC coefficients in these speech coders is quite important, so various LPC parameter representations, such as LARs and LSF, that are robust to quantisation errors, will be reviewed. Finally, the rest of the chapter is dedicated to providing experimental results of the multi-frame GMM-based block quantiser and switched split vector quantiser on line spectral frequency (LSF) quantisation as well as discussing how they compare with other quantisation schemes in terms of spectral distortion, computational complexity, and memory requirements.

Publications resulting from this research: [130, 166, 169]

5.2 Preliminaries of Speech Coding

5.2.1 Narrowband versus Wideband Speech

Speech needs to be converted to digital form before being transmitted over band-limited communication channels. The channels of traditional telephone speech are band-limited from 300 Hz to 3400 Hz. Hence speech that is recorded from a microphone is initially filtered using an anti-aliasing filter with a cut-off frequency of 3.4 kHz, before being sampled at 8 kHz at a resolution of 16 bits. The bitrate required to transmit this speech, with no coding applied, is therefore 128 kbps. This is termed as telephone or *narrowband speech* (300–3400 Hz) [124].

With the introduction of high-speed data services in wireless communication systems, speech of wider bandwidth can be accommodated [19]. *Wideband speech* (50–7000 Hz) is sampled at 16 kHz and, compared with narrowband speech, has improved naturalness and intelligibility due to the added bandwidth. For more information about wideband speech coding, the reader should refer to Chapter 6.

5.2.2 Speech Production

Speech sounds can be broadly classified as either *voiced* or *unvoiced*. Voiced sounds, which include vowels such as */iy/* (as in s*ee*) and nasal sounds, such as */m/*, are periodic and possess a harmonic structure that is not present in unvoiced sounds, such as */s/*, which are aperiodic and noise-like. These are best visualised in Figure 5.1, which shows the waveform and spectrogram of the sentence, *she had your dark suit in greasy wash-water all year*, and highlights the voiced and unvoiced sections in the first word, *she*. Notice that the spectrum for */sh/* is flat, similar to that of noise, while the spectrum of */iy/* shows a harmonic structure, as characterised by the alternating bands.

Figure 5.2 shows the anatomy of the human vocal tract. Voiced sounds are produced when air from the lungs is excited by the vibrating vocal folds in the larynx. A glottal wave, with a fundamental frequency of f_0 and harmonics at multiples of the fundamental frequency, is generated and this wave passes through the vocal tract, which can be viewed as an acoustic tube that starts at the larynx and terminates at the lips. This tube changes shape by altering various cross-sectional areas to create resonances and anti-resonances

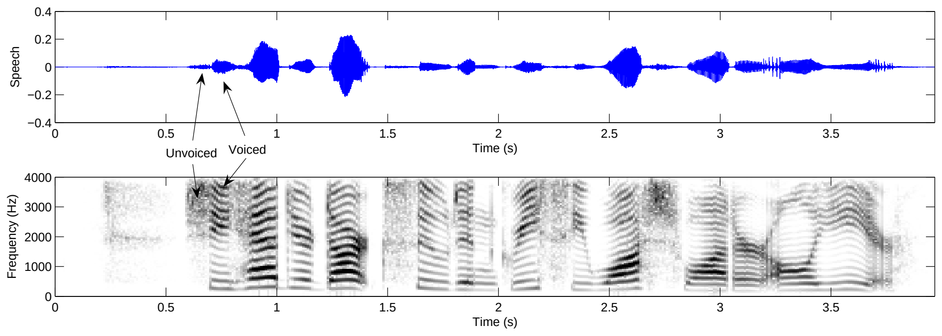


Figure 5.1: Waveform and spectrogram of narrowband speech. The sentence that is spoken is *'she had your dark suit in greasy wash-water all year'*, and with the unvoiced /*s*/ and voiced /*iy*/ sounds in *she*, highlighted.

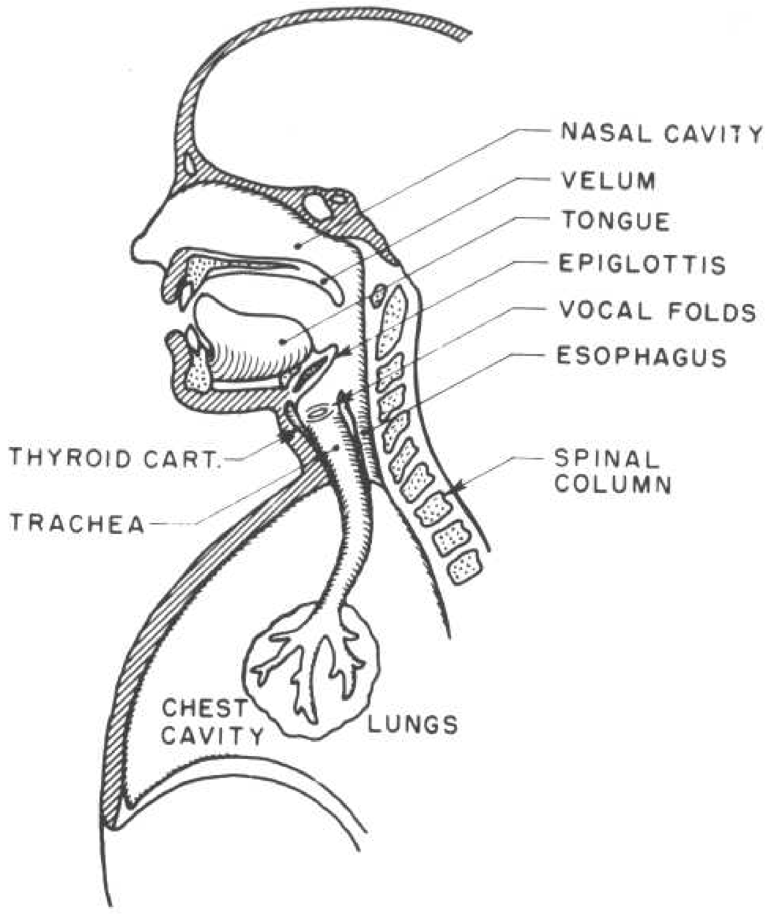


Figure 5.2: Anatomy of the human vocal tract (after [139])

that emphasise and de-emphasise certain parts of the spectrum, respectively. *Formants* occur where the spectrum has been emphasised by the resonances of the vocal tract. Along with changes in the articulators (the lips, tongue, jaws, and teeth), different quasi-periodic sounds can be produced [73, 151]. The vocal folds do not vibrate for unvoiced sounds but instead, the vocal tract is constricted by the articulators and air passes through rapidly to produce a noise-like sound [151].

5.2.3 Autoregressive Modelling and Linear Prediction Analysis of the Speech Signal

Speech production can be modelled as consisting of a *source* and *filter* component. The source component represents the glottal excitation (which is aperiodic noise for unvoiced sounds and periodic at the fundamental frequency for voiced sounds) while the filter component models the vocal tract and its resonances and anti-resonances, by emphasising and de-emphasising certain parts of the spectrum using poles and zeros, respectively. Because the zeros of the vocal tract for unvoiced and nasal (voiced) sounds have been observed to lie within the unit circle, they can be approximated by a large number of poles[1], hence an all-pole or autoregressive (AR) model can be used [14]:

$$H(z) = \frac{G_p}{A(z)} \tag{5.1}$$

$$A(z) = 1 + \sum_{k=1}^{p} a_{p,k} z^{-k} \tag{5.2}$$

where p is the filter order, $a_{p,k}$ are the p filter coefficients or AR model parameters, and G_p is the gain to conserve the total energy between the speech and impulse response of $H(z)$ [195]. $H(z)$ and $A(z)$ are called the synthesis and analysis filters, respectively. Figure 5.3 shows the source-filter model of speech, where speech, $x(n)$, is synthesised by the filter, $H(z)$, which is driven by an excitation signal, $u(n)$. The AR model is equally valid when the excitation signal is white random noise or an impulse, because the autocorrelations (and hence the PSDs) of both signals are identical [107]. This fits into the voiced/unvoiced speech production model perfectly.

Finding the AR model parameters and gain for a particular segment of speech involves

[1]The zero, $1 - az^{-1}$, can be approximated by $1/(1 + az^{-1} + a^2z^{-2} + \dots)$ if $|a| < 1$ [14].

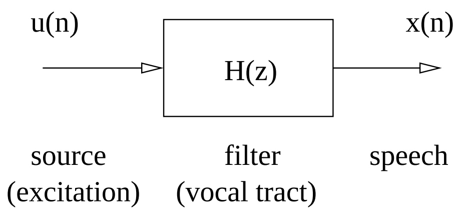


Figure 5.3: Source-filter synthesis model of speech

solving the Yule-Walker equations [65, 107]:

$$\sum_{k=1}^{p} a_{p,k} R(j-k) = -R(j) \tag{5.3}$$

$$G_p^2 = R(0) + \sum_{k=1}^{p} a_{p,k} R(k) \tag{5.4}$$

where $j = 1, 2, \ldots, p$. When (5.3) is expressed in matrix notation:

$$\begin{bmatrix} R(0) & R(1) & \ldots & R(p-1) \\ R(1) & R(0) & \ldots & R(p-2) \\ \vdots & \vdots & \ddots & \vdots \\ R(p-1) & R(p-2) & \ldots & R(0) \end{bmatrix} \begin{bmatrix} a_{p,1} \\ a_{p,2} \\ \vdots \\ a_{p,p} \end{bmatrix} = \begin{bmatrix} -R(1) \\ -R(2) \\ \vdots \\ -R(p) \end{bmatrix} \tag{5.5}$$

then it can be seen that the autocorrelation matrix has a special Toeplitz structure, which guarantees that the resulting filter is minimum phase (and hence, is stable) [107] and also allows the use of computationally efficient algorithms such as the Levinson-Durbin or Schur recursion algorithms to solve (5.5) [125]. The Levinson-Durbin algorithm is given below:

For $m = 1, 2, \ldots, p$:

$$a_{m,m} = -\frac{R(m) + \sum_{i=1}^{m-1} a_{m-1,i} R(m-i)}{P_{m-1}} \tag{5.6}$$

$$a_{m,i} = a_{m-1,i} + a_{m,m} a_{m-1,m-i}, \text{ where } i = 1, 2, \ldots, m-1 \tag{5.7}$$

$$P_m = P_{m-1}(1 - a_{m,m}^2) \tag{5.8}$$

where the filter gain, G_p, is equal to P_p. To initialise the Levinson-Durbin algorithm, $P_0 = R(0)$. The $a_{m,m}$'s are also known as the reflection coefficients (RCs) or partial correlation (PARCOR) coefficients [77] which will be discussed in a later section on LPC parameter representations.

Another approach that arrives at the same set of equations is *linear prediction analysis*, where the speech signal is modelled as consisting of the summation of a predicted part and a residual part[2], where the former is a weighted average of past (or future) samples:

$$\hat{x}(n) = -\sum_{k=1}^{p} a_{p,k} x(n-k) \tag{5.9}$$

where $x(n)$ is the signal to be modelled, $\hat{x}(n)$ is the predicted signal, and p is the order of the prediction. The aim of linear prediction analysis is to determine the weights (or linear prediction coefficients), $a_{p,k}$, that minimise the average prediction error, P, over a certain range of time, and depending on that range, various methods can be obtained. The average prediction error is given by:

$$\begin{aligned} P &= E[(x(n) - \hat{x}(n))^2] & (5.10)\\ &= E\left[\left(x(n) + \sum_{k=1}^{p} a_{p,k} x(n-k)\right)^2\right] & (5.11)\end{aligned}$$

where $E[\bullet]$ is the expectation operator.

The *autocorrelation method* [108] tries to minimise the average prediction error over the time range of $-\infty < n < \infty$ and assumes the signal to exist for all time. If a finite segment of the signal is available (which is always the case, in practice), then it is assumed to be a windowed version of the original signal. Also, the signal is assumed to be wide-sense stationary (WSS), where $E[x(n+i)x(n+j)] \equiv E[x(n)x(n+|i-j|)]$. After minimising the average prediction error via setting the partial derivative to zero, the following equations are obtained for the autocorrelation method [107]:

$$\begin{aligned} \sum_{k=1}^{p} a_{p,k} R(|k-l|) &= -R(l) & (5.12)\\ P_{min} &= R(0) + \sum_{k=1}^{p} a_{p,k} R(k) & (5.13)\end{aligned}$$

[2]This type of decomposition is known as the *Wold decomposition*.

where $l = 1, 2, \ldots, p$. The autocorrelation coefficients[3], $R(k)$, are given by:

$$R(k) = \frac{1}{N} \sum_{n=0}^{N-1-k} x(n)x(n+k) \tag{5.14}$$

It can be seen that these equations are the same as the Yule-Walker equations. Also, the WSS assumption results in the Toeplitz structure of the autocorrelation matrix, and hence guarantees the stability of the synthesis filter.

The *covariance method* [17] minimises the prediction error over the finite range, $0 \leq n \leq N-1$, where N is the length of the available speech segment to be analysed [107], hence it makes no assumptions of the signal outside the range (ie. no windowing). Also no stationarity assumptions are made about the signal. After minimising the average prediction error via setting the partial derivative to zero, the following equations are obtained for the covariance method [17]:

$$\sum_{k=1}^{p} a_{p,k}\phi_{k,l} = -\phi_{0,l} \tag{5.15}$$

$$P_{min} = \phi_{0,0} + \sum_{k=1}^{p} a_{p,k}\phi_{0,k} \tag{5.16}$$

where the covariance coefficients, $\phi_{i,j}$, are given by:

$$\phi_{i,j} = \frac{1}{N-p} \sum_{n=p}^{N-1} x(n-i)x(n-j) \tag{5.17}$$

The covariance matrix of ϕ's is symmetric, but unlike the autocorrelation matrix in the previous method, it is not of Toeplitz form. Therefore, the computationally efficient Levinson-Durbin algorithm cannot be used. Also, it is possible for some of the resulting poles to fall outside the unit circle, which renders the filter unstable[4] [15]. The likelihood of instability decreases as more data is available, since as $N \to \infty$, the covariance method converges to the autocorrelation method [107]. Methods for stabilising the covariance method include adding a very small number to the diagonal elements of the covariance matrix [107].

The power spectral density (PSD) of the linear prediction filter, $H(z)$, which is given

[3]This is a biased autocorrelation since the data size is finite and for higher lags, the number of data points used in the calculation will be less than N.

[4]The stability of the filter is only an issue for applications that require synthesis. For these applications, the autocorrelation method is preferred.

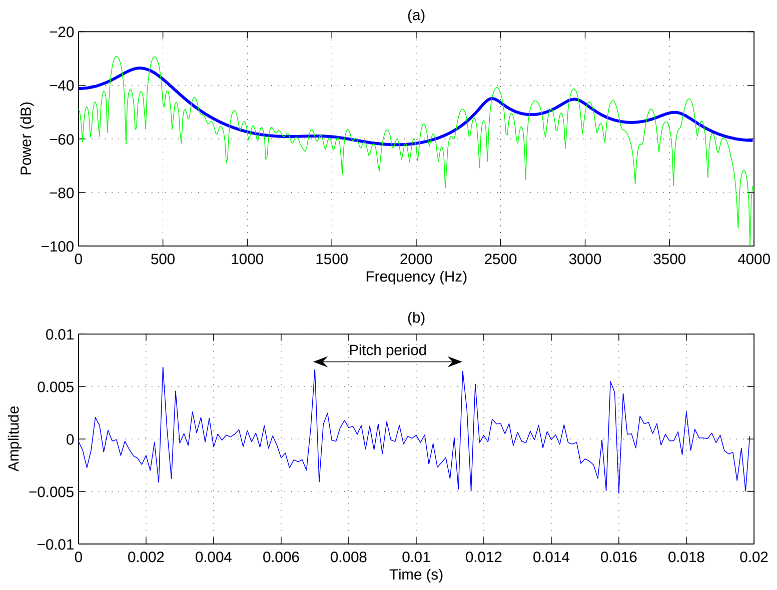


Figure 5.4: Estimation of the spectral envelope and excitation/residual signal of voiced speech, $/i/$ as in 'y*ea*r', using a 12th order linear prediction filter: (a) Periodogram (thin line) and spectral envelope estimate (thick line) using the autocorrelation method; (b) the residual or excitation signal.

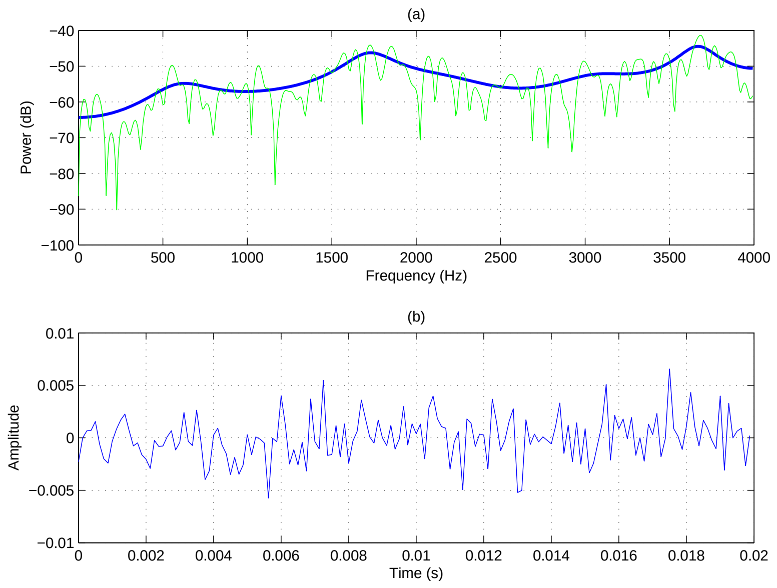


Figure 5.5: Estimation of the spectral envelope and excitation/residual signal of unvoiced speech, $/s/$ as in '*s*uit', using a 12th order linear prediction filter: (a) Periodogram (thin line) and spectral envelope estimate (thick line) using the autocorrelation method; (b) the residual or excitation signal.

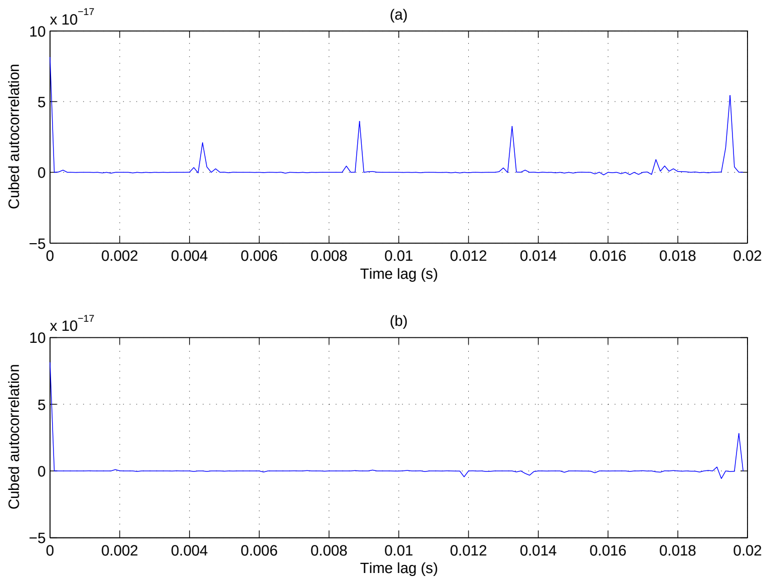


Figure 5.6: Cubed autocorrelation (unbiased) of the residual signal for: (a) voiced speech, /*i*/ as in 'y*ea*r'; (b) unvoiced speech, /*s*/ as in '*s*uit'.

by:

$$|H(e^{j\omega})|^2 = \frac{G_p^2}{|1 + \sum_{k=1}^{p} a_{p,k} e^{-j\omega k}|} \tag{5.18}$$

estimates the spectral envelope of the power spectral density of the speech to be modelled. This is shown in Figure 5.4(a), which shows the spectral envelope estimate from a 12th order linear prediction filter overlaid on top of the periodogram of the voiced speech segment to be analysed. The length of the speech was 20 ms and the autocorrelation method was used for the linear prediction analysis. Most of the formants, which represent the resonances of the vocal tract, are captured by the linear prediction filter. The spectral envelope, as represented by the linear prediction coefficients, is also known as short-term correlation information because the analysis is based on relatively low autocorrelation lags, $(p + 1)$.

The excitation or residual signal is shown in Figure 5.4(b), where we can see the presence of periodic peaks. In order to measure the periodicity of the residual, we can calculate its autocorrelation function, which is shown in Figure 5.6(a). We have used the unbiased autocorrelation[5] to avoid the suppression at higher lags. Furthermore, the autocorrelation has been cubed to enhance the peaks. The first distinct peak is situated at 4.4 ms, which corresponds to a fundamental frequency (pitch) of about 227 Hz. This is

[5] $R_{unbiased}(k) = \frac{1}{N-k} \sum_{k=0}^{N-1-k} x(n)x(n+k)$

to be expected as the speech is voiced, where the vocal folds vibrate to produce a quasi-periodic glottal excitation. The spectrum of the residual signal, which highlights the fine structure of speech, is also known as long-term correlation information since the strongest correlations or similarities in speech (in particular, voiced speech) appear at lags that correspond to, and are multiples of, the pitch period (as shown in Figure 5.6(b)). These lags are much longer than those used in the spectral envelope estimate.

Figures 5.5(a) and (b) show the PSD estimate and excitation/residual signal from a 12th order linear prediction filter of unvoiced speech. We can observe the relative flatness of the spectrum, which is the noise-like characteristic of unvoiced speech. The residual signal appears aperiodic and can be verified in Figure 5.6(b), which shows the cubed autocorrelation function of the residual for unvoiced speech. There are no distinct peaks observable, hence the residual lacks periodicity and correlation.

From these observations, it is apparent that linear prediction can be used to separate the spectral envelope from the fine structure of speech, though as noted in [17], this should be distinguished from the separation of the spectrum of the vocal tract from that of the source. In fact, it is known for linear predictors of low order, that some spectral envelope information remains in the residual signal [74]. This can be seen in Figure 5.4, where the spectral envelope estimate resolves one initial peak, when in fact, there appears to be two peaks shown by the periodogram.

From a linear prediction perspective, we can see that the predictor is good at estimating the signal within each pitch period for voiced speech. If we plot the normalised residual error [107]:

$$V_p = \frac{P_{min}}{R(0)} \tag{5.19}$$

for each linear prediction order for voiced and unvoiced speech, as shown in Figure 5.7, we can also see that the prediction error for unvoiced speech is higher than that for voiced speech. Low residual errors are associated with large dynamic ranges of the spectrum [107] and it has been suggested that V_p is a useful parameter for classifying speech as either voiced or unvoiced [108].

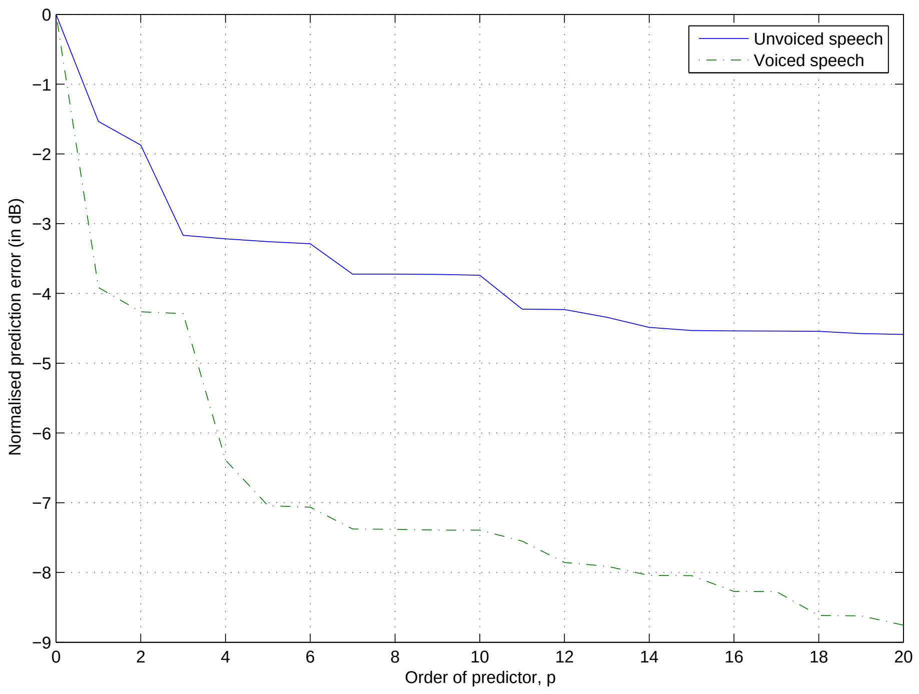


Figure 5.7: Plot of normalised residual error versus order of linear prediction for voiced and unvoiced speech

5.3 Linear Prediction-Based Speech Coding

The exploitation of redundancy is important in improving the bitrate-versus-distortion efficiency of quantisation. That is, if a part of the signal can be reproduced or synthesised at the decoder side via a smaller set of parameters, then significant gains in coding can be achieved. As we have seen in the previous sections, speech contains both short-term correlations (spectral envelope) and long-term correlations (residual), that can be exploited by using linear prediction-based coding.

There are two types of prediction that are considered and these are shown in Figure 5.8. In *forward prediction* (Figure 5.8(a)), the prediction coefficients are calculated based on previous input samples. Because the decoder does not have access to the input samples, it needs to receive the prediction coefficients from the encoder explicitly as side information. Also, a delay is required in order to obtain enough samples for the prediction analysis. In *backward prediction* (Figure 5.8(b)), the prediction coefficients are calculated based on previously reconstructed samples [124]. Consequently, the decoder has enough information to derive the prediction coefficients, thus no side information is required. Also, the backward prediction coder does not impart any delay. Apart from the advantage of having no

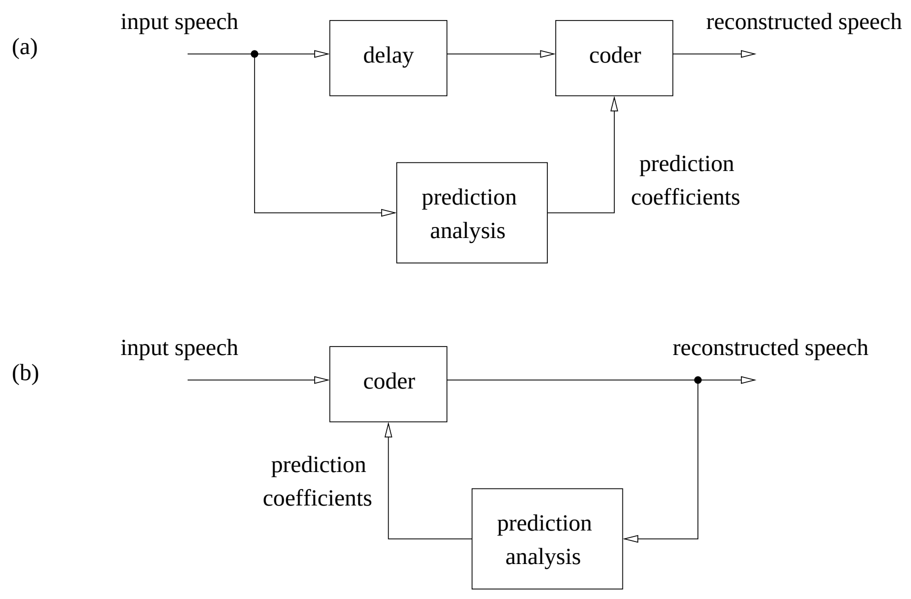


Figure 5.8: Different types of prediction used in speech coding (after [91]): (a) forward prediction, where the prediction coefficients are calculated based on input speech; (b) backward prediction, where the prediction coefficients are calculated based on output or reconstructed speech.

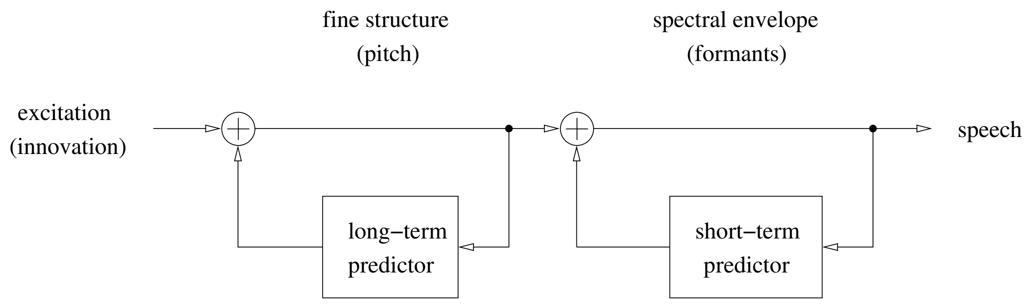


Figure 5.9: Block diagram of generalised speech synthesis (after [153])

delay, backward prediction is known to have lower prediction gain than forward prediction. Furthermore, the efficiency of the backward predictor is dependent on the quality of the reconstructed speech, thus this approach becomes less effective at low bitrates [91, 124]. For this reason, low bitrate speech coders normally employ forward prediction.

The generalised speech synthesis model is shown in Figure 5.9. The long-term predictor and short-term predictor generate the pitch periodicity (for voiced speech) and spectral envelope, respectively [153]. The excitation (also known as the *innovation*), is usually taken as white random noise. This speech synthesis model is used in speech coders that are based on the *analysis-by-synthesis* principle, which will be discussed in Section 5.3.4.

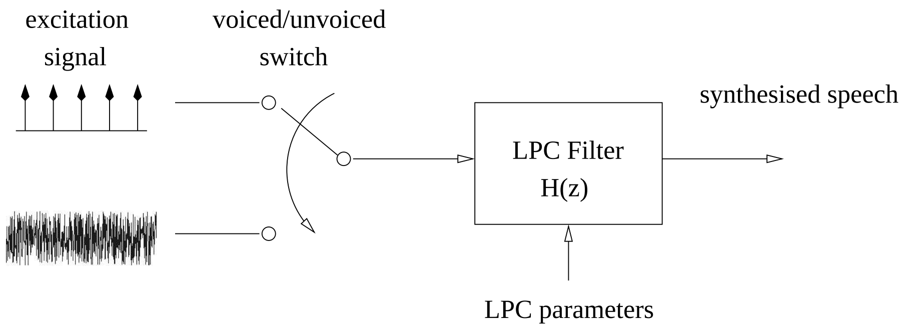


Figure 5.10: Block diagram of the LPC vocoder (after [16]). The excitation for voiced speech is a train of impulses, separated by the pitch period, while for unvoiced speech, it is white random noise.

In the following sections, we examine some speech coding approaches that have been investigated in the literature or are being used in mobile communications standards such as GSM.

5.3.1 The LPC Vocoder

Compared with waveform coders, which attempt to reconstruct the original speech as faithfully as possible in a perceptually efficient manner [124], voice coders or *vocoders*[6] synthesise speech based on a parametric model, thus they belong to the class of parametric coders. Their only advantage is the ability to operate at relatively low bitrates (less than 4 kbps), though this is at the expense of generally lower speech quality and intelligibility [16].

Figure 5.10 shows the block diagram of a simple LPC vocoder, where we observe its similarity to the source-filter model of speech production. Speech is assumed to be either voiced or unvoiced speech and based on this decision, a different excitation signal is fed into the LPC filter to produce the synthesised speech. The parameters that need to be quantised and transmitted, in order to allow the decoder to synthesise the speech, are:

- the LPC parameters;
- signal power;
- pitch period (for voiced speech); and

[6]The LPC vocoder is related to the channel vocoder, that was developed by Dudley [41].

- voiced/unvoiced decision.

Speech is broken into frames at 50 Hz and an LPC analysis is performed to obtain the LPC parameters. These parameters are usually not the LPC coefficients themselves, as they are their sensitivity to quantisation errors often leads to filter instability. The signal power of the frame is also calculated. Following this, a pitch estimation algorithm is applied on the low-pass residual signal or weighted speech signal to determine whether the framed speech is voiced or unvoiced [124]. During this process, the pitch period is also obtained and transmitted for voiced frames.

An example of an LPC vocoder is the US Department of Defense LPC-10 vocoder (FS-1015), which operates at 2.4 kbps and is intended for secure voice terminals [31]. In the LPC-10 vocoder, reflection coefficients are used as the LPC parameter representation and pitch estimation is performed by finding the minimum in the average magnitude difference function (AMDF) of the low-pass inverse filtered speech [49].

It is noted, however, that the naturalness and intelligibility of the LPC vocoder is rather poor and this is mainly due to the binary decision between voiced and unvoiced speech [124]. It is a well known fact that in some segments of speech, the binary classification is often difficult and there may be more than two modes of vocal tract excitations (ie. mixed excitations). Also, the assumption of a single impulse excitation within each pitch period is inadequate and it has been shown that multiple impulses are necessary [16]. Atal and Remde [16] introduced an improved quality LPC vocoder based on a multi-pulse excitation model. The excitation model relied on an analysis-by-synthesis procedure with a perceptually weighted error measure to determine appropriate impulse locations and amplitudes. Also, this new vocoder did not require a voiced/unvoiced classification as the same excitation model is used for both voiced and unvoiced speech.

5.3.2 The RPE-LTP Speech Coder

The regular pulse excitation, long term prediction (RPE-LTP) coder is used in the full-rate (13 kbps) GSM narrowband speech coder. This coder, as shown in Figure 5.11, exploits the long-term and short-term correlations in speech, which are represented in the form of LTP gain and lag as well as log-area-ratios (LARs), respectively. After the removal of these two types of redundancy, the residual excitation is then coded.

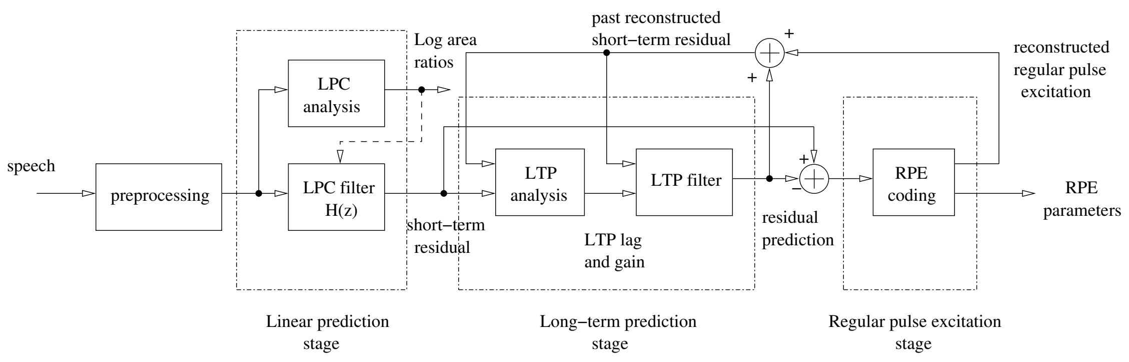


Figure 5.11: Block diagram of the RPE-LTP speech coder used in GSM 06.10 (after [113])

Speech is first windowed into 20 ms frames of 160 samples and then pre-emphasised. A short-term LPC analysis is performed using the Schur recursion to produce eight reflection coefficients, which are then converted to LARs and quantised using 36 bits. The short-term residual signal is then obtained by filtering the speech with the LPC filter (derived from the quantised LARs). For the long-term prediction (LTP) stage, each frame is split into 4 sub-frames of 40 samples each, where an LTP lag and gain are calculated for each sub-frame. These LTP parameters are obtained from the cross-correlation between the current residual sub-frame and the previous reconstructed residual sub-frames[7] [113]. The aim of the LTP analysis is to search for a past reconstructed residual sub-frame that is the most similar to (ie. most correlated with) the current residual sub-frame [36]. Using the two LTP parameters and past reconstructed residual sub-frames, an LTP filter generates a predicted short-term residual sub-frame, which is subtracted from the current short-term residual sub-frame to give the regular pulse excitation (RPE) sub-frame. The 40 samples in the RPE sub-frame are then decimated and interleaved to 13 values which are then coded using adaptive PCM [113]. The reconstructed RPE sub-frame is then added with the predicted short-term residual to give the past residual sub-frame which is used for subsequent LTP analyses.

5.3.3 Differential Pulse Coded Modulation with Perceptual Weighting

Though this type of speech coder, first introduced by Atal and Schroeder [15], may not be as sophisticated as the modern speech coders, it provided the foundation for powerful analysis-by-synthesis coders such as CELP, which will be discussed in the next section.

[7]The LTP gain is the maximum correlation normalised by the energy of the sub-frame [36].

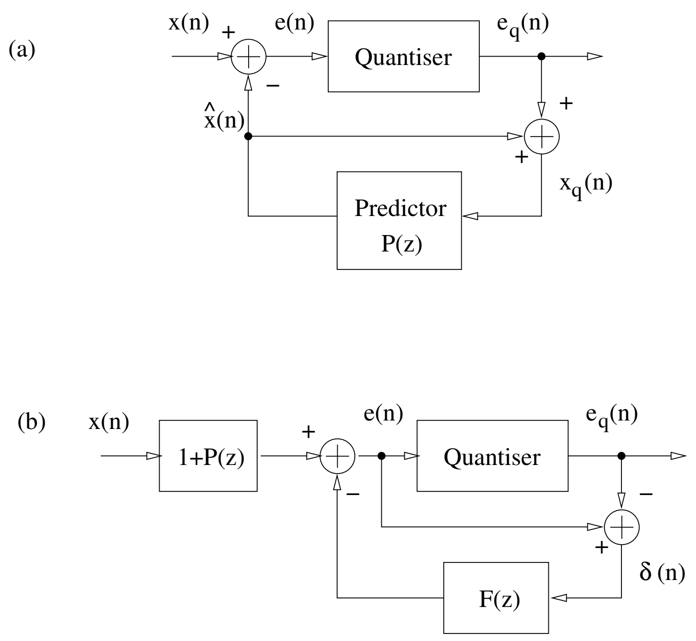


Figure 5.12: Block diagram of the DPCM coder: (a) without noise shaping; (b) with noise shaping filter, $F(z)$ (after [15])

The main novelty of the DPCM speech coder is the use of filtering to spectrally shape the quantisation noise in order to exploit perceptual masking. Traditionally, DPCM coders attempt to minimise the MSE between the reconstructed and original speech, which is equivalent to the error introduced by the scalar quantiser [15]. Assuming an efficient predictor is used, then the quantisation error will be similar to uncorrelated noise, which has a flat power spectral density. In the presence of the speech signal which has a non-uniform PSD, the human auditory response to the quantisation noise will also be non-uniform, hence minimising the MSE is clearly inadequate, from a perceptual point of view.

The theory of auditory masking says that strong tones tend to mask other tones and noise within the spectral vicinity, hence it was suggested in [15] that a large part of the perceived noise from speech coding comes from quantisation noise that is situated in regions where the speech spectrum is low. In order words, the perceptual quality can be improved by spectrally shifting some quantisation noise from the low power frequency regions to the stronger formant regions.

In order to see how Atal and Schroeder [15] spectrally shaped the quantisation noise in DPCM, they considered the traditional DPCM coder, as shown in Figure 5.12(a). The reconstructed speech signal, $x_q(n)$, can be expressed as:

$$\begin{aligned} x_q(n) &= \hat{x}(n) + e_q(n) \\ &= \hat{x}(n) + e(n) + \delta(n) \qquad (5.20) \\ &= x(n) + \delta(n) \qquad (5.21) \end{aligned}$$

where $\delta(n) = e(n) - e_q(n)$ is the error introduced by the quantiser. Therefore, the error between the reconstructed and original speech is equal to the error introduced by the quantiser only. The predictor, $P(z)$, attempts to generate a predicted speech sample, based on the past p reconstructed samples:

$$\begin{aligned} \hat{X}(z) &= P(z)X_q(z) \qquad (5.22) \\ \text{where } P(z) &= -\sum_{k=1}^{p} a_k z^{-k} \qquad (5.23) \\ \text{hence } \hat{x}(n) &= -\sum_{k=1}^{p} a_k x_q(n-k) \qquad (5.24) \end{aligned}$$

Therefore, the residual signal, $e(n)$, can be expressed as:

$$\begin{aligned} e(n) &= x(n) - \hat{x}(n) \\ &= x(n) + \sum_{k=1}^{p} a_k x_q(n-k) \qquad (5.25) \end{aligned}$$

Substituting (5.21) into (5.25), we get:

$$\begin{aligned} e(n) &= x(n) + \sum_{k=1}^{p} a_k[x(n-k) + \delta(n-k)] \\ &= \underbrace{x(n) + \sum_{k=1}^{p} a_k x(n-k)}_{\text{prediction error}} + \underbrace{\sum_{k=1}^{p} a_k \delta(n-k)}_{\text{filtered quantisation error}} \qquad (5.26) \\ E(z) &= [1 + P(z)]X(z) + P(z)\Delta(z) \qquad (5.27) \end{aligned}$$

Therefore, the residual, $e(n)$, can be expressed as the summation of the prediction error and filtered quantisation error [15]. In (5.27), we can apply a different filter, $F(z) = -\sum_{k=1}^{p} b_k z^{-k}$, in order to shape the quantisation noise, $\Delta(z)$:

$$E(z) = [1 + P(z)]X(z) + F(z)\Delta(z) \qquad (5.28)$$

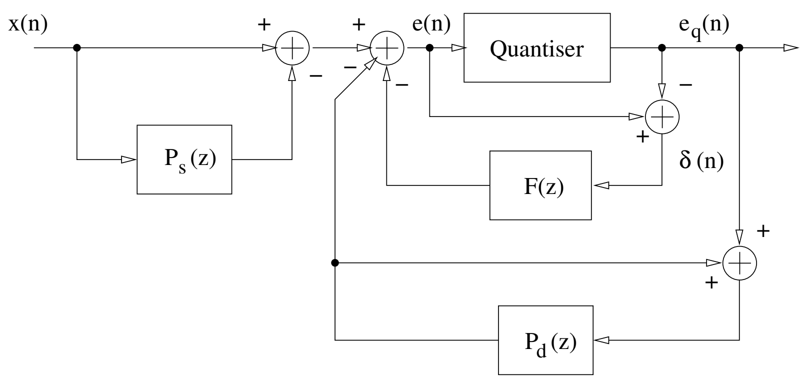


Figure 5.13: Block diagram of the generalised DPCM speech coder (after [15])

This new DPCM with quantisation noise shaping is shown in Figure 5.12(b). Given that the reconstructed speech is given by:

$$X_q(z) = \frac{E_q(z)}{1 + P(z)} \tag{5.29}$$

and $E_q(z) = E(z) + \Delta(z)$, (5.29) can be rearranged to give [15]:

$$X_q(z) - X(z) = \Delta(z)\frac{1 + F(z)}{1 + P(z)} \tag{5.30}$$

Equation (5.30) basically says the error between the reconstructed speech and the original speech is equal to the filtered quantisation noise. For example, when $F(z) = P(z)$, equation (5.30) reduces to $X_q(z) - X(z) = \Delta(z)$, which is the traditional DPCM with no noise shaping [15]. In other words, the error spectrum is equal to that of scalar quantisation which is flat, assuming the prediction error to be white. Therefore, by changing $F(z)$, the frequency spectrum of the quantisation noise can be adjusted.

Atal and Schroeder [15] showed that the average logarithm of the squared magnitude of $\frac{1+F(e^{j\omega})}{1+P(e^{j\omega})}$ is equal to zero. This suggests that the average power of the reconstructed error remains unaltered and adjusting the filter, $F(z)$, redistributes the noise power from one frequency to another [15]. In the case of speech coding, it is desirable to shift quantisation noise from frequencies where the speech PSD is low to the formant regions, where the speech power is stronger and therefore, is able to perceptually mask the noise.

Figure 5.13 shows the generalised DPCM speech coder, which includes a long-term

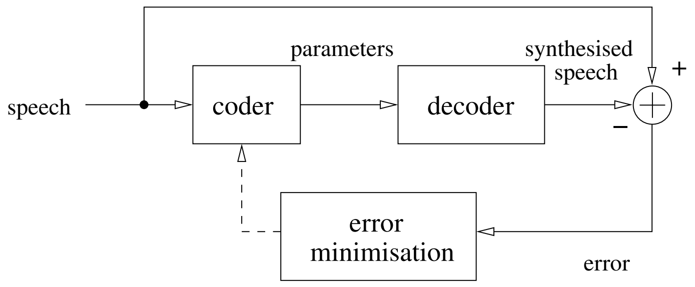


Figure 5.14: The analysis-by-synthesis speech coder (after [91])

predictor, $P_d(z)$:

$$P_d(z) = -(\beta_1 z^{-D+1} + \beta_2 z^{-D} + \beta_3 z^{-D-1}) \tag{5.31}$$

where D is the predictor delay, which is usually set to be equal to (or, a multiple of) the pitch period. The long-term predictor is used for removing the periodic redundancy in the spectral fine structure, along with the short-term predictor, $P_s(z)$:

$$P_s(z) = -\sum_{k=1}^{p} a_k z^{-k} \tag{5.32}$$

which is used to remove (or 'whiten') the spectral envelope. The noise shaping filter, $F(z)$, is set to be equal to the bandwidth-widened $P_s(z)$ [15]:

$$F(z) = P_s(z/\alpha) \tag{5.33}$$

where α is between 0 and 1. This tends to move the zeros of $1 + P_s(z)$ closer to the origin, which widens the resonant bandwidths [15].

5.3.4 Code Excited Linear Predictive Speech Coders

The Code Excited Linear Predictive (CELP) speech coder was first introduced by Schroeder and Atal [153] and is the basis for modern speech coders, which achieve high speech quality at low bitrates. Examples of speech coding standards based on CELP include the US Federal Standard 4.8 kbps speech coder (FS-1016) [27], the Adaptive Multi-Rate (AMR) speech coder [2], the GSM Enhanced Full Rate (EFR) speech coder [113], etc.

The CELP coder is based on the *analysis-by-synthesis principle*, shown in Figure 5.14, where the coder parameters are evaluated by synthesising the speech and comparing it

Table 5.1: Comparing the characteristics of the DPCM with perceptual weighting and CELP speech coders. The $\triangle$ indicates that the coding scheme has the specified characteristic. (*) means that this is ambiguous.

Characteristics	DPCM	CELP
Analysis-by-synthesis coder	(*)	$\triangle$
Fixed codebook	$\triangle$ (scalar)	$\triangle$ (vector)
Adaptive codebook	$\triangle$ (scalar)	$\triangle$ (vector)
Perceptual weighting	$\triangle$	$\triangle$

with the the input speech. The coder parameters which minimise the distortion are then chosen. The speech synthesis is based on the generalised model (Figure 5.14), where short-term and long-term predictors, representing the spectral envelope and fine structure, respectively, are driven by an excitation or innovation signal. Unlike the LPC vocoder, where the excitation is parametrically modelled as either an impulse train with a pitch period or white random noise with a certain variance, the CELP coder quantises the excitation, $U(z)$, using a vector quantiser (hence the name, *code excited* linear prediction). That is, the random innovation is generated by selecting from a fixed codebook of white Gaussian random vectors, called the *stochastic codebook* [153], and the periodic excitation is generated by either an *adaptive codebook* or *pitch predictor* [91]. The distortion criteria for selecting the best code-vector from each codebook is the difference or residual, $E(z)$, between the synthesised speech and original speech that is perceptually weighted by the filter, $W(z)$. Figure 5.15 shows a block diagram of the CELP coder.

When comparing the CELP speech coder with the DPCM speech coder from the previous section, we notice a few similarities. Table 5.1 lists the similarities between the two coding schemes. Firstly, both schemes are analysis-by-synthesis coders, though the DPCM case can be somewhat ambiguous [124]. In the DPCM scheme, the short-term and long-term correlations are removed from the original speech signal, giving a residual/excitation that is compared with (or, quantised by) a scalar codebook. Because the quantiser error is perceptually weighted and fed back, the scalar codebook search is done on the basis of minimising the perceptual error. Also, because the long-term predictor uses quantised residual samples, its operation is adaptive and is dependent on the scalar codebook search.

Whereas in the case of CELP, short-term and long-term correlations are added to the

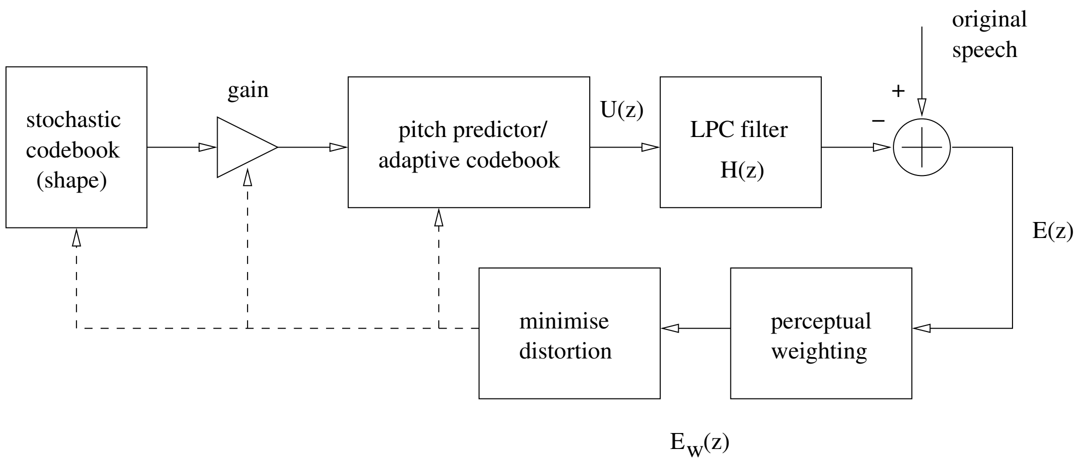


Figure 5.15: Block diagram of the CELP coder. Dashed lines indicate control lines.

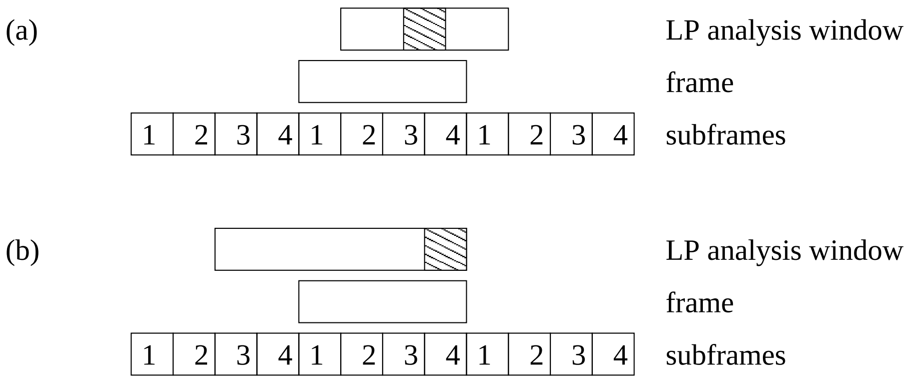


Figure 5.16: Relationship between frames, subframes, and analysis windows (after [2] and [91]): (a) with lookahead and symmetric windows; (b) with asymmetric windows and without lookahead. The hatched blocks indicate the region of emphasis by the window.

code excitation, which is then compared with the original speech. The error is perceptually weighted and the fixed and adaptive codebook search is also performed on the basis of minimising the perceptual error. The codebook which represents the long-term correlation is also adaptive. The major difference between DPCM and CELP is the use of scalar and vector codebooks, respectively. It is well known in coding theory that processing vectors is more efficient that processing scalars, hence CELP coders generally achieve similar speech quality to that of DPCM, but at lower bitrates.

Frames, Subframes and Analysis Windows

The LPC analysis is performed on frames of speech which are normally 20 ms (160 samples for 8 kHz speech) in length, while the excitation is determined over smaller blocks (5 ms or 40 samples), called *subframes*. A tapered window, such as a Hamming window, is used to reduce the effects of spectral leakage. The analysis window often overlaps into the next frame, which is called *lookahead*, and is shown in Figure 5.16(a). This is done to assist in LPC parameter interpolation, and prevents transitional problems that occur from the abrupt change between frames [91]. However, frame lookahead introduces delay, so some CELP coders, such as the AMR coder, use asymmetric Hamming windows that overlap with the previous frame, as is shown in Figure 5.16(b). In this case, an asymmetric analysis window is used that is specifically shaped to emphasise a certain region of the frame. In the case of the AMR speech coder, the window emphasises the fourth subframe[8] [2]. The LPC parameters for the first, second, and third subframe are then interpolated.

LPC Analysis and Filtering

For each frame of speech, a 10th order LPC analysis is performed to obtain the LPC coefficients, $\{a_k\}_{k=1}^{10}$, and these are often pre-processed using techniques such as bandwidth expansion [125] and high frequency compensation [15], which we will describe in Section 5.3.5. The autocorrelation method is normally used due to the availability of computationally efficient algorithms such as the Levinson-Durbin recursion, as well as the guaranteed stability of the synthesis filter. The LPC coefficients are then transformed to another LPC parameter representation (for more details, see Section 5.4) and quantised (for more details, see Section 5.5).

For the synthesis stage, the quantised LPC parameters are converted back to LPC coefficients, $\{\hat{a}_k\}_{k=1}^{10}$. The LPC synthesis filter, $H(z)$, is given by:

$$H(z) = \frac{1}{1 + \sum_{k=1}^{10} \hat{a}_k z^{-k}} \tag{5.34}$$

Perceptual Weighting Filter

The purpose of the perceptual weighting filter is to appropriately bias the error spectrum such that less emphasis is placed in minimising the quantisation noise in the formant regions. In effect, perceptual weighting tends to shift quantisation noise from frequency regions where the speech power is low (spectral valleys) to regions where the power is high

[8]In the highest bitrate (12.2 kbps) mode, the AMR coder performs the LPC analysis twice, using a second Hamming window that emphasises the second subframe [2].

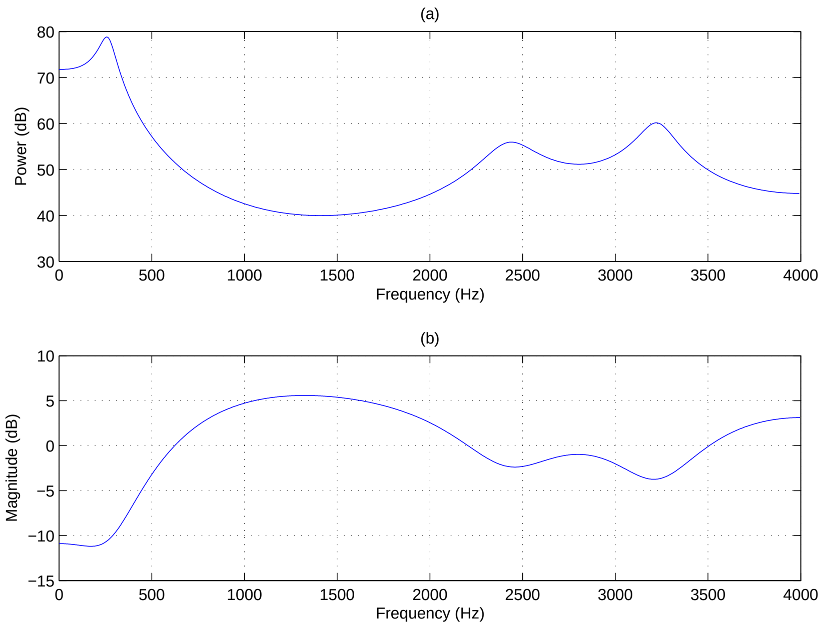


Figure 5.17: Frequency response of a perceptual weighting filter for a frame of speech: (a) power spectral density of the speech frame; (b) frequency response (in dB) of the corresponding perceptual weighting filter, $W(z)$, with $\gamma_1 = 0.9$ and $\gamma_2 = 0.6$.

(formants). This exploits auditory masking, where strong tones mask noise and other tones within the spectral vicinity [15].

The perceptual weighting filter, $W(z)$, is given by [91]:

$$W(z) = \frac{A\left(\frac{z}{\gamma_1}\right)}{A\left(\frac{z}{\gamma_2}\right)} \tag{5.35}$$

where $0 < \gamma_2 < \gamma_1 \leq 1$, $A(z)$ is the LPC analysis filter, and $A\left(\frac{z}{\gamma}\right)$ can be expressed as [91]:

$$A\left(\frac{z}{\gamma}\right) = 1 + \sum_{k=1}^{10} a_k \gamma^k z^{-k} \tag{5.36}$$

Because the value of γ is less than one, the poles of $H(z)$ move closer to the origin, increasing the bandwidth of the spectral resonances [91].

Typical values include $\gamma_1 = 0.9$ (or, $\gamma_1 = 0.94$ for lower bitrate modes) and $\gamma_2 = 0.6$ for the AMR speech coder [2]. The frequency response of the corresponding perceptual weighting filter is shown in Figure 5.17(b), where we observe a widening of the bandwidths of each spectral resonance. The filter attenuates the error the most (-10 dB) in the region

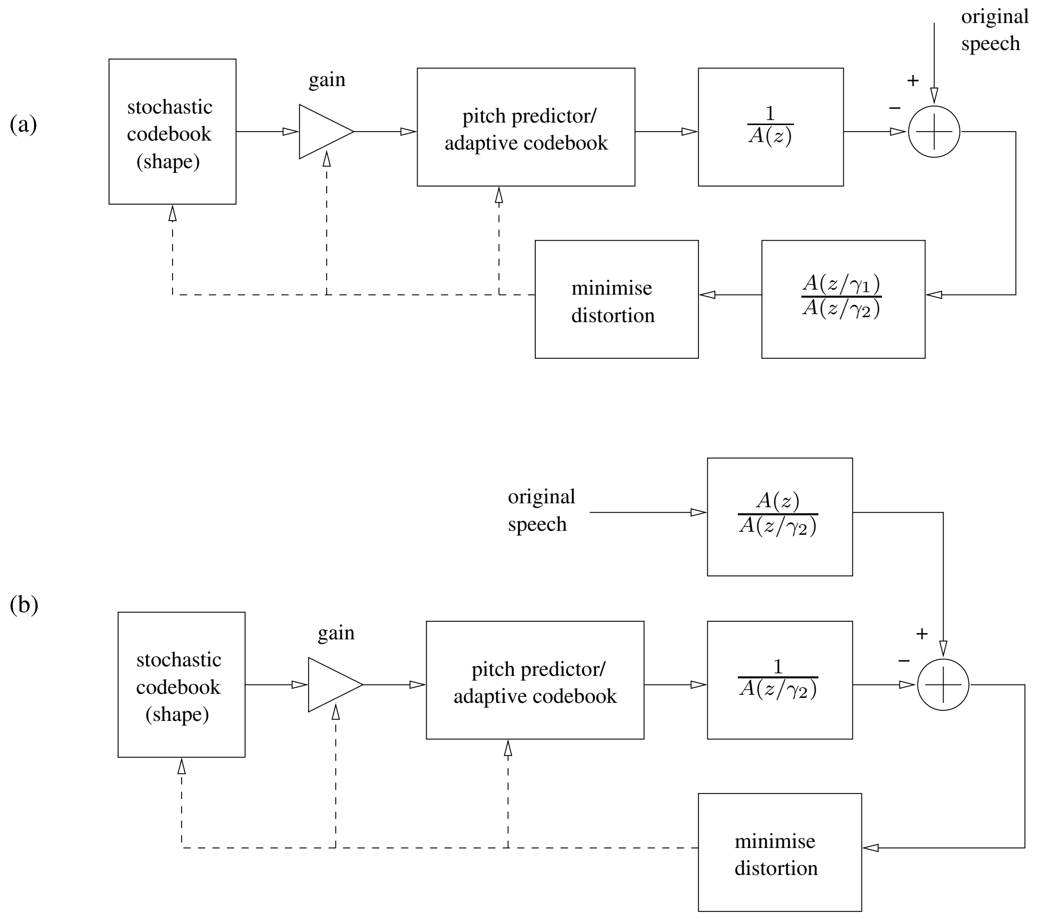

(a)
stochastic codebook (shape)
gain
pitch predictor/ adaptive codebook
$\frac{1}{A(z)}$
original speech
+
−
minimise distortion
$\frac{A(z/\gamma_1)}{A(z/\gamma_2)}$
(b)
original speech
$\frac{A(z)}{A(z/\gamma_2)}$
stochastic codebook (shape)
gain
pitch predictor/ adaptive codebook
$\frac{1}{A(z/\gamma_2)}$
+
−
minimise distortion

Figure 5.18: Comparing the typical and modified CELP coder with lower computational complexity: (a) typical configuration; (b) modified configuration with $\gamma_1 = 1.0$.

of the first formant while it amplifies the error by 5 dB within the region between 1000 and 1500 Hz. Another set of commonly used values are $\gamma_1 = 1.0$ and $\gamma_2 = 0.8$, which allows a simplification of the algorithm since the numerator of $W(z)$ cancels out denominator of the short-term LPC filter, $H(z)$. The modified coder is shown in Figure 5.18(b).

Stochastic Codebook

The stochastic codebook forms part of a shape-gain vector quantiser, which is a product code vector quantiser [55]. The codebook is fixed and contains code excitation vectors, each consisting of a subframe (eg. 40 samples) of white random Gaussian numbers, since it has been observed previously that the prediction error samples have a Gaussian PDF [153]. The code excitation vector is multiplied by an appropriate gain and passed through the pitch predictor and LPC synthesis filters, to introduce speech periodicity and the spectral envelope, respectively [153]. Through the analysis-by-synthesis procedure, the best code excitation vector and gain are chosen such that the perceptually weighted error is minimised, and the relevant indices are sent to the decoder. This is in contrast to typical vector quantisation, where the codevector is directly compared to the vector to be quantised.

Pitch Predictor/Adaptive Codebook

The purpose of this stage is to introduce long-term correlations (that is, periodicity) to the excitation vector. However, unlike the LPC vocoder, which assumes strict periodicity (in the form of impulses), the pitch predictor or adaptive codebook can construct a quasi-periodic excitation by repeating previously constructed excitations. There are two approaches that have been proposed in the literature [124]: the *pitch predictor* and *adaptive codebook*.

The pitch prediction method is the earliest approach [124] and uses the following synthesis filter, which is a third-order predictor [15, 91]:

$$\frac{1}{1 + P_d(z)} = \frac{1}{1 + \beta_1 z^{-D+1} + \beta_2 z^{-D} + \beta_3 z^{-D-1}} \tag{5.37}$$

where β_1, β_2 and β_3 are the predictor coefficients and D is the predictor delay, which is usually set to be equal to (or, a multiple of) the pitch period. Therefore, the pitch period

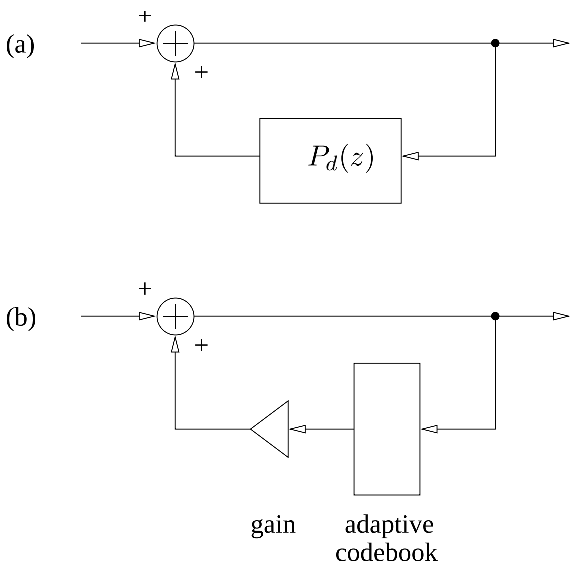


Figure 5.19: Block diagram of long-term predictors: (a) pitch predictor; (b) adaptive codebook

needs to be found either through closed-loop (analysis-by-synthesis) or open-loop analysis. As noted in [27], the pitch prediction method has problems with high pitched speakers, since the pitch period (and therefore the predictor delay, D) can become smaller than the subframe length of 5 ms. At these small delays, the effect of the stochastic excitation must be taken into account [91].

The adaptive codebook approach resolves the problems that inhibit the pitch predictor, by using a codebook of repeated and overlapping past reconstructed excitations. The pth vector in the adaptive codebook, $\boldsymbol{c}_p^{(a)}$, can be expressed as [124]:

$$\boldsymbol{c}_p^{(a)} = [e(-d(p)), \ldots, e(M-1-d(p))]^T \tag{5.38}$$

where M is the length of the subframe, $e(n)$ is the past reconstructed excitation signal, and $d(p)$ is the delay of the pth entry in the adaptive codebook. Like the stochastic codebook, the adaptive codebook has a corresponding gain factor that is multiplied with the code-vector.

Therefore, in this approach, the excitation that drives the short-term LPC filter can be said to consist of a contribution from a fixed (stochastic) codebook and an adaptive codebook. Selecting the corresponding gain factors and code excitations from each of the codebooks needs to be done in a sequential manner [124]:

1. The code-vector and gain from the adaptive codebook are determined;
2. the excitation generated from the adaptive codebook is passed through the LPC filter to generate synthesised speech;
3. the synthesised speech is subtracted from original speech to give a modified speech signal; and
4. using this modified speech signal, the code-vector and gain from the fixed stochastic codebook are determined.

5.3.5 Pre-Processing of LPC Coefficients: Bandwidth Expansion and High Frequency Compensation

Bandwidth expansion overcomes the problem with inaccurate LPC analysis for high-pitched speech (such as that from female speakers). Because of the high pitch frequency, the spectral harmonics become more widely separated, which leads to inaccurate spectral envelope estimation by the LPC analysis, such as the underestimation of formant bandwidths. The resulting synthesised speech may become unnatural and metallic sounding [125]. Bandwidth expansion is achieved by multiplying each LPC coefficient, a_k, by the factor, γ^k, which results in the following the LPC synthesis filter:

$$H(\gamma^{-1}z) = \frac{1}{1 + \sum_{k=1}^{p} a_k \gamma^k z^{-k}} \tag{5.39}$$

This has the effect of moving the poles closer toward the origin, and hence increases the bandwidth of each spectral resonance. The amount of bandwidth expansion (in Hz) is controlled by the value of γ:

$$\Delta B = -\frac{F_s}{\pi} \ln(\gamma) \tag{5.40}$$

where F_s is the sampling frequency. A typical value of γ is 0.994127 for an expansion of 15 Hz in the FS-1016 4.8 kbps CELP coder [27]. Figure 5.20 shows the effect of bandwidth expansion on the spectral envelope estimate.

Another problem that is encountered in LPC analysis is the effect of the anti-aliasing filter that is applied before the speech is sampled. For 8 kHz speech, an anti-aliasing filter with a cutoff at 3.4 kHz is used. The roll-off of the anti-aliasing filter near cutoff attenuates the high frequency parts of the speech spectrum, which causes the eigenvalues

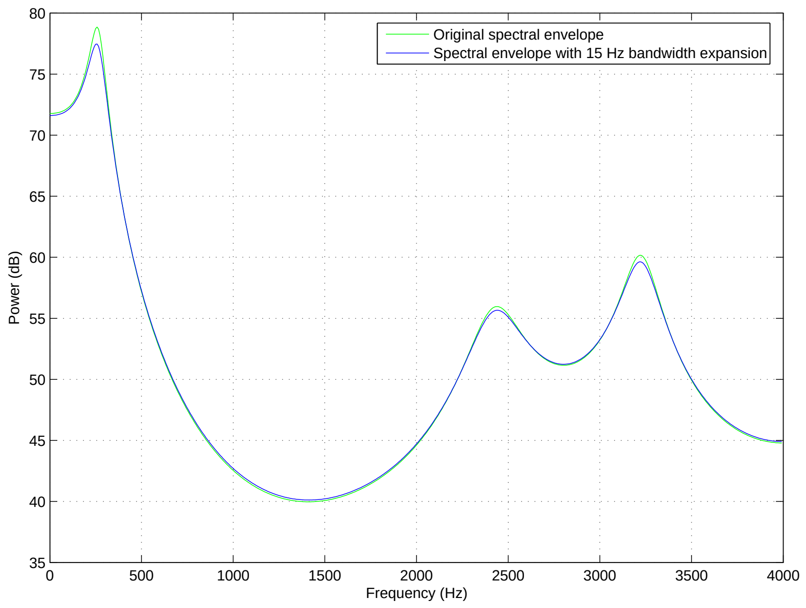


Figure 5.20: Illustrating the effect of bandwidth expansion on the spectral envelope estimate

of this region to be almost zero, leading to an ill-conditioned covariance matrix. This ill-conditioning of the covariance matrix can result in non-unique solutions for the predictor coefficients. This can be undesirable for two reasons. Firstly, multiple sets of predictor coefficients describing the same power spectral envelope leads to increased entropy, which degrades the coding performance. Secondly, some of these solutions can result in excessive power gains in the feedback loop of the LPC filter, which can also degrade the coding performance[9] [15].

High frequency compensation is a technique, first introduced by Atal and Schroeder [15], that corrects the effects of the anti-aliasing filter roll-off and avoids the ill-conditioning of the covariance matrix. Though this method was originally developed for the covariance method of LPC analysis, the procedure can be applied to the autocorrelation method as well. In high frequency compensation, the scaled covariance matrix of high-pass filtered white noise is added to the covariance matrix of the speech before analysis [15]:

$$\hat{\phi}(i,j) = \phi(i,j) + \lambda\epsilon_{min}\mu_{i-j}, \text{ (for the covariance method)} \quad (5.41)$$

$$\hat{R}(i) = R(i) + \lambda\epsilon_{min}\mu_{i}, \text{ (for the autocorrelation method)} \quad (5.42)$$

[9]The ill-conditioning of the covariance matrix is also the cause of stability problems encountered with the covariance method of LPC analysis. Thus high frequency compensation is often used along with the covariance method, which is often termed the *stabilised covariance method*.

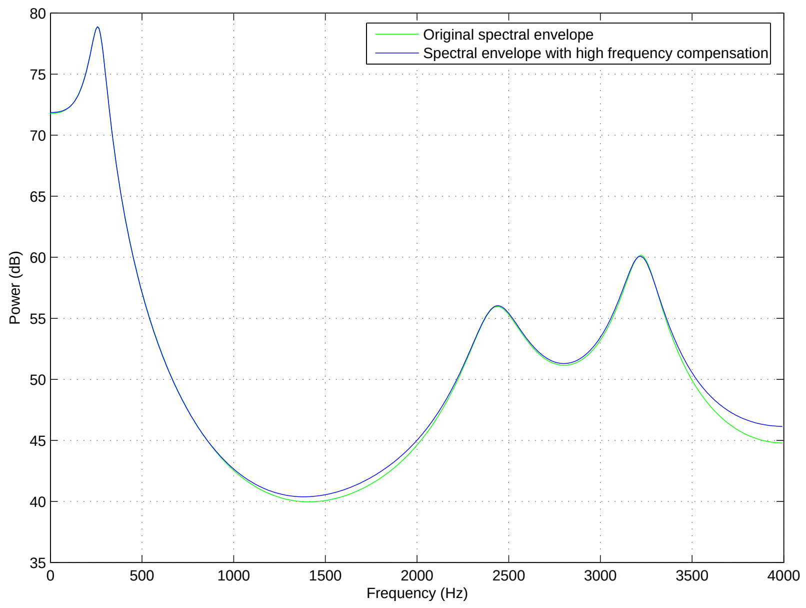


Figure 5.21: Illustrating the effect of high frequency compensation on the spectral envelope estimate

where $\hat{\phi}(i,j)$ and $\phi(i,j)$ are the compensated and original (i,j)th covariance coefficient, $\hat{R}(i)$ and $R(i)$ are the compensated and original ith autocorrelation coefficient, ϵ_{min} is the minimum prediction error, and $\mu = \left[\frac{3}{8}, -\frac{1}{4}, \frac{1}{16}, 0, 0, \ldots\right]$ are the autocorrelations of the high-pass filter, $\left[\frac{1}{2}(1 - z^{-1})\right]^2$. The value of λ is usually set to 0.1. The procedure for applying high frequency compensation can be summarised as follows:

1. Perform an LPC analysis (covariance or autocorrelation method) on the speech frame, obtaining the minimum residual error, $\epsilon_{min} = P_{min}$;

2. Correct the covariance (or, autocorrelation) matrix using (5.41) and (5.42) with $\lambda = 0.1$, $\mu_1 = 0.375, \mu_2 = -0.25, \mu_3 = 0.0625$;

3. Repeat the LPC analysis using the corrected covariance (or, autocorrelation) matrix.

Figure 5.21 shows the effect of high frequency compensation on the spectral envelope estimate. It can be seen that spectral envelope between 3400 and 4000 Hz has been 'lifted up', which coincides with the roll-off region of an anti-aliasing filter with cutoff at 3400 Hz.

5.4 LPC Parameter Representations

In the forward linear prediction-based speech coders considered thus far, the parameters representing the LPC synthesis filter for each frame need to be quantised and transmitted to the decoder. These LPC parameters must be quantised accurately in order to ensure that the reconstructed speech has good quality and intelligibility.

One important property that must be considered is the *stability* of the LPC synthesis filter. It has been found that direct quantisation of the LPC coefficients is not desirable as small quantisation errors lead to large spectral errors and can also compromise the stability of the resulting filter [125]. Another desirable property is the *natural ordering* of the coefficients, where the interchanging of any two coefficients results in a different and unique filter. Inherent ordering in the coefficients allows for more efficient coding [195]. Therefore, alternative transformations or representations of the LPC coefficients, that are both robust to quantisation as well as possessing inherent ordering and simple checks for stability, have been investigated in the literature [195, 75, 77, 23].

In the following sections, we describe some of the LPC parameter representations that have been considered for LPC speech coding. Each of the representations contain equivalent information as the LPC coefficients and are reversible.

5.4.1 Reflection Coefficients, Log Area Ratios and Arcsine Reflection Coefficients

The *reflection coefficients* (RCs), k_i, which are also known as *partial correlation* (PARCOR) coefficients, are related to the reflection properties of the vocal tract, which is modelled as having p equal length cylindrical sections of different cross-sectional areas [107]. This is shown in Figure 5.22. The reflection coefficients relate to the different cross-sectional areas, A_i, in the following way [195]:

$$A_i = A_{i+1}\frac{1+k_i}{1-k_i}, \text{ for } i = 1, 2, \ldots, p \tag{5.43}$$

where $A_{p+1} = 1$. The $\frac{A_i}{A_{i+1}}$'s are known as *area ratios.*

The reflection coefficients can be obtained as a by-product of the Levinson-Durbin recursion algorithm ($k_i = a_{i,i}$) or obtained directly from the LPC coefficients via the fol-

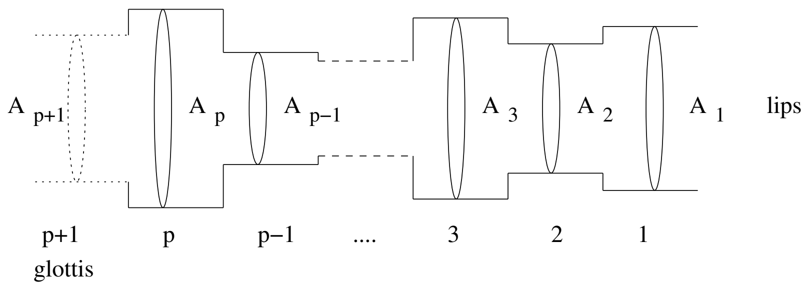


Figure 5.22: The acoustic tube model of the vocal tract

lowing backward recursion [107, 195]:

For $i = p, p-1, \ldots, 1$:

$$k_i = a_{i,i} \tag{5.44}$$

$$a_{i-1,j} = \frac{a_{i,j} - a_{i,i}a_{i,i-j}}{1-k_i^2}, \text{ where } 1 \leq j \leq i-1. \tag{5.45}$$

They can also be calculated from the autocorrelation coefficients using the Schur recursion [97]. Given below is the MATLAB code for performing the Schur recursion[10]:

```
% Inputs:  p = order of LPC analysis, rxx = p+1 autocorrelation coeffs
% Output:  rc = p reflection coeffs
function rc=schurRecur(rxx,p)
i=1:p;
rc=zeros(1,p);
g=rxx(i+1)/rxx(1);
d=g;
rch=g(1);
rc(1)=rch; % first RC
err=1-rch*rch;
for i=2:p
    for j=1:p-i+1
        g(j)=g(j+1)-rch*d(j);
        d(j)=d(j)-rch*g(j+1);
    end
```

[10]This was derived from the C source code of the FS-1016 4.8 kbps CELP coder, PC version 3.2

```
    rch=g(1)/err;
    rc(i)=rch;
    err=err*(1-rch*rch);
end
rc=-rc;
```

The reflection coefficients are naturally ordered and the filter stability is ensured if the following conditions are met:

$$|k_i| < 1, \text{ for } i = 1, 2, \ldots, p \tag{5.46}$$

When using uniform (linear) quantisation, Viswanathan and Makhoul [195] showed that, via sensitivity analyses, the reflection coefficients were not the best representation. Small variations in the reflection coefficients, whose magnitude are close to one, resulted in considerably larger spectral errors than those with a magnitude closer to zero. Therefore, finer quantisation needs to be applied for reflection coefficients close to one while coarser quantisation can be used for magnitudes close to zero.

The *log area ratios* (LARs) are formed by applying a non-linear transformation to the area ratios, which by comparison with the RCs, have sensitivity curves that are more flat:

$$g_i = \ln\left(\frac{1 + k_i}{1 - k_i}\right) \tag{5.47}$$

The filter stability is ensured if the LARs are not unbounded, hence they can have a large range, depending on the type of signal. Often in practice, the range of the LARs is artificially limited [195].

Another representation that resolves the sensitivity problems is formed by applying a non-linear transformation to the reflection coefficients, which results in the *arcsine reflection coefficients* (ASRC), which are defined as [125]:

$$j_i = \sin^{-1}(k_i) \tag{5.48}$$

Both LARs and ASRCs have similar performance and in the presence of scalar quantisation, outperform the RCs by 2 bits/frame [125].

5.4.2 Line Spectral Frequencies

The *line spectral frequencies* (LSFs), also known as *line spectral pairs* (LSPs), were introduced by Itakura [75] as an alternative LPC parameter representation. They comprise the resonant frequencies of the acoustic tube model in two extremal states, that are formed through applying artificial boundary conditions at the glottal end. These are shown in Figure 5.23.

The Closed Glottis ($k_{p+1} = 1$)

By setting the $(p+1)$th reflection coefficient, $k_{p+1} = 1$, the glottal end becomes completely closed (Figure 5.23(a)), making the acoustic tube of the lossless type [185] and possessing the following transfer function:

$$\begin{aligned} P(z) &= A(z) - z^{-(p+1)}A(z^{-1}) && (5.49)\\ &= 1 + (a_1 - a_p)z^{-1} + (a_2 - a_{p-1})z^{-2} + \ldots + (a_p - a_1)z^{-p} - z^{-(p+1)} && (5.50)\end{aligned}$$

where $A(z)$ is the analysis filter of the vocal tract. We can see that $P(z)$ is a symmetric polynomial [175].

The Open Glottis ($k_{p+1} = -1$)

Conversely, by setting the $(p+1)$th reflection coefficient, $k_{p+1} = -1$, the glottal end becomes completely open (Figure 5.23(b)), making the acoustic tube of the lossless type [185] and possessing the following transfer function:

$$\begin{aligned} Q(z) &= A(z) + z^{-(p+1)}A(z^{-1}) && (5.51)\\ &= 1 + (a_1 + a_p)z^{-1} + (a_2 + a_{p-1})z^{-2} + \ldots + (a_p + a_1)z^{-p} + z^{-(p+1)} && (5.52)\end{aligned}$$

We can see that $Q(z)$ is an anti-asymmetric polynomial and together with $P(z)$ [185]:

$$A(z) = \frac{1}{2}[P(z) + Q(z)] \qquad (5.53)$$

Both $P(z)$ and $Q(z)$ can be factorised in the following way, depending on the value of the LPC analysis order, p [185]. For an even p:

$$P(z) = (1 - z^{-1}) \prod_{k=1}^{p/2} (1 - 2\cos\omega_k^{(p)} z^{-1} + z^{-2}) \tag{5.54}$$

$$Q(z) = (1 + z^{-1}) \prod_{k=1}^{p/2} (1 - 2\cos\omega_k^{(q)} z^{-1} + z^{-2}) \tag{5.55}$$

For an odd p:

$$P(z) = (1 - z^{-1}) \prod_{k=1}^{(p-1)/2} (1 - 2\cos\omega_k^{(p)} z^{-1} + z^{-2}) \tag{5.56}$$

$$Q(z) = (1 + z^{-1}) \prod_{k=1}^{(p-1)/2} (1 - 2\cos\omega_k^{(q)} z^{-1} + z^{-2}) \tag{5.57}$$

The coefficients of $P(z)$ and $Q(z)$ are real, hence their zeros occur in complex conjugate pairs [185].

These pairs of zeros, which constitute the line spectral frequencies, have the following important properties [175]:

1. All zeros of $P(z)$ and $Q(z)$ lie on the unit circle (*spectral line property*);
2. the zeros of $P(z)$ and $Q(z)$ are interlaced with each other (*ascending order property*); and
3. the ascending ordering of the zeros ensures that the filter is stable (*minimum phase property*)

Because all the zeros of $P(z)$ and $Q(z)$ lie on the unit circle (property 1) and occur in complex conjugate pairs, they can be expressed as $p/2$ unique individual frequencies. That is, the zeros of $P(e^{j\omega})$ and $Q(e^{j\omega})$ can be represented by the frequencies (in radians), $\{\omega_i^{(p)}\}_{i=1}^{p/2}$ and $\{\omega_i^{(q)}\}_{i=1}^{p/2}$, respectively. Therefore, p LPC coefficients, $[a_1, a_2, \ldots, a_p]$, can be converted to p line spectral frequencies, $[\omega_1^{(p)}, \omega_1^{(q)}, \omega_2^{(p)}, \omega_2^{(q)}, \ldots, \omega_{p/2}^{(p)}, \omega_{p/2}^{(q)}]$ [185]. It can also be observed that LSFs are constrained within the finite range of $[0 \ldots \pi]$, unlike log area ratios which have an infinite range.

The stability of the LPC filter is guaranteed iff.:

$$0 < \omega_1^{(p)} < \omega_1^{(q)} < \omega_2^{(p)} < \omega_2^{(q)} < \ldots < \omega_{p/2}^{(p)} < \omega_{p/2}^{(q)} < \pi \tag{5.58}$$

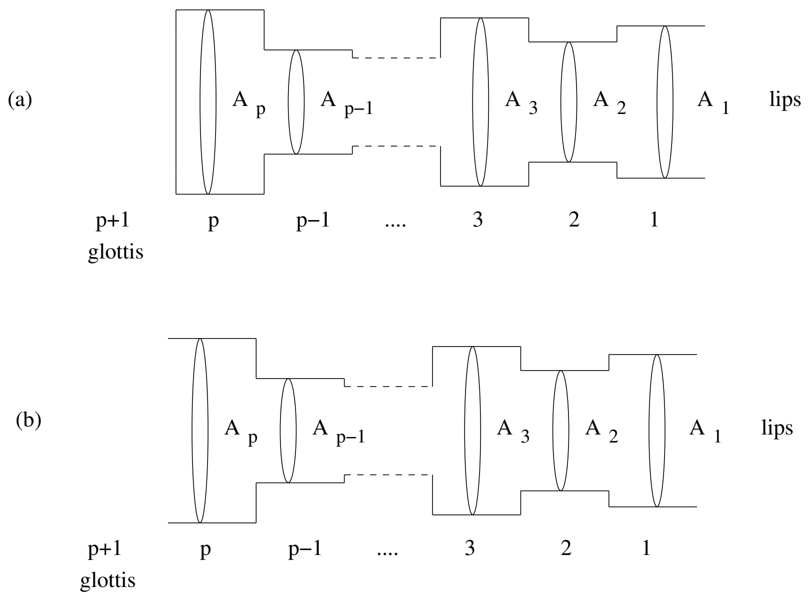


Figure 5.23: The acoustic tube with two artificial boundary conditions for the line spectral frequency representation: (a) $k_{p+1} = 1$, where the glottis is completely closed; (b) $k_{p+1} = -1$, where the glottis is completely open

Therefore, it is relatively easy to check the stability by ensuring that the LSFs are in ascending order.

Figures 5.24(a) and (b) show the zeros of $P(z)$ and $Q(z)$, respectively. It can be observed that they occur in conjugate pairs and all lie on the unit circle. Figure 5.25 shows the locations of the line spectral frequencies with respect to the spectral envelope. As noted by Sugamura and Itakura [185], the density of the LSFs are an indication of the presence of resonant formants. That is, in the vicinity of the strong formants, two or three LSFs cluster together, while in the spectral valleys, they are spread apart at almost equal intervals.

The superior quantisation performance of LSFs has been reported numerous times in the literature. Sugamura and Itakura [185] compared the scalar quantisation of LSFs with reflection coefficients. They reported that for a spectral distortion of 1 dB to be achieved, RCs required 50 bits/frame while LSFs required only 35 bits/frame, when using uniform bit allocation. With non-uniform bit allocation, RCs and LSFs required approximately 40 bits/frame and 35 bits/frame, respectively. They noted that RCs suffered from non-uniform spectral sensitivies, earlier reported by Viswanathan and Makhoul [195]. Similar scalar quantisation results were also reported in [176] and [125]. Paliwal and Atal [123]

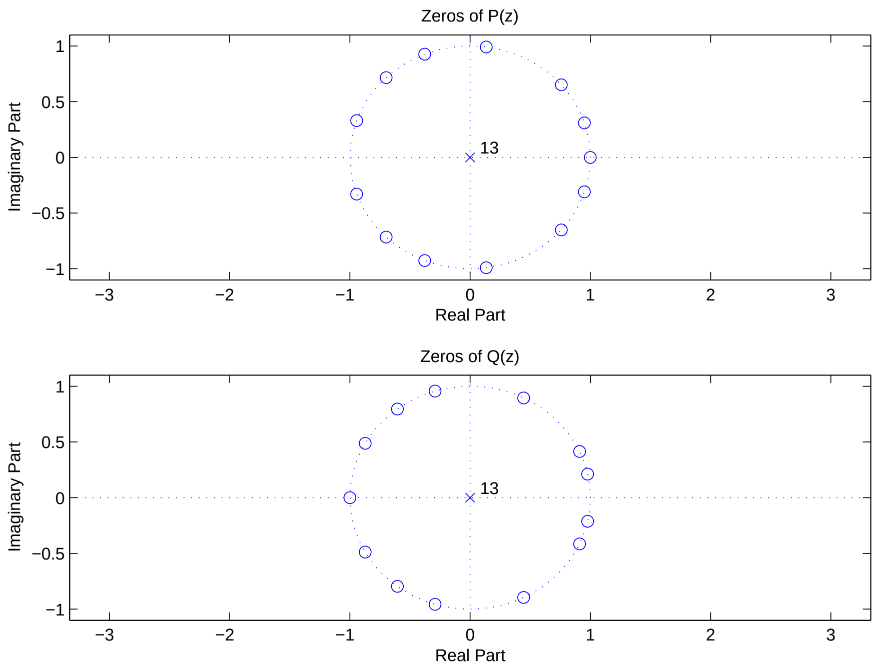


Figure 5.24: Pole-zero plot of $P(z)$ and $Q(z)$ of 12th order LPC analysis

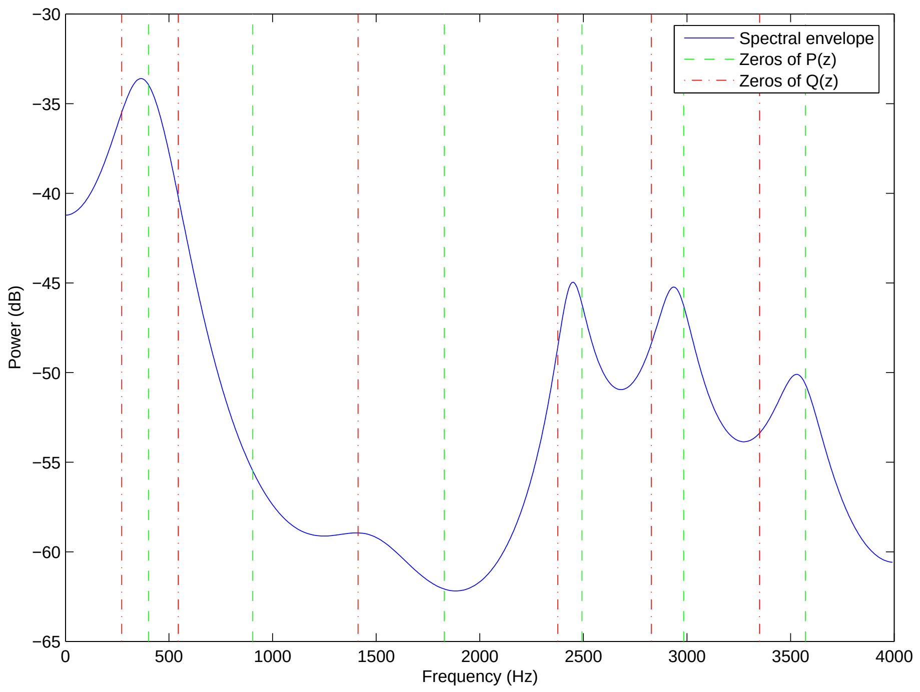


Figure 5.25: Spectral envelope estimate from a 12th order LPC analysis and line spectral frequency locations

investigated the product code vector quantisation of various LPC parameter representations, where they reported LSFs to achieve lower spectral distortion and outliers than LARs and ASRCs at 24 bits/frame.

LSFs possess quantisation properties that are favourable for speech coding. Not only do they have uniform spectral sensitivity [185], but also for each parameter, the quantisation error corresponds to spectral errors that localised in the vicinity of the LSF [123]. The FS-1016 4.8 kbps CELP coder [27] and AMR narrowband speech coder [2] use LSFs for representing the short-term information.

5.4.3 Immittance Spectral Pairs

The *immittance spectral pairs* (ISP) representation was introduced by Bistritz and Peller [23]. It consists of the poles and zeros of the following immittance[11] function at the glottis [23]:

$$\mathcal{I}_p(z) = \frac{A(z) - z^{-p}A(z^{-1})}{A(z) + z^{-p}A(z^{-1})} \tag{5.59}$$

as well as a reflection coefficient. The immittance function, $\mathcal{I}_p(z)$, can be factorised as follows [23]. For an even p:

$$\mathcal{I}_p(z) = \frac{K(1 - z^{-2})\prod_{k=1}^{p/2-1}(1 - 2\cos\omega_{2k}^{(i)} z^{-1} + z^{-2})}{\prod_{k=1}^{p/2}(1 - 2\cos\omega_{2k-1}^{(i)} z^{-1} + z^{-2})} \tag{5.60}$$

For an odd p:

$$\mathcal{I}_p(z) = \frac{K(1 - z^{-1})\prod_{k=1}^{(p-1)/2}(1 - 2\cos\omega_{2k}^{(i)} z^{-1} + z^{-2})}{(1 + z^{-1})\prod_{k=1}^{(p-1)/2}(1 - 2\cos\omega_{2k-1}^{(i)} z^{-1} + z^{-2})} \tag{5.61}$$

Because the coefficients of the immittance functions are real, the roots of both the numerator and denominator will occur in complex conjugate pairs. Therefore, there are a total of $p-1$ poles and zeros (excluding the zeros and/or poles at -1 and 1), which lie on the unit circle, and along with a 'constant' gain, which is expressed as the pth reflection coefficient:

$$k_p = \frac{K - 1}{K + 1} \tag{5.62}$$

[11]This is formed from the two words, *im*pedance and ad*mittance* [23].

constitute the p parameter ISP representation, $[\cos\omega_1^{(i)}, \cos\omega_2^{(i)}, \ldots, \cos\omega_{p-1}^{(i)}, k_p]$. Alternatively, the poles and zeros can be transformed to frequencies, $[\omega_1^{(i)}, \omega_2^{(i)}, \ldots, \omega_{p-1}^{(i)}, \frac{1}{2}\cos^{-1} k_p]$, which are sometimes known as *immittance spectral frequencies* (ISFs) [3].

The ISP representation possesses similar properties to the LSFs, namely that the roots are interlaced and lie on the unit circle. The stability of the LPC filter is guaranteed via the ascending order of the ISPs along with the further condition that the reflection coefficient, $|k_p| < 1$ [23].

Comparing the immittance function, defined in (5.59), with the polynomials $P(z)$ and $Q(z)$ of the LSF representation, defined in (5.55) and (5.55), it can be seen that the ratio of the LSF polynomials is equivalent to the immittance function of order $(p+1)$, $\mathcal{I}_{p+1}(z)$, that is obtained by extending the LPC polynomial, $A(z)$, with a zero at the origin [23].

The literature on the quantisation performance of ISPs is less than that of LSFs, though Bistritz and Peller [23] report a 1 bit saving, when using ISPs compared with LSFs, in differential scalar quantisation experiments. ISPs are used for representing short-term information in the AMR wideband speech coder [19].

The use of block and vector quantisation of ISPs may need to be handled in a special way as the ISP representation consists of a concatenation of two different variables: $p-1$ frequencies and a reflection coefficient. Each will have their own unique quantisation characteristics and sensitivity. This is in contrast to the LSFs, which are all of the same type (frequency). Therefore, it is customary to quantise ISFs, where an arc-cosine is applied to the reflection coefficient which tends to flatten its sensitivity curve. In our wideband LPC parameter quantisation experiments in Chapter 6, we find that LSFs are superior to ISFs in block and vector quantisation schemes. In light of this observation, we will investigate the quantisation of LSFs for narrowband speech coding.

5.5 Quantisation of LPC Parameters

As we have mentioned previously, all forward linear predictive speech coders require accurate quantisation of the LPC parameters. Assuming a tenth order linear prediction analysis at a rate of 50 frames/second, a linear predictive speech coder requires the transmission of 500 floating point values every second in order to convey the short-term correlation information to the decoder. Scalar quantising the LPC parameters at 20 to 40 bits results in a bitrate of 1 to 2 kbps for encoding the spectral envelope alone, hence it is a major contributor to the overall bitrate of low-rate speech coders [125]. In fact, for bitrates less than 3 kbps, it is common for about half the total number of bits to be allocated to LPC parameter quantisation [163]. Therefore, the efficient coding of LPC parameters has been a problem of interest in low bitrate speech coding research.

In this section, we examine the various strategies that have been investigated in the speech coding literature for efficiently quantising LPC parameters. We also present some results on the performance of each scheme on LSF vectors from the TIMIT database. We have chosen the LSF representation because, as mentioned in the previous section, our wideband LPC parameter quantisation experiments (in Chapter 6) suggest that LSFs are superior to ISFs, in joint vector quantisation schemes. For more details on the experimental setup which will be used in results throughout the rest of the chapter, the reader should refer to Section 5.5.2. Firstly, it is important to review the methods used to evaluate the performance of different quantisation schemes as well as defining an appropriate distance measure for quantiser design.

5.5.1 LPC Parameter Distortion Measures

Spectral Distortion and Transparent Coding

A popular method for objectively evaluating the quantisation performance of LPC parameters is the use of the average *spectral distortion* of all frames. The full-band spectral distortion of each frame is defined as the root mean squared error between the power spectral density estimate of the original and reconstructed LPC parameter vector [127]:

$$D_{sd}(i) = \sqrt{\frac{1}{F_s}\int_0^{F_s}\left[10\log_{10} P_i(f) - 10\log_{10}\hat{P}_i(f)\right]^2 df} \tag{5.63}$$

where F_s is the sampling frequency and $P_i(f)$ and $\hat{P}_i(f)$ are the LPC power spectra of the reconstructed and original ith frame, respectively.

Another form of spectral distortion is termed the *partial-band* spectral distortion, which was defined by Atal *et al.* [17]:

$$D_{psd}(i) = \sqrt{\frac{1}{F_2 - F_1} \int_{F_1}^{F_2} \left[10 \log_{10} P_i(f) - 10 \log_{10} \hat{P}_i(f)\right]^2 df} \qquad (5.64)$$

where F_1 and F_2 is equal to 0 and 3 kHz for narrowband speech, respectively. The partial-band spectral distortion is normally used for evaluating quantisation schemes that use a weighted distance measure.

Transparent coding means that the coded speech is indistinguishable from the original speech through listening tests. The following conditions for transparent coding from LPC parameter quantisation have been used in the literature [127]:

1. The average spectral distortion (SD) is approximately 1 dB;
2. there is no outlier frame having more than 4 dB of spectral distortion; and
3. less than 2% of outlier frames are within the range of 2–4 dB.

It should be noted that these conditions for transparent coding were determined using waveform-matching coders such as multi-pulse and CELP speech coders. For parametric speech coders (eg. LPC and sinusoidal vocoders), which generally have lower reconstructed speech quality, other conditions for transparent coding may be required.

Weighted Euclidean Distance Measure

The following weighted distance measure was introduced by Paliwal and Atal [123] to replace the mean squared error (MSE) in vector quantiser design and operation. The weighting consists of:

1. fixed or static weights, $\{c_i\}$, which place emphasis on the lower LSFs in order to account for the difference in sensitivity of the human ear to low and high frequencies; and
2. varying or dynamic weights, $\{w_i\}$, which emphasise the LSFs where the power spectral density is higher (ie. formant regions).

The weighted distance measure, $d_w(\boldsymbol{f}, \hat{\boldsymbol{f}})$, between the original vector, $\boldsymbol{f}$, and the approximated vector, $\hat{\boldsymbol{f}}$, is defined as [123]:

$$d_w(\boldsymbol{f}, \hat{\boldsymbol{f}}) = \sum_{i=1}^{10} \left[c_i w_i (f_i - \hat{f}_i) \right]^2 \tag{5.65}$$

where f_i and $\hat{f}_i$ are the ith LSF in the original and approximated vector respectively. The dynamic weights, $\{w_i\}$ are given by [123]:

$$w_i = [P(f_i)]^r \tag{5.66}$$

where $P(f)$ is the LPC power spectral density and r is a constant (typical value used is 0.15). The static weights, $\{c_i\}$, are given by [123]:

$$c_i = \begin{cases} 1.0, & \text{for } 1 \leq i \leq 8 \\ 0.8, & \text{for } i = 9 \\ 0.4, & \text{for } i = 10 \end{cases} \tag{5.67}$$

In order to see the benefit of the dynamic weights, Figure 5.26 shows the original and reconstructed spectral envelope estimates from shifting two different LSFs. In Figure 5.26(a), the 9th LSF, which falls on top of a formant, has been shifted and we can see that the distortion in the spectrum is localised. In fact, the position of the formant has followed the shift. When we shift an LSF that is not in the near vicinity of a formant, as shown in Figure 5.26(b), we can see that the effects are not as dramatic as in the first case. The spectral distortion of the second case is also much less. Therefore, the dynamic weights emphasise the LSFs that are situated near the peaks in the power spectral density, so that they are more finely quantised.

5.5.2 Experimental Setup for LSF Quantisation Experiments

The speech database used for evaluating the various LSF quantisation experiments is the DARPA TIMIT speech database [63]. The database consists of 6300 sentences in total, with 10 sentences spoken by 630 speakers that represent the eight major dialects of American English. The speech from the TIMIT database was sampled at 16 kHz so we have downsampled the speech down to 8 kHz using a 3.4 kHz anti-aliasing filter. A 20 ms

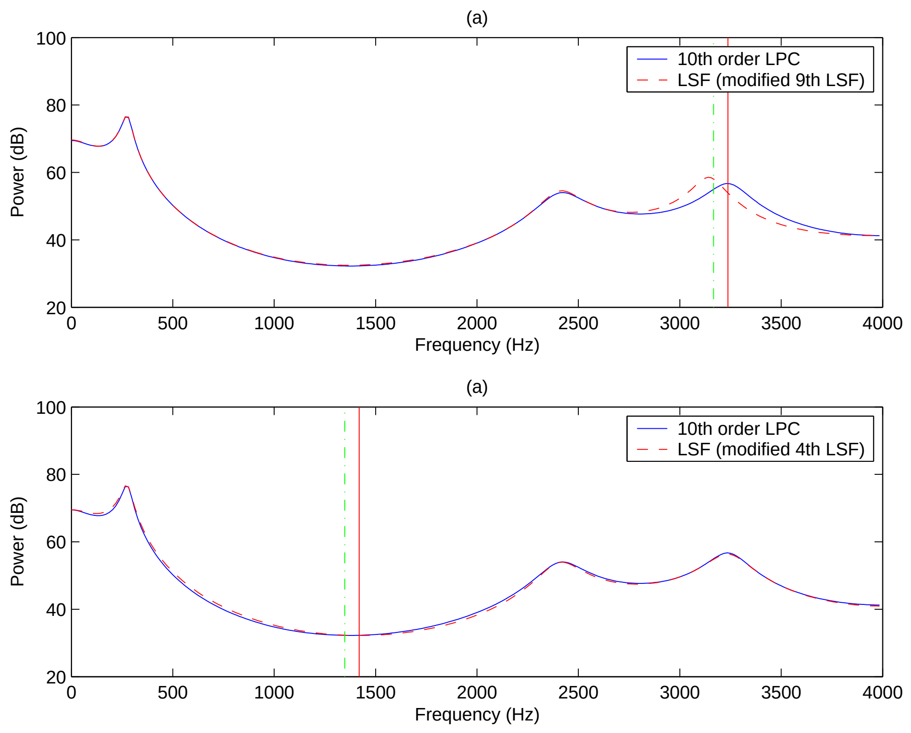


Figure 5.26: Original and reconstructed spectral envelope for a 10th order LPC analysis: (a) shifting the 9th LSF (SD=1.3944 dB); (b) shifting the 4th LSF (SD=0.4827 dB). The solid and dashed vertical lines show the original and shifted LSFs, respectively.

Hamming window is used and a tenth order linear predictive analysis is performed on each frame using the autocorrelation method [125]. There is no overlap between successive speech frames and silent frames were included. High frequency compensation and a bandwidth expansion of 15 Hz[12] was used to correct the effects of the anti-aliasing filter [15] as well as formant underestimation, respectively [91]. The training data consists of 333789 of the 707438 vectors from the TIMIT training set. The evaluation set, which contains speech that is exclusive of the training, contains all 85353 vectors.

5.5.3 PDF-Optimised Scalar Quantisers with Non-Uniform Bit Allocation

The simplest strategy is to quantise each LPC parameter using a scalar quantiser. Sugamura and Itakura [185] reported the scalar quantisation of LSFs and RCs using uniform and non-uniform bit allocation. The change from uniform to non-uniform bit allocation had the most dramatic effect on the performance of the RCs, while LSFs saw only moderate improvement. To achieve a spectral distortion of 1 dB, LSF quantisation required 33 and 35 bits/frame while RCs required 40 and 50 bits/frame for uniform and non-uniform bit allocation, respectively.

Soong and Juang [175] quantised LSF and LAR differences using differential pulse coded modulation (DPCM). The LSF representation achieved a lower likelihood ratio distortion than the LARs at a lower bitrate (30 bits cf. 43 bits). In a later paper [176], they designed globally optimal scalar quantisers with non-uniform levels and bit allocation for differential LSF quantisation. The difference between LSF frames are quantised using scalar quantisers, whose levels are designed using a Lloyd-like algorithm which minimises a non-trivial, data dependent spectral distortion [176]. A greedy bit allocation was used, where bits are allocated one at a time to each differential LSF that resulted in the largest marginal improvement in distortion. They reported a spectral distortion of 1 dB at 32 bits/frame.

Bistritz and Peller [23] evaluated uniform scalar quantisation of difference LSFs and ISPs, where a spectral distortion of 1 dB was achieved at 35 bits/frame and 34 bits/frame, respectively.

[12]This is the same high frequency compensation and bandwidth expansion contained in the source code of the US Federal Standard 1016 4.8 kbps CELP coder described in [27].

Table 5.2: Partial-band spectral distortion (SD) performance of uniform scalar quantisers operating at 34 bits/frame from the FS-1016 4.8 kbps CELP coder on the TIMIT database

Bits/frame	Avg. SD (in dB)	Outliers (in %)	
		2–4 dB	> 4 dB
34	1.44	10.24	0.02

Paliwal and Atal [123] reported results for non-uniform scalar quantisation of LSFs, ASRCs, LARs, and difference LSFs at various bitrates. Each of the scalar quantisers were designed using the LBG algorithm [100] and bit allocation was performed using the greedy algorithm of [176]. A spectral distortion of 1 dB was achieved by all representations at 34 bits/frame, with the LSF differences achieving the lowest distortion, at the expense of a higher percentage of outlier frames. Also, the LSF scalar quantiser suffered from less outliers than the other representations.

In the US Federal Standard 4.8 kbps CELP coder (FS-1016), LPC coefficients are converted to the LSF representation, before being quantised using 34 bits/frame using 10 independent, non-uniform scalar quantisers [27]. The bit allocation to the scalar quantisers is fixed at $(3, 4, 4, 4, 4, 3, 3, 3, 3, 3)$ [27]. We have evaluated the LSF quantiser of this speech coder on the TIMIT database, which is shown in Table 5.2.

Table 5.3 shows the results of a non-uniform scalar quantisation scheme on LSFs from the TIMIT database. Each scalar quantiser was designed using the generalised Lloyd algorithm and bit allocation performed using an exhaustive greedy algorithm, similar to the one in [176]. We can see that 1 dB spectral distortion is achieved at 33 bits/frame. Comparing Tables 5.3 and 5.2, we can see that using non-uniform scalar quantisers with dynamic bit allocation achieves considerably lower distortion and percentage of outliers at lower bitrates.

It is important to ensure that the LSFs are in the correct ascending order, which becomes an issue when using coarse quantisers that operate independently. A simple check of whether the quantised LSF is larger than the previous quantised LSF can be applied.

Table 5.3: Partial-band spectral distortion (SD) performance of non-uniform scalar quantisers for different bitrates on LSF vectors from the TIMIT database

Bits/frame	Avg. SD	Outliers (in %)	
	(in dB)	2–4 dB	> 4 dB
34	0.925	0.89	0.01
33	1.001	1.30	0.01
32	1.002	1.32	0.01
31	1.153	3.68	0.01

5.5.4 Product Code Vector Quantisation

It has been shown in various studies that vector quantisers perform better than scalar quantisers, in terms of the number of bits needed for the transparent coding of LPC parameters. For example, in the FS-1016 4.8 kbps CELP coder [27], a total of 34 bits are required for independent non-uniform scalar quantisers to code the LSFs of each frame [27, 123]. Whereas, extrapolating from the operating curve of unconstrained vector quantisation suggests that we need only 20 bits/frame to achieve transparent coding of these parameters [125], while high rate analysis predicts a lower bound of 23 bits/frame[13] [66]. However, it is not possible to design codebooks at these rates and in addition, the computational cost of the resulting full search, unconstrained vector quantiser is very high.

Less complex but suboptimal product code vector quantisers such as multistage and split vector quantisers (MSVQ and SVQ) have been investigated in the speech coding literature.

Split Vector Quantisers

Paliwal and Atal [122, 123] investigated the split vector quantiser (SVQ) for the coding of LPC parameters (LSFs, LARs, and ASRCs). The 10 dimensional LPC parameter vectors are split into $(4, 6)$ and $(3, 3, 4)$ for the two-part and three-part SVQ, respectively. The bits are allocated uniformly to each part, where-ever possible. The LSFs were shown to be the better representation for SVQ, in terms of achieving lower spectral distortion at a fixed bitrate. A weighted Euclidean distance measure, given by (5.65), was introduced to the vector quantiser design and operation and it was shown that this reduced the bitrate

[13]This is the lower bound for full-band spectral distortion (0–4 kHz) while for partial-band (0–3 kHz), the bound is 22 bits/frame [66].

Table 5.4: Partial-band spectral distortion (SD), computational complexity, and memory requirements (ROM) of the two-part split vector quantiser as a function of bitrate on LSF vectors from the TIMIT database

Bits/frame	Avg. SD	Outliers (in %)		kflops/	ROM
$(b_1 + b_2)$	(in dB)	2–4 dB	> 4 dB	frame	(floats)
24 (12+12)	0.943	0.54	0.00	163.8	40960
23 (12+11)	1.023	1.09	0.00	114.7	28672
22 (11+11)	1.080	1.44	0.00	81.9	20480

Table 5.5: Partial-band spectral distortion (SD), computational complexity, and memory requirements (ROM) of the three-part split vector quantiser as a function of bitrate on LSF vectors from the TIMIT database

Bits/frame	Avg. SD	Outliers (in %)		kflops/	ROM
$(b_1 + b_2 + b_3)$	(in dB)	2–4 dB	> 4 dB	frame	(floats)
26 (9+9+8)	0.892	0.55	0.00	16.4	4096
25 (9+8+8)	1.001	1.38	0.00	13.3	3328
24 (8+8+8)	1.061	1.68	0.01	10.2	2560

for transparent coding from 26 bits/frame to 24 bits/frame for the two-part SVQ on LSF vectors. At 24 bits/frame, the two-part SVQ with weighted distance measure achieved a spectral distortion of 1.03 dB. For the three-part SVQ, which has a considerably smaller computational complexity, 25 bits/frame were need to achieve transparent coding (with a spectral distortion of 1.05 dB) [123].

Table 5.4 shows the spectral distortion performance, computational complexity, and memory requirements of the two-part split vector quantiser with weighted distance measure, on LSF vectors from the TIMIT database. We can see that transparent coding is achieved at 23 bits/frame. However, the computational complexity and memory requirements are quite high. Table 5.5 shows the spectral distortion performance of the three-part split vector quantiser, where transparent coding is achieved at 25 bits/frame. The loss of performance is mostly due to the extra vector split, resulting in a loss of the vector quantiser advantages. However, the computational complexity of the three-part SVQ is considerably less than the two-part SVQ.

Sinervo *et al.* [163] investigated two-part and three-part split vector quantisation of LSFs with different bit allocations and splitting schemes. The splitting was performed on the basis of intraframe correlation. That is, LSFs that are weakly correlated should

Table 5.6: Partial-band spectral distortion (SD), computational complexity, and memory requirements (ROM) of the two-stage multistage vector quantiser as a function of bitrate on LSF vectors from the TIMIT database

Bits/frame	Avg. SD (in dB)	Outliers (in %)		kflops/ frame	ROM (floats)	Unstable frames
		2–4 dB	> 4 dB			
24	0.908	0.68	0.00	327.7	81920	12
23	0.967	1.06	0.00	245.8	61440	7
22	1.038	1.58	0.00	163.8	40960	8

be split into different subvectors. They noted that the best bit allocation was dependent on the type of splitting used [163]. However, from a computational and memory point of view, non-uniform bit allocation is not a preferred solution since the size of the codebook increases exponentially as more bits are given. The $(4, 6)$ split was found to perform quite well with uniform bit allocation [163].

Multistage Vector Quantisers

Paliwal and Atal [123] investigated the use of multistage vector quantisation (MSVQ) on various LPC parameters. A two-stage MSVQ was found to also benefit from the weighted Euclidean distance measure and achieved a spectral distortion of 0.99 dB at 25 bits/frame, which was 1 bit/frame more than the SVQ.

Table 5.6 shows the spectral distortion and complexity of the two-stage multistage vector quantiser on LSF vectors from the TIMIT database. The weighted Euclidean distance measure was used in this experiment. We can see that transparent coding is achieved at 22 bits/frame. However, the computational complexity and memory requirements of the two-stage MSVQ are considerably higher than the two-part SVQ, as the codebooks are of the same dimension of 10.

LeBlanc *et al.* [21, 93] introduced a multistage vector quantisation scheme that was more efficient in terms of better distortion performance. Codebook searching in normal MSVQ is performed in an independent fashion which leads to suboptimal quantisation performance. Using an M-L searched MSVQ, where the multistage codebooks are searched along M paths and the one which gives the least distortion is chosen, they showed that the M-L search achieved a performance that was close to an optimal search for a relatively small M [93]. A spectral distortion of 1 dB can be achieved at bitrates as low as 22 bits/frame using two-stage MSVQ (4096-2)[14] with $M = 2$, though the complexity of the scheme is quite high (246 kflops/frame, 40960 floats). By using a four-stage MSVQ (64-4), the complexity can be considerably reduced with transparent coding reported to be achieved at 24 bits/frame using $M = 2$, with a computational complexity of 18 kflops/frame and 2560 floats [183].

Filter Stability Checking in Product Code Vector Quantisers

Unconstrained vector quantisers that operate on entire vectors are able to preserve the ordering property of the LSFs. Ordered LSFs can be visualised as points in a 10-dimensional vector space that are constrained above a hyperplane. This is shown in Figure 5.27, which shows the first and second LSFs in a two-dimensional space. We can see that, because of the ordering property, the LSFs only occur above the hyperplane where both LSFs are equal. Due to the space-filling and shape advantages, vector quantisers can represent this space accurately. However, this is not true for product code vector quantisers since they consist of independent quantisers, which in theory, operate outside the influence of the other quantisers. For example, consider the two-part SVQ operating on the two-dimensional LSFs of Figure 5.27. Each dimension would be quantised by a scalar quantiser, which due to the lower dimensionality, is not aware of the ordering constraint and will place quantiser levels in the region below the hyperplane. Thus at low bitrates, where the scalar quantisers become coarse, it is possible for codepoints below the hyperplane to be selected, thus producing quantised LSFs that violate the ascending order property.

For two-part SVQ, a stability check can be included during the search of the second subvector codebook, ensuring that the first quantised LSF is larger than the last quantised LSF from the code-vector selected in the first subvector codebook. In three-part SVQ, it is preferable to quantise using the second subvector codebook first. Then the first and third subvector codebook are searched and checked with the second to ensure the correct LSF ordering.

For multistage vector quantisers, where each stage is independent of the previous stage, the ordering of the LSFs cannot be easily checked. This is because subsequent stages are quantising residual vectors, which do not necessarily exhibit consistent ordering. However,

[14](*codebook size, number of stages*)

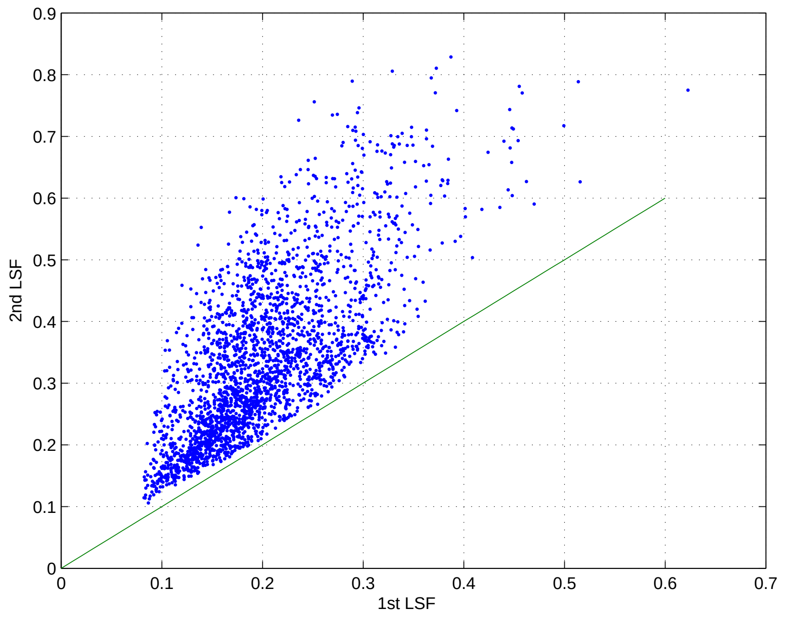


Figure 5.27: Scatter diagram of the first and second line spectral frequencies. The line is the hyperplane where both LSFs are equal.

the ordering of the quantised LSF vector can be checked when using the M-L searched MSVQ, since after the searching, M candidates are available for selection. In addition to the minimum distortion criteria, an extra criteria can be added to check whether the candidate vector is correctly ordered.

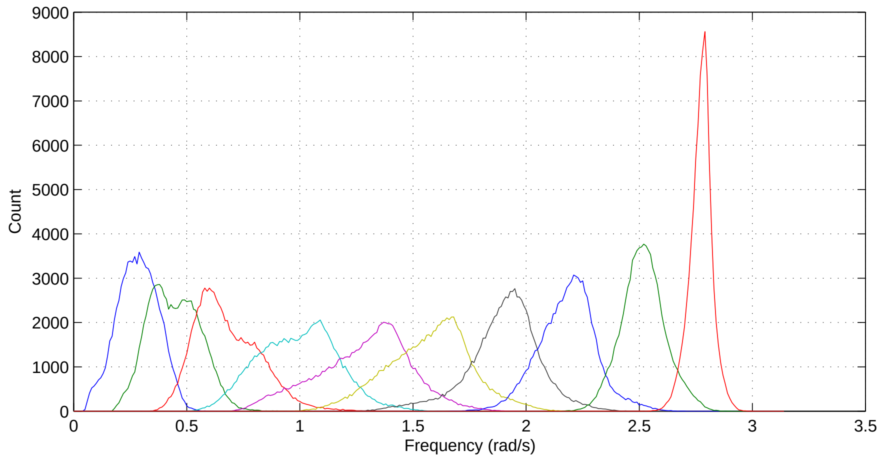


Figure 5.28: Histograms of 10 line spectral frequencies for narrowband speech

5.5.5 GMM-Based Block Quantisation

Memoryless GMM-Based Block Quantisers

Subramaniam and Rao [183] applied the GMM-based block quantiser to quantise speech LSFs. The distortion measure used for the minimum distortion block quantisation was spectral distortion. Figure 5.28 shows the histograms of the line spectral frequencies from the test set of the TIMIT database. We can see that higher frequency LSFs tend to be unimodal, while the lower frequency LSFs have a multimodal distribution. Therefore, the GMM-based block quantiser[15] is expected to perform better than the single Gaussian block quantiser, because of the former's ability to estimate and quantise the multimodal PDFs of the lower frequency LSFs, which are perceptually more important.

Subramaniam and Rao reported a spectral distortion of 1.0295 dB at a fixed-rate of 24 bits/frame [183]. However, the percentage of outlier frames having a spectral distortion between 2 and 4 dB was 3.05%, which is higher than the 2% required for transparent coding.

Table 5.7 shows the spectral distortion performance of the fixed-rate GMM-based block quantiser on LSF vectors from the TIMIT database. The GMM consists of 16 clusters and during the training process, we have used 20 iterations of the EM algorithm. We can

[15]Note that we have not evaluated the GMM-DCT-based block quantiser for LPC parameter quantisation because LSF and ISF data are only lightly correlated and do not possess Gauss-Markov properties, unlike image data. We have found the DCT to be much less effective than the KLT on LSF and ISF data.

Table 5.7: Average spectral distortion (SD) performance of the 16 cluster, memoryless, fixed-rate GMM-based block quantiser using spectral distortion criterion at different bitrates on LSF vectors from the TIMIT database

Bits/frame	Avg. SD (in dB)	Outliers (in %)	
		2–4 dB	> 4 dB
25	0.981	0.76	0.00
24	1.042	1.12	0.00
23	1.107	1.66	0.00

Table 5.8: Bitrate independent computational complexity (in kflops/frame) and memory requirements (ROM) of the GMM-based block quantiser using spectral distortion-based quantiser selection as a function of the number of clusters

m	kflops/frame	ROM (floats)
4	63.5	776
8	126.9	1296
16	253.9	2336
32	507.7	4416

observe that transparent coding is achieved at 25 bits/frame on the TIMIT database.

By comparing Tables 5.7 and 5.3, we can see that the 25 bits/frame GMM-based block quantiser is comparable to the 33 bits/frame PDF-optimised scalar quantisers. Both quantisation schemes utilise scalar codebooks that are optimised for the PDF of each LSF. Therefore, it is apparent that the saving of up to 8 bits/frame is mostly due the decorrelating aspect of the multiple Karhunen-Loève transforms in the GMM-based block quantiser.

Table 5.8 shows the computational complexity and memory requirements of the GMM-based block quantiser, when using the spectral distortion to select the appropriate block quantiser. These were calculated using Table 2.1 with $n_{dist} = 15.265$ kflops/frame for spectral distortion[16] and Equation (2.70). We can see that the computational complexity of the GMM-based block quantiser, while it is independent of the bitrate, is quite high

[16]The spectral distortion involves calculating the root-mean-squared-error between the log periodograms of both the original and quantised LPC coefficients. Therefore, two 256-point split-radix FFTs are used, each expending 6664 flops, according to Table 6.2 of [136]. The reciprocal of the squared magnitude is calculated for each FFT, requiring an additional 516 flops, which followed by the decibel calculation, brings the total computations to 14.876 kflops. The RMS calculation adds an extra 389 flops. The number of flops required for the square root and logarithm calculation are not known and are assumed to be 1 flop, which brings the total computational complexity of the spectral distortion calculation to approximately 15.27 kflops. Note that in our work, each multiplication, addition, and comparison is considered one floating point operation (flop).

Table 5.9: Average spectral distortion of the 16 cluster predictive GMM-based block quantiser (trained case) using SD criterion as a function of bitrate on LSF vectors from the TIMIT database

Bits/frame	Avg. SD (dB)	Outliers (in %)	
		2–4 dB	> 4 dB
25	0.877	1.97	0.26
24	0.939	2.56	0.34
23	1.003	3.25	0.38
22	1.077	4.26	0.48
21	1.154	5.55	0.60

Table 5.10: Prediction coefficients for the predictive GMM-based block quantiser (calculated using the covariance method)

i	a_i
1	0.969321
2	0.983092
3	0.988771
4	0.994216
5	0.996263
6	0.997416
7	0.998717
8	0.999198
9	0.999600
10	0.999832

due to the frequent use of the spectral distortion calculation[17].

Predictive GMM-Based Block Quantisers

Subramaniam and Rao [183] also reported results for their predictive GMM-based block quantiser (for more details on this scheme, see Section 2.7.2), which exploits correlation between consecutive frames using a first-order predictor. They reported a spectral distortion of 0.9943 dB at 22 bits/frame with 2.75% of outlier frames having a spectral distortion between 2 and 4 dB.

We have used the trained case of the predictive GMM-based block quantiser to quantise LSF vectors from the TIMIT database and the spectral distortion performance is shown

[17]It should be noted that this is the full spectral distortion calculation and there exist simpler approximations, such as the one proposed in [52].

in Table 5.9. The prediction coefficients were calculated using the covariance method and are shown in Table 5.10. We can see that a spectral distortion of 1 dB is achieved at 23 bits/frame, as opposed to 25 bits/frame for the memoryless case (Table 5.7). Note that this quantisation scheme suffers from a higher percentage of outlier frames in both our results and those of [183]. This is mostly because of the poor performance of the predictor, due to the low correlation between rapidly changing LSF frames. Better performance may be expected by using a higher order predictor or a safety-net scheme [45], where each frame is quantised using the memoryless and predictive quantiser, and the one which incurs the least distortion is then chosen. Both these methods will add further complexity and delay.

Filter Stability Checking in GMM-Based Block Quantisers

Because the GMM-based block quantiser and its derivatives comprise of a set of independent scalar quantisers, which is a special case of split vector quantisation, then it is possible for unstable LSF frames to be produced. Therefore, it is necessary to add a stability check in the minimum distortion block quantisation stage to ensure that not only is the distortion minimised, but also that the reconstructed frame is stable as well. However, this does not correct the case where all cluster block quantisers produce unstable frames, though from our experience, this situation is quite rare.

5.6 LSF Quantisation Experiments

In this section, we present and discuss the results of LSF quantisation experiments using the two new quantisation schemes that have been introduced and discussed in this dissertation, namely the multi-frame GMM-based block quantiser and the switched split vector quantiser. Also included is the traditional block quantiser, which can be viewed as a single cluster, GMM-based block quantiser. This will provide a useful baseline for comparison with the other quantisation schemes.

5.6.1 The KLT-Based Block Quantiser

Table 5.11 shows the spectral distortion performance of the traditional block quantiser which uses a single KLT as the transform. In this scheme, the PDF of the LSFs is

Table 5.11: Average spectral distortion of the KLT-based block quantiser as a function of bitrate on LSF vectors from the TIMIT database

Bits/frame	Avg. SD (dB)	Outliers (in %)	
		2–4 dB	> 4 dB
31	0.911	1.23	0.03
30	0.971	1.68	0.03
29	1.026	2.12	0.06
28	1.093	2.76	0.08
27	1.167	3.80	0.11
24	1.409	9.66	0.26

assumed to Gaussian. We can see that a spectral distortion of approximately 1 dB is achieved at 29 bits/frame.

Comparing the performance with that of the scalar quantiser from the FS-1016 4.8 kbps CELP coder in Table 5.2, we can see that the block quantiser at 24 bits/frame is comparable to the 34 bits/frame scalar quantiser. Also, if we compare the block quantiser with the scalar quantiser with dynamic non-uniform bit allocation (Table 5.3), we can see that the block quantiser at 29-30 bits/frame is comparable to a 32-33 bits/frame scalar quantiser. This highlights the advantage of using a decorrelating transform before scalar quantisation. That is, there is a degree of intraframe correlation that can be exploited.

If we compare Table 5.11 with Table 5.7, we can see that the 24 bits/frame GMM-based block quantiser is roughly equivalent to a 29 bits/frame block quantiser. The saving of 5 bits/frame is mostly due to the more accurate modelling of the PDF of the LSF vectors (particularly of the lower order LSFs, as seen in Figure 5.28), in addition to the use of multiple transforms which are designed to decorrelate local vector spaces.

5.6.2 The Multi-Frame GMM-Based Block Quantiser

Using MSE Criterion for Block Quantiser Selection

Table 5.12 shows the spectral distortion performance of the 16 cluster, multi-frame GMM-based block quantiser for varying bitrates and number of concatenated frames, p. The cluster selection criterion used in the minimum distortion block quantisation stage is mean-squared-error (MSE), which is more computationally efficient than spectral distortion[18]. A

[18]Referring to Table 2.1, for MSE, $n_{dist} = 3n$ flops/frame.

Table 5.12: Average spectral distortion of the 16 cluster multi-frame GMM-based block quantiser using MSE criterion as a function of bitrate and number of concatenated frames, p

p	Bits/frame	Avg. SD (dB)	Outliers (in %)	
			2–4 dB	> 4 dB
1	22	1.199	3.39	0.01
	23	1.129	2.34	0.00
	24	1.069	1.66	0.00
	25	1.008	1.16	0.00
2	21	1.112	2.67	0.01
	22	1.047	1.84	0.00
	23	0.987	1.25	0.00
	24	0.929	0.86	0.00
	25	0.875	0.62	0.00
3	21	1.063	2.14	0.01
	22	1.001	1.42	0.00
	23	0.941	0.98	0.00
	24	0.887	0.66	0.00
	25	0.836	0.48	0.00
4	21	1.042	1.88	0.01
	22	0.982	1.33	0.00
	23	0.926	0.94	0.00
	24	0.871	0.57	0.00
	25	0.821	0.42	0.00

Table 5.13: Bitrate independent computational complexity (in kflops/frame) and memory requirements (ROM) of the multi-frame GMM-based block quantiser using MSE criterion as a function of number of concatenated vectors, p, and number of clusters, m

m	p	kflops/frame	ROM (floats)
16	1	10.09	2336
	2	16.49	7616
	3	22.89	16096
	4	29.28	27776
32	1	20.19	4416
	2	32.98	14976
	3	45.77	31936
	4	58.57	55296

spectral distortion of 1 dB is achieved at 22 bits/frame with $p = 3$. For any given bitrate, the spectral distortion decreases as more frames are concatenated together. This may be attributed to the decorrelation of LSFs within and across frames by the KLT. Because the dimension of the vectors is larger, the block quantiser can operate at a higher bitrate and therefore, the performance is expected to improve. As well as spectral distortion, the percentage of outlier frames has also decreased with an increase in p.

Comparing Table 5.12 and Table 5.7, which shows the results for the memoryless ($p = 1$) GMM-based block quantiser using 16 clusters and spectral distortion criterion, we can see that the multi-frame GMM-based block quantiser is more efficient in terms of bitrate and distortion. Comparing with Table 5.12, it can be observed that by coding two frames jointly, a spectral distortion of approximately 1 dB is achieved using 23 bits/frame while the memoryless quantiser requires 25 bits/frame. By quantising more frames ($p = 3$ and 4) jointly, 22 bits/frame are needed to achieve the same level of spectral distortion. In comparison, the 21 bits/frame multi-frame GMM-based block quantiser ($p = 4$, $m = 16$) is equivalent to the 24 bits/frame memoryless GMM-based block quantiser, in terms of spectral distortion. Therefore, the exploitation of memory across 4 successive frames by the KLT enables a saving of 3 bits/frame.

The memoryless scheme uses spectral distortion (SD) to select the appropriate block quantiser which incurs a high computational complexity, as shown in Table 5.8. Comparing this with Table 5.13, we can see that the use of MSE in the multi-frame scheme leads to a more computationally efficient scheme, along with better quantisation performance. For example, the $p = 3$ multi-frame GMM-based block quantiser has a computational

Table 5.14: Average spectral distortion of the 32 cluster multi-frame GMM-based block quantiser using MSE criterion as a function of bitrate and number of concatenated frames, p

p	Bits/frame	Avg. SD (dB)	Outliers (in %)	
			2–4 dB	> 4 dB
1	22	1.163	2.79	0.00
	23	1.099	1.95	0.00
	24	1.036	1.34	0.00
	25	0.977	0.80	0.00
2	21	1.069	1.97	0.00
	22	1.008	1.39	0.00
	23	0.950	0.94	0.00
	24	0.895	0.61	0.00
	25	0.842	0.39	0.00
3	20	1.084	2.49	0.01
	21	1.022	1.69	0.00
	22	0.963	1.11	0.00
	23	0.908	0.74	0.00
	24	0.855	0.51	0.00
	25	0.804	0.34	0.00
4	20	1.066	2.12	0.01
	21	1.006	1.40	0.00
	22	0.949	0.94	0.00
	23	0.894	0.64	0.00
	24	0.841	0.43	0.00
	25	0.792	0.25	0.00

complexity that is less than 10% of the complexity of a single-frame GMM-based block quantiser, with the former achieving transparent coding at 22 bits/frame compared with 25 bits/frame for the latter. These advantages come at the expense of higher memory requirements though.

Table 5.14 shows that when using 32 clusters, only 21 bits/frame are required for a spectral distortion of 1 dB. It can also be seen that the percentage of outliers has decreased as a result of using more clusters. This is to be expected as there are more cluster quantisers to choose from. However, the computational complexity and memory requirements of the 32 cluster scheme (Table 5.13) are much higher than those of the 16 cluster one. Hence from an implementation standpoint, a saving of 1 bit/frame may not justify the increase in complexity that accompanies the use of more clusters.

Table 5.15: Average spectral distortion of the 16 cluster multi-frame GMM-based block quantiser using SD criterion as a function of bitrate and number of concatenated frames, p

p	Bits/frame	Avg. SD (dB)	Outliers (in %)	
			2–4 dB	> 4 dB
2	21	1.091	2.13	0.00
	22	1.027	1.41	0.00
	23	0.967	0.92	0.00
	24	0.911	0.63	0.00
	25	0.858	0.42	0.00
3	21	1.046	1.84	0.01
	22	0.985	1.18	0.00
	23	0.925	0.79	0.00
	24	0.872	0.50	0.00
	25	0.823	0.36	0.00
4	21	1.027	1.63	0.00
	22	0.968	1.10	0.00
	23	0.912	0.72	0.00
	24	0.858	0.44	0.00
	25	0.810	0.32	0.00

Using Spectral Distortion Criterion for Block Quantiser Selection

Table 5.15 shows the spectral distortion performance of the multi-frame GMM-based block quantiser that uses spectral distortion[19] for determining the best cluster quantiser, as is done in the original memoryless GMM-based block quantiser of [183]. For each concatenated frame, the spectral distortion is calculated for each subframe and these are added together to give the final distortion. Because spectral distortion is used as the final objective measure, matching the distortion criteria is expected to be more optimal than a mismatched scheme (in our case, MSE with SD). This is shown when comparing the spectral distortions of Tables 5.12 and 5.15. The matched case yields spectral distortions which are roughly 0.02 dB lower than the MSE-based scheme. Results for a 32 cluster SD-based scheme, given in Table 5.16, show the SD-based scheme to be superior to the MSE-based one by approximately 0.02 dB. It would appear that the improvement in quantisation performance, as a result of using an SD-based scheme rather than an MSE-based one, is less than 1 bit/frame.

The advantage of using MSE over SD is the computational simplicity of the former

[19]We used the ‘exact’ full-band spectral distortion calculation.

Table 5.16: Average spectral distortion of the 32 cluster multi-frame GMM-based block quantiser using SD criterion as a function of bitrate and number of concatenated frames, p

p	Bits/frame	Avg. SD (dB)	Outliers (in %)	
			2–4 dB	> 4 dB
2	21	1.045	1.52	0.00
	22	0.985	1.05	0.00
	23	0.928	0.66	0.00
	24	0.874	0.43	0.00
	25	0.822	0.26	0.00
3	21	1.003	1.37	0.01
	22	0.945	0.90	0.00
	23	0.891	0.56	0.00
	24	0.838	0.37	0.00
	25	0.788	0.25	0.00
4	20	1.048	1.82	0.00
	21	0.989	1.15	0.00
	22	0.932	0.73	0.00
	23	0.879	0.47	0.00
	24	0.826	0.31	0.00
	25	0.779	0.18	0.00

Table 5.17: Bitrate independent computational complexity (in kflops/frame) and memory requirements (ROM) of the multi-frame GMM-based block quantiser using SD criterion as a function of number of concatenated vectors, p, and number of clusters, m

m	p	kflops/frame	ROM (floats)
16	1	253.9	2336
	2	276.3	7616
	3	311.5	16096
	4	359.5	27776
32	1	507.7	4416
	2	552.5	14976
	3	622.9	31936
	4	718.9	55296

Table 5.18: Partial-band spectral distortion (SD) and computational performance of the two-part switched split vector quantiser at 24 bits/frame as a function of the number of switch directions using different bit allocations

m	Total bits/frame	Avg. SD	Outliers (in %)		kflops/
	$(b_m + b_1 + b_2)$	(in dB)	2–4 dB	> 4 dB	frame
1	24 (0+12+12)	0.943	0.54	0.00	163.9
2	24 (1+12+11)	0.943	0.62	0.00	114.8
	24 (1+11+12)	**0.920**	0.45	0.00	131.1
4	24 (2+11+11)	0.924	0.56	0.00	82.1
	24 (2+10+12)	**0.917**	0.50	0.00	114.8
8	24 (3+11+10)	0.926	0.68	0.00	57.7
	24 (3+10+11)	**0.903**	0.43	0.00	65.9
16	24 (4+10+10)	0.912	0.57	0.00	41.6
	24 (4+9+11)	**0.903**	0.46	0.00	58.0

over the latter, as can be seen by comparing Tables 5.13 and 5.17. For example, the 16 cluster multi-frame GMM-based block quantiser with $p = 3$, at 24 bits/frame requires only 22.89 kflops/frame for the MSE criterion, compared with 311.5 kflops/frame for the SD criterion, though the difference in average spectral distortion between the two schemes is only 0.015 dB. Therefore, the significant reduction in computational complexity, as a result of using the MSE criterion, more than outweighs the minor gains in quantisation performance from using the spectral distortion criterion, though one may substitute the spectral distortion calculation with a simpler high-rate approximation from [52].

Table 5.19: Partial-band spectral distortion (SD) and computational performance of the three-part switched split vector quantiser at 24 bits/frame as a function of the number of switch directions using different bit allocations

m	Total bits/frame ($b_m + b_1 + b_2 + b_3$)	Avg. SD (in dB)	Outliers (in %) 2–4 dB	Outliers (in %) > 4 dB	kflops/ frame
1	24 (0+8+8+8)	1.061	1.68	0.01	10.3
2	24 (1+8+8+7)	**0.982**	0.97	0.01	8.27
	24 (1+8+7+8)	1.057	1.97	0.01	8.78
	24 (1+7+8+8)	1.019	1.15	0.01	8.78
4	24 (2+8+7+7)	1.006	1.39	0.00	6.81
	24 (2+7+8+7)	1.084	1.95	0.00	6.81
	24 (2+7+7+8)	1.031	1.55	0.00	7.32
	24 (2+8+8+6)	**0.968**	0.97	0.00	7.32
8	24 (3+8+7+6)	0.954	0.95	0.00	5.95
	24 (3+8+8+5)	**0.950**	1.01	0.00	6.97
16	24 (4+7+7+6)	**0.925**	0.75	0.00	4.73
	24 (4+8+8+4)	0.979	1.75	0.00	7.04
	24 (4+8+7+5)	0.948	1.02	0.00	5.76
32	24 (5+7+7+5)	**0.921**	0.86	0.00	4.86
	24 (5+7+6+6)	0.936	0.85	0.00	4.60
	24 (5+8+6+5)	0.959	1.20	0.00	5.63

5.6.3 The Switched Split Vector Quantiser

Performance as a Function of the Number of Switch Directions

Table 5.18 shows the spectral distortion and computational performance of the two-part SSVQ at 24 bits/frame with varying number of switching directions and different bit allocations. For each number of switching directions, m, the least spectral distortion is highlighted in bold. We can see that non-uniform bit allocation generally results in lower spectral distortion, particularly for cases where the number of bits is divisible by two. Therefore, the bit allocation scheme we have adopted, where bits are uniformly allocated where-ever possible, is not optimal in the distortion sense. However, we also observe that non-uniform bit allocation leads to an increase in the number of computations required, thus there is a distortion/computational complexity trade-off.

Table 5.19 shows the spectral distortion and computational performance of the three-part SSVQ at 24 bits/frame with varying number of switching directions and different bit allocations. For each number of switching directions, m, the least spectral distortion is

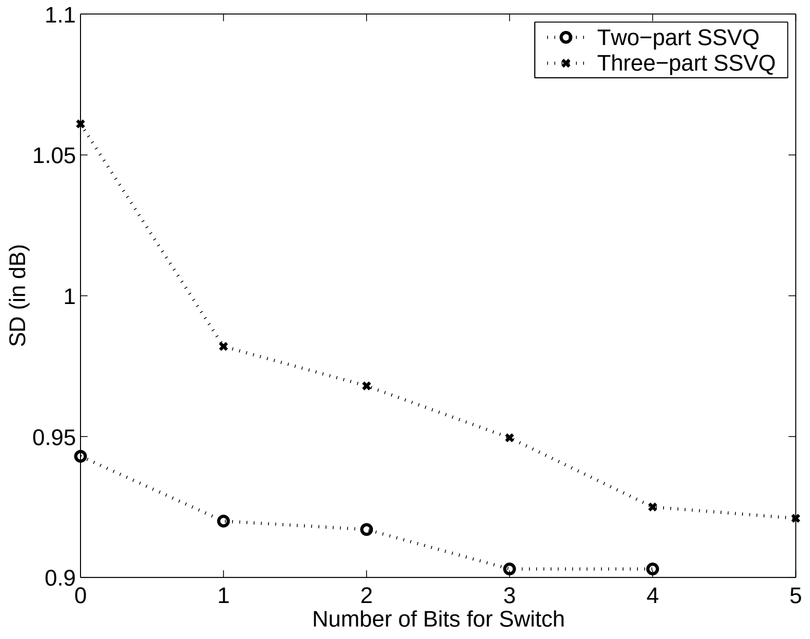


Figure 5.29: Average spectral distortion (SD) of two-part and three-part SSVQ at 24 bits/frame as a function of the bits required for switching

highlighted in bold. It can be seen that lower spectral distortion can be achieved when the first and second parts are given more bits than the third.

Another important observation to be made from Tables 5.18 and 5.19 is the consistent downward trend of the spectral distortion, as the number of switching directions is increased, which is plotted in Figure 5.29. This is due to the increased ability of the unconstrained switch vector quantiser to exploit dependencies and PDF shape as its codebook becomes larger. Also, three-part SSVQ shows larger reductions in distortion, as a result of using more switching directions, than two-part SSVQ. In order to explain this, consider a three-part split vector quantiser. Vectors are split into three parts, rather than two, and these are then independently quantised. The use of more independent quantiser stages results in more degradation in overall quantiser performance since there is less exploitation of full vector dependencies. SSVQ compensates this loss of quantiser performance, caused by more vector splitting, by using the unconstrained switch vector quantiser to exploit full vector dependencies before coding using split vector quantisers. As more bits are diverted away from the SVQs to the switch vector quantiser, it is expected that compensating the suboptimality of three-part SVQ with a full-vector VQ leads to larger performance improvements than compensating a two-part SVQ with a full-vector VQ.

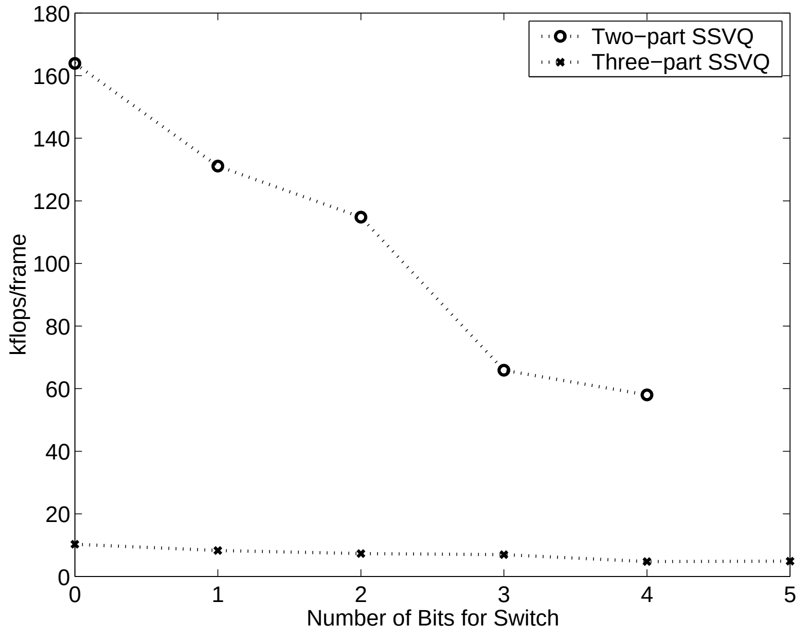


Figure 5.30: Computational complexity of two-part and three-part SSVQ at 24 bits/frame as a function of the bits required for switching

As well as this, there is a reduction in the number of computations required, which is shown in Figure 5.30. By using three-part split vector quantisers in the SSVQ, we can reduce the number of computations by as much as one tenth in exchange for only a modest increase in spectral distortion. Therefore, for the same bitrate, SSVQ achieves gains in both distortion and computational complexity. We should note that there is a saturation of the distortion for large values of m because there are an insufficient number of training vectors.

Performance as a Function of Bitrate

Table 5.20 shows the spectral distortion and computational performance of the two-part switched split vector quantiser as a function of bitrate. It can be observed that transparent coding was achieved using 22 bits while requiring 17.7 kflops for each LSF frame. Table 5.21 shows the performance of the three-part SSVQ where transparent coding was achieved using 23 bits/frame.

Also shown in Tables 5.20 and 5.21 are the memory requirements (ROM) of SSVQ. It can be observed that while SSVQ achieves gains in quantiser performance with reduced computational complexity, they are at the expense of larger memory requirements.

Table 5.20: Partial-band spectral distortion (SD), computational complexity, and memory requirements (ROM) of the two-part switched split vector quantiser as a function of bitrate and number of switch directions

m	Total bits/frame	Avg. SD	Outliers (in %)		kflops/	ROM
	$(b_m + b_1 + b_2)$	(in dB)	2–4 dB	> 4 dB	frame	(floats)
8	24 (3+10+11)	0.902	0.43	0.00	65.9	131152
	23 (3+10+10)	0.973	0.84	0.00	41.3	82000
	22 (3+9+10)	1.031	1.20	0.00	33.1	65616
16	24 (4+10+10)	0.912	0.57	0.00	41.6	164000
	23 (4+9+10)	0.965	0.78	0.00	33.4	131232
	22 (4+9+9)	1.038	1.29	0.00	21.1	82080
32	24 (5+9+10)	0.903	0.52	0.00	34.0	262464
	23 (5+9+9)	0.972	0.94	0.00	21.8	164160
	22 (5+8+9)	1.029	1.26	0.00	17.7	131392

Table 5.21: Partial-band spectral distortion (SD), computational complexity, and memory requirements (ROM) of the three-part switched split vector quantiser as a function of bitrate and number of switch directions

m	Total bits/frame	Avg. SD	Outliers (in %)		kflops/	ROM
	$(b_m + b_1 + b_2 + b_3)$	(in dB)	2–4 dB	> 4 dB	frame	(floats)
8	24 (3+7+7+7)	0.955	0.90	0.00	5.44	10320
	23 (3+7+7+6)	1.006	1.21	0.00	4.41	8272
	22 (3+7+6+6)	1.112	2.46	0.01	3.64	6736
16	24 (4+7+7+6)	0.925	0.75	0.00	4.73	16544
	23 (4+7+6+6)	1.002	1.38	0.00	3.96	13472
	22 (4+6+6+6)	1.080	1.97	0.01	3.20	10400
32	24 (5+7+7+5)	0.921	0.86	0.00	4.86	28992
	23 (5+6+6+6)	0.991	1.13	0.00	3.84	20800
	22 (5+6+6+5)	1.049	1.70	0.00	3.32	16704

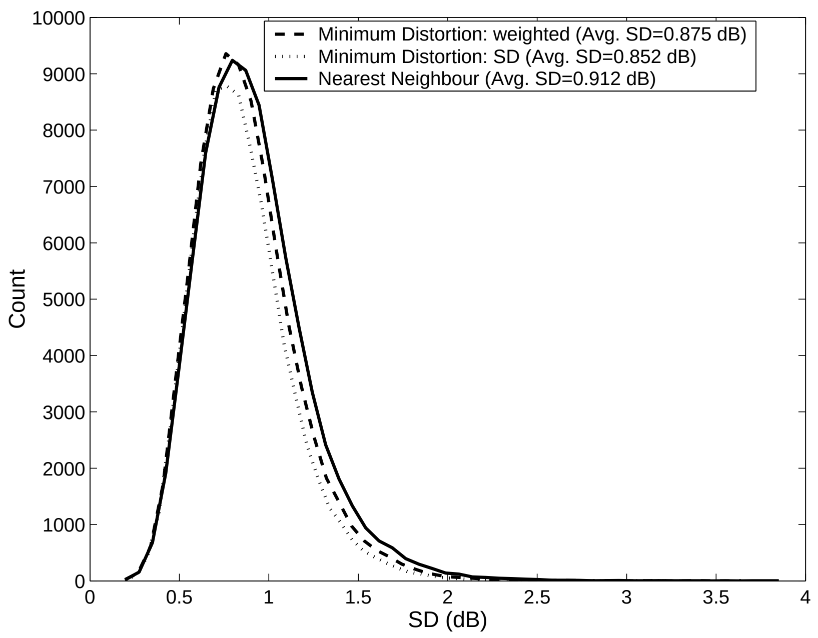


Figure 5.31: Partial-band spectral distortion (SD) histograms for the 24 bits/frame two-part switched split vector quantiser ($m = 16$) using minimum distortion (using weighted distance measure and spectral distortion) and nearest neighbour selection

Three-part SSVQ with 8 switching directions at 23 bits/frame is therefore, the best coder configuration with a good balance of low spectral distortion (1.006 dB), computational complexity (4.41 kflops/frame) and memory requirements (8272 floats).

Minimum Distortion Versus Nearest Neighbour Selection

Better performance, in terms of spectral distortion, can be achieved by quantising a vector using all split vector quantisers and then choosing the one which incurs the least distortion. However, such a scheme would involve a large amount of computations as each vector to be coded needs to be quantised multiple times in order to find the best representation. Therefore, we adopted a classification approach where each Voronoi region, from which split vector quantisers are designed, is represented by its centroid, and a switching decision, based on the nearest neighbour criteria, is made before it is quantised by the corresponding split vector quantiser. We have argued that the penalty in spectral distortion would be more than offset by the reduction in computational complexity. Figure 5.31 shows the histograms of the spectral distortion from SSVQ operating at 24 bits/frame using minimum distortion (weighted distance measure and spectral distortion) and nearest neighbour selection. As expected, the spectral distortion version of minimum distortion selection gives the lowest distortion (0.852 dB). However the resulting quantiser is very computationally intensive since m spectral distortion calculations need to be performed for each vector to be coded. The weighted MSE version of minimum distortion selection gave a slightly higher average spectral distortion (0.875 dB), while the spectral distortion of the nearest neighbour selection was the worst out of the three (0.912 dB). However, when we compare the number of computations required in minimum distortion selection (approx. 655 kflops/frame) compared with the nearest neighbour selection (42 kflops/frame), we see the computational complexity reduced by 15 times at the expense of an extra 0.04-0.06 dB. It is clear that the nearest neighbour version of SSVQ is the best in terms of the distortion versus computational complexity trade-off.

Comparison with Split Vector Quantisers

Tables 5.4 and 5.5 show the performance of the two-part and three-part split vector quantiser, described in [123], on the TIMIT database, respectively. By comparing Tables 5.20 with 5.4 and Tables 5.21 with 5.5, we can see that SSVQ outperforms SVQ for all bitrates. The 23 bits/frame three-part SSVQ is comparable to a three-part split vector quantiser operating at 25 bits/frame while requiring only about 33% of the number of computations of the latter. This shows the effectiveness of SSVQ in exploiting correlation between subspaces for lower distortion performance while requiring less computations, at the same bitrate.

In general, SSVQ requires more codebook memory than SVQ. However, the 23 bits/frame three-part SSVQ has slightly better spectral distortion than the 23 bits/frame two-part SVQ but with a third of the memory requirements (8272 cf. 28672 floats) and a fraction of the complexity (4.4 cf. 114 kflops/frame).

Comparison with Multistage Vector Quantisers

Table 5.6 lists the results of the sequentially searched, two stage multistage vector quantiser (MSVQ) on the TIMIT database where transparent coding has been achieved at 22 bits/frame. Comparing this with Table 5.21, we observe comparable spectral distortion performance between SSVQ and MSVQ. However, MSVQ is considerably more complex

(246 cf. 4.4 kflops/frame) and has higher codebook memory requirements (61440 cf. 8272 floats) than SSVQ at 23 bits/frame. Also, the sequential searched MSVQ is prone to producing unstable frames. Stability in a frame is determined by the ascending ordering of the LSFs. Split vector quantisers can be designed to ensure correct ordering between subvectors by including an extra condition when choosing codevectors, based on the relative value of LSFs on the boundaries. The ability to verify this ordering is lost in MSVQ because the second stage operates on residual vectors.

Comparison with Scalar Quantisers

Table 5.3 shows the spectral distortion performance of independent, non-uniform scalar quantisers. A spectral distortion of 1 dB is achieved at 32 bits/frame. It can be seen that the 23 bits/frame three-part SSVQ is comparable in performance to scalar quantisers operating at 32-34 bits/frame. Table 5.2 shows the performance of the non-uniform scalar quantisers that are used in the U.S. Federal Standard 1016, 4.8 kbps CELP coder [27]. It is clear that SSVQ performs considerably better than the scalar quantisers in the FS-1016 4.8 kbps CELP coder.

Comparison with GMM-based Block Quantisers

Table 5.7 presents the spectral distortion performance of the GMM-based block quantiser (single frame) on the TIMIT database. Comparing Table 5.7 with 5.21, the 23 bits/frame three-part SSVQ gives comparable spectral distortion to that of the GMM-based block quantiser operating at 25 bits/frame, while requiring less than 2% of the number of computations.

Furthermore, this quantisation scheme is particularly intensive in computation as each vector is quantised multiple times and spectral distortions are calculated for each cluster. This is highlighted in Table 5.8. As an informal comparison of computational performance, we measured the times it took to quantise LSF vectors at 24 bits/frame using the 16 cluster GMM-based block quantiser[20] and three-part SSVQ (8 switching directions). The platform used to perform the test was a 2.4 GHz Intel Pentium 4 processor running the Linux operating system. The 24 bits/frame GMM-based block quantiser took, on average,

[20]Note that the full spectral distortion was calculated. There is a simpler approximation of spectral distortion given in [52].

301 seconds to quantise 85353 LSF vectors (SD of 1.049 dB) while the 24 bits/frame three-part SSVQ took 7 seconds (SD of 0.955 dB). Therefore, in terms of spectral distortion and computational complexity, SSVQ performs better than the memoryless, fixed rate GMM-based block quantiser in coding LSFs.

5.7 Chapter Summary

In this chapter, we first reviewed the basics of speech coding, such as speech production and the modelling of speech using linear prediction analysis. The operation of various speech coders was also described and this highlighted the role and importance of LPC quantisation. Different LPC parameter representations, that are both robust to quantisation and provide simple checks for filter stability, were covered. The line spectral frequencies are one of the more popular representations and were thus used in our evaluation of various quantisation schemes.

The first quantisation scheme that we evaluated was the multi-frame GMM-based block quantiser, which has the advantage of bitrate scalability and bitrate independent complexity. By extending the decorrelating transform to exploit the linear dependencies between multiple frames, the multi-frame GMM-based block quantiser was able to achieve transparent coding at bitrates as low as 21 bits/frame, though the computational complexity and memory requirements become an issue. This quantisation scheme was compared with scalar quantisers, the split vector quantiser, the multistage vector quantiser, and the single-frame GMM-based block quantiser, and was generally found to perform better in terms of spectral distortion, bitrate, and complexity.

The switched split vector quantiser was also evaluated as an LSF quantiser. Transparent coding was achieved at bitrates as low as 22 bits/frame, though the memory requirements of the two-part SSVQ were relatively high. It was determined that the three-part SSVQ, with transparent coding at 23 bits/frame, was well-balanced in terms of quantisation performance and complexity. Compared with other single-frame quantisers, the SSVQ achieved generally better spectral distortion performance. One aspect that the SSVQ excelled was the low computational complexity.

Chapter 6

LPC Parameter Quantisation in Wideband Speech Coding

6.1 Abstract

In this chapter, we report on the contributions to LPC parameter quantisation in the area of wideband speech coding. We begin with the definition of wideband speech and describe its advantages over toll-quality narrowband speech, such as improved naturalness and the ability to distinguish between fricatives, as well as allowing better presence of the speaker, all of which can alleviate listener fatigue. Next, we review some of the state-of-the-art coding schemes for wideband speech, such as the Transform Coded Excitation (TCX) coder, and the industry standard coders such as the ITU-T G.722, which is a 64 kbps subband/ADPCM coder, and ITU-T G.722.2, which is the Adaptive Multirate wideband (AMR-WB) ACELP coder.

In the remaining half of the chapter, we provide a review of the various quantisation schemes for coding LPC parameters that have been reported in the wideband speech coding literature. In addition to this, we evaluate some of these schemes, such as PDF-optimised scalar quantisers with non-uniform bit allocation, vector quantisers, and the GMM-based block quantiser on the two competing LPC parameter representations: line spectral frequencies (LSFs) and immittance spectral pairs (ISPs). Our experimental results indicate that ISPs are superior to LSFs by 1 bit/frame in independent quantiser schemes, such as scalar quantisers; while it is the opposite for joint vector quantiser schemes. We also

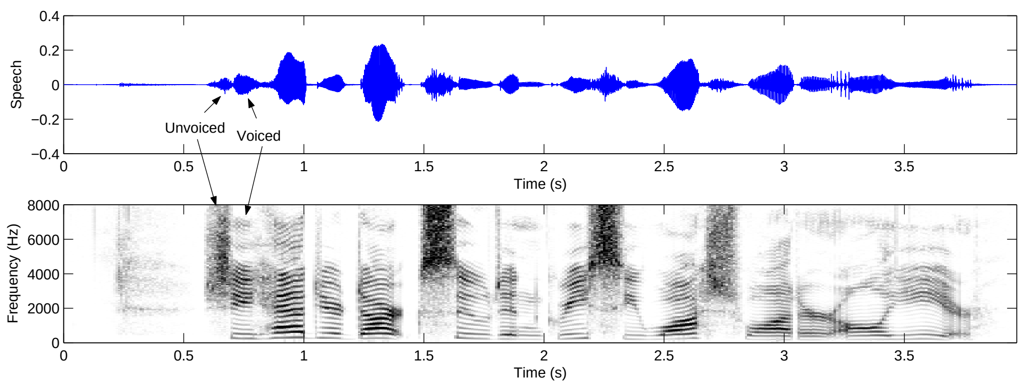


Figure 6.1: Waveform and spectrogram of wideband speech. The sentence that is spoken is *'she had your dark suit in greasy wash-water all year'*, and with the unvoiced /*s*/ and voiced /*iy*/ sounds in *she*, highlighted.

derive informal lower bounds, 35 bits/frame and 36 bits/frame, for the transparent coding of LSFs and ISPs, respectively, via the extrapolation of the operating distortion-rate curve of the unconstrained vector quantiser. Finally, we present and discuss the results of applying the switched split vector quantiser (SSVQ) with weighted distance measure and the multi-frame GMM-based block quantiser, which achieve transparent coding at 42 bits/frame and 37 bits/frame, respectively, for LSFs. ISPs were found to be inferior to the LSFs by 1 bit/frame.

Publications resulting from this research: [168, 170, 171, 172, 173]

6.2 Introduction

6.2.1 The Improved Quality of Wideband Speech

While narrowband speech has acceptable quality (otherwise known as toll quality), that is similar to telephone speech, it was found to be inadequate for applications that demanded higher quality reconstruction, such as videophones, teleconferencing, multimedia, etc. Problems with narrowband speech include lack of naturalness and speaker 'presence', as experienced in face-to-face speech communication, as well as difficulty in distinguishing between fricative sounds, such as /*s*/ and /*f*/ [5]. All of these can lead to listener fatigue.

By lowering the low frequency cut-off from 300 Hz to 50 Hz, the naturalness and

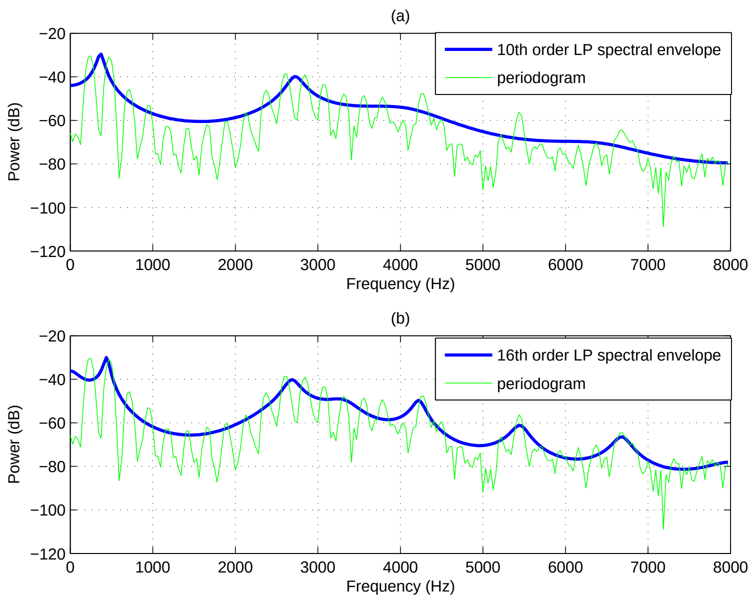


Figure 6.2: Spectral envelope estimate of 20 ms of wideband speech (starting from 3.53 s) for different orders of linear prediction analysis: (a) using 10th order autocorrelation method; (b) using 16th order autocorrelation method.

fullness of the speech can be improved, while extending the high frequency cut-off from 3400 Hz to 7000 Hz, improves the distinguishing of fricative sounds [134]. This extended range of 50–7000 Hz of wideband speech roughly corresponds to the bandwidth of speech sampled at 16 kHz. Figure 6.1 shows the waveform and spectrogram of 16 kHz speech. We can see that most of the spectral information for voiced speech, in the form of formants, occurs below 3.4 kHz. However, for unvoiced speech, there is some spectral information extending beyond 3.4 kHz that may be important for discriminating fricative sounds. For example, the spectrum starting at 1.5 and 2.25 seconds is stronger at frequencies above 4 kHz (darker areas) than the spectra for other unvoiced sections.

6.2.2 LPC Analysis of Wideband Speech

Figure 6.2 shows the spectral envelope estimates and power spectral densities of wideband speech using different orders of linear prediction analysis. Using the speech utterance of Figure 6.1, a 20 ms frame was extracted starting from 3.53 s. A Hamming window was applied to prevent the masking of high frequency formants by spectral leakage and the autocorrelation method was used in the linear prediction analysis.

We can see that the 10th order linear prediction analysis has captured the two formants below 4 kHz (Figure 6.2(a)), which corresponds to the frequency range of narrowband speech. However, three formants can be observed above 4 kHz which are not captured by the 10th order linear prediction analysis. Therefore, higher orders are required for accurate estimation of the short-term correlation information in wideband speech. Figure 6.2(b) shows the spectral envelope estimate from a 16th order linear prediction analysis, where we can see the capturing of the higher order formants.

6.2.3 Coding of Wideband Speech

The wideband speech coders that have appeared in the literature can be broadly classified as either transform coders or CELP-based coders. Examples of transform coders include the 64 kbps ITU-T G.722 subband coder and those of Wuppermann and de Bont [202], Quackenbush [137], and Crossman [33]. These transform coders are not designed to exploit the properties of speech and hence can accommodate other types of audio, such as music. However, better coding performance can be achieved with the CELP-based coders, which are specifically designed for speech coding, though they perform poorly when other types of audio are present [5].

CELP-based coders for wideband speech were first investigated by Shoham [162], Laflamme *et al.* [92], and Roy and Kabal [148]. It was recognised that due to the increase in bandwidth and sampling frequency, maintaining acceptable quality at low bitrates often required a significant increase in the complexity of CELP, which was already computationally demanding, and this inhibited the use of full-band CELP in wideband speech coding. For instance, very large excitation codebooks are needed due to the need to use larger frame sizes [92]. Full-band CELP also suffers from an intermittent background hiss. This is because full-band CELP generally achieves accurate coding of speech in the low frequency spectrum but does poorly in coding the high frequency spectral region, due to the spectral tilt that can be large as 35 dB [190].

Various techniques were applied to reduce the computational burden of CELP and/or improve its quality, which included the use of algebraic CELP (ACELP) at 16 kbps [92], which uses a more compact and search efficient algebraic codebook, and split-band CELP at 16 kbps [148], which splits the excitation signal into two frequency bands while placing more emphasis on coding the low frequency band. However, enough emphasis is placed on

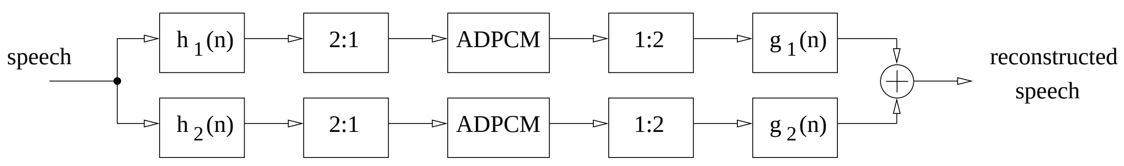


Figure 6.3: Block diagram of the 64 kbps ITU-T G.722 wideband speech coder (after [5]). The blocks labelled as '2:1' and '1:2' are decimators and interpolators, respectively

the high frequency band to overcome the limitations of full-band CELP in handling the spectral tilt. Consequently, split-band CELP has better quality and lower complexity than full-band CELP. Other methods that deal with the spectral tilt include pre-processing the speech with a pre-emphasis filter and modified error weighting [190].

Harborg *et al.* [64] reported a full-band CELP coder at 16 kbps which had a low complexity and was able to run in real-time on a TMS320C31 DSP. Ubale and Gersho [190] introduced the multi-band CELP approach that uses off-line filtered multi-band excitation codebooks. Multi-band CELP was designed to overcome the quality issues in split-band CELP that were due to the band overlap. Finally a hybrid scheme called transform coded excitation (TCX), which combined transform coding with the CELP approach, was investigated by Lefebvre *et al.* [95] and this was reported to achieve high quality speech reconstruction at 16 kbps. The TCX coder will be described in Section 6.3.2.

6.3 Wideband Speech Coders

6.3.1 The ITU-T G.722 Wideband Speech Coder

In 1986, the ITU-T standardised G.722, which is a 64 kbps wideband speech coder [5]. As shown in Figure 6.3, it is a two-band subband coder using ADPCM. The 24-tap quadrature mirror filters (QMFs) split the speech frequency spectrum into two overlapping bands. Because most of the energy appears in the lower half of the spectrum (0–4 kHz), the G.722 coder allocates 6 bits/sample to the lower band, and 2 bits/sample to the upper band [5].

The G.722 speech coding standard allows other data channels to be accommodated by lowering the overall rate to free up some bits. This is achieved by varying the number of

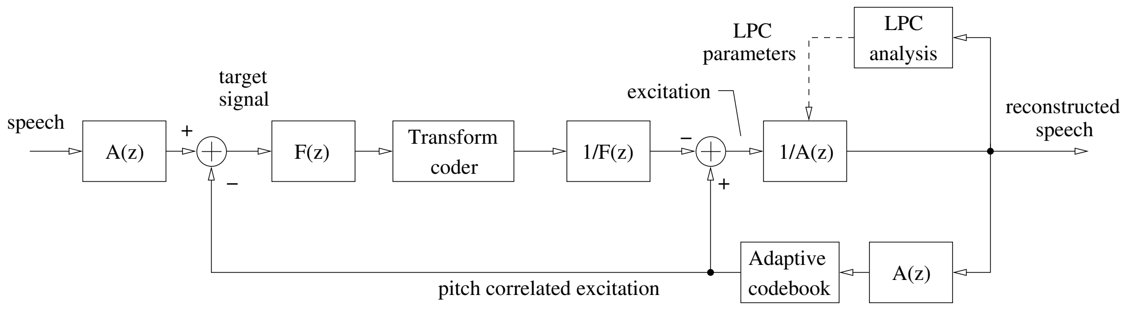


Figure 6.4: Block diagram of the 16 kbps TCX wideband speech coder (after [94]).

bits given to the lower band, which can take the values of 6, 5, and 4 bits/sample, and these correspond to a total bitrate of 64, 56, and 48 kbps, respectively [5].

The G.722 speech coder is classified as a waveform-approximating coder, where the aim is to minimise the quantisation error between the reconstructed and original speech. The distortion measure is non-weighted mean squared error and because of this, auditory masking properties are not exploited. Waveform-approximating coders achieve excellent reconstruction at medium to high bitrates, but degrade considerably at low bitrates [124]. Therefore, parametric coders that are based on the CELP principle have been investigated for wideband speech coding at lower bitrates and these are discussed in the following sections. The G.722 coder, though, serves as a useful quality reference for these low bitrate coders [5].

6.3.2 The Transform Coded Excitation (TCX) Wideband Speech Coder

Representing the state-of-the-art in wideband speech coding is the transform coded excitation (TCX) wideband speech coder, which was introduced by Lefebrve *et al.* [94] as a low complexity alternative to the traditional CELP speech coders at bitrates, where there are a large number of bits to encode the excitation signal [5]. Originally used for narrowband speech coding, it was later applied to wideband speech coding in [95], where at 16 kbps, there were enough bits to encode the excitation signal.

The CELP coder achieves acceptable quality at lower bitrates because of the vector quantisation of the excitation signal, coupled with an analysis-by-synthesis approach. In CELP, each code-vector is scaled by a gain and filtered using the long-term and short-term predictors before the error between the synthesised and original speech is calculated. Based on minimising the weighted version of this error, the appropriate excitation vector is then selected. This code-vector search of the fixed innovation codebook using the analysis-by-synthesis approach is the most computationally intensive procedure in the CELP coder.

The TCX coder, which is shown in Figure 6.4, operates at a lower complexity by removing the analysis-by-synthesis procedure and directly quantising the excitation signal (otherwise, known as the target signal) using a transform coder in a weighted domain [5]. Speech is initially pre-emphasised using the filter, $H(z) = 1 - \mu z^{-1}$ where $\mu = 0.5$, to remove the spectral tilt of the speech spectrum and emphasise high frequency formants [95]. Backward adaptive LPC analysis is performed on the past reconstructed speech and these parameters are used in the synthesis filter, $1/A(z)$. This filter is driven by an excitation signal which is comprised of a pitch-correlated and innovation component. The adaptive codebook, which is used to generate a pitch-correlated excitation, is similar to the one used in CELP coders, but instead uses past synthesised speech that has been whitened by $A(z)$.

In order to code the speech, short-term and long-term correlations are removed initially by whitening with the filter, $A(z)$, and then subtracting the pitch-correlated excitation, respectively. The resulting residual signal, which is termed the *target* signal, is then weighted with the filter, $F(z) = 1/A(z/\gamma)$ where $\gamma = 0.9$, and then coded using a Fourier transform coder. In speech decoding, the coded innovation excitation is decoded and then long-term and short-term correlation information is introduced by adding the pitch excitation and filtering with the synthesis filter, $1/A(z)$, respectively.

It is of interest to note the similarities between the TCX coder and the error-weighted DPCM coder of [15]. Both schemes remove long-term and short-term correlations from the speech before directly quantising the residual in a weighted error domain. The fundamental difference between these two schemes is the use of a transform coder in TCX, which is generally more efficient in a rate-distortion sense, than the scalar quantiser in DPCM.

6.3.3 The Adaptive Multirate Wideband (AMR-WB) Speech Coder

The AMR-WB wideband speech coder was adopted by the 3GPP in 2000 and standardised by the ITU-T as recommendation G.722.2. It is based on the ACELP coder and can operate at different bitrates (23.85, 23.05, 19.85, 18.25, 15.85, 14.25, 12.65, 8.85, and 6.6

kbps) [19]. In this section, we describe briefly the main points of operation, specifically the pre-processing, LPC analysis, and LPC parameter quantisation. For more details on the AMR-WB speech coder, the reader should consult the following references: [3, 19].

Speech Pre-Processing

The 16 kHz speech is processed in two frequency bands, 50–6400 Hz and 6400–7000 Hz, to decrease complexity and prioritise bit allocation to the subjectively important lower band [19]. In the lower band, speech is down-sampled to 12.8 kHz and pre-processed using a high pass filter, $H_h(z)$, and pre-emphasis filter, $H_p(z)$, which are given by [3]:

$$H_h(z) = \frac{0.989502 - 1.979004z^{-1} + 0.989502z^{-2}}{1 - 1.978882z^{-1} + 0.979126z^{-2}} \tag{6.1}$$

$$H_p(z) = 1 - 0.68z^{-1} \tag{6.2}$$

The role of the high pass filter is to remove undesired low frequency components while the pre-emphasis filter removes the spectral tilt in the speech spectrum and emphasises higher frequency formants for more accurate LPC analysis.

LPC Analysis

Speech frames of 20 ms are extracted using a 30 ms asymmetric Hamming window which emphasises the fourth subframe (subframes are 5 ms long each). Autocorrelations are calculated with a 5 ms lookahead. The autocorrelation method is used to perform a 16th order LPC analysis with a 60 Hz bandwidth expansion (using the lag window method) and a white noise correction factor of 1.0001 [3]. The 16 LPC coefficients are transformed to immittance spectral pairs (ISPs) [23], $\{q_i\}_{i=1}^{16}$, and then further converted to immittance spectral frequencies[1] (ISFs), $\{f_i\}_{i=1}^{16}$, for quantisation [3]:

$$f_i = \begin{cases} \frac{F_s}{2\pi}\cos^{-1}(q_i) & \text{, for } i = 1, 2, \ldots, 15 \\ \frac{F_s}{4\pi}\cos^{-1}(q_i) & \text{, for } i = 16 \end{cases} \tag{6.3}$$

where $F_s = 12.8$ kHz is the sampling frequency.

[1]It should be noted that the last ISF corresponds to half the arc-cosine of the 16th reflection coefficient, thus it is a different quantity to the first 15 ISFs.

Quantisation of Residual ISF Vectors

In order to exploit interframe correlation, a first order moving average (MA) predictor is applied:

$$\boldsymbol{r}(n) = \boldsymbol{f}(n) - \boldsymbol{p}(n) \tag{6.4}$$

where $\boldsymbol{r}(n)$ is the prediction residual vector, $\boldsymbol{f}(n)$ is the current frame of ISFs, and $\boldsymbol{p}(n)$ is the predicted ISF that is given by:

$$\boldsymbol{p}(n) = \frac{1}{3}\hat{\boldsymbol{r}}(n-1) \tag{6.5}$$

where $\hat{\boldsymbol{r}}(n-1)$ is the quantised residual vector of the previous frame [3].

The residual ISF vectors are quantised using split-multistage vector quantisation (S-MSVQ) at 36 bits for the lowest bitrate mode (6.6 kbps); and 46 bits for the other higher bitrate modes. S-MSVQ first appeared in the ATCELP wideband speech coder described by Combescure *et al.* [30] and is essentially a multistage vector quantiser with split vector quantisers in each stage. It combines the low complexity characteristics of both split vector quantisers (SVQ) and multistage vector quantisers (MSVQ), making it practical to quantise 16-dimensional vectors using 46 bits. The S-MSVQ and its performance will be discussed in a later section.

6.4 Quantisation of Wideband LPC Parameters

6.4.1 Introduction

In this section, we provide a brief review of the LPC quantisation schemes that have been reported in the wideband speech coding literature. Also included are results from some popular quantisation schemes, such as PDF-optimised scalar quantisers with non-uniform bit allocation, unconstrained vector quantisers, the split-multistage vector quantiser from the AMR-WB coder, and the GMM-based block quantiser [183]. These will serve as a bases for comparison, when we apply the two schemes that were introduced in this dissertation, namely the switched split vector quantiser and multi-frame GMM-based block quantiser. We will also attempt to extrapolate the operating distortion-rate curve of the unconstrained vector quantiser to find an informal lower bound on the bitrate required for transparent coding of wideband LPC parameters.

Because of the adoption of immittance spectral pairs (ISPs) as the LPC parameter representation in the AMR-WB speech coder, we also have included quantisation results for ISPs as well as LSFs and compare their relative performance. More specifically, we quantise the ISPs in the frequency domain, which are referred to from here on as *immittance spectral frequencies* (ISFs). Therefore, we will use the terms ISP and ISF interchangeably in the rest of the chapter. We follow the definition of ISFs given by (6.3) [3].

6.4.2 Literature Review

Harborg *et al.* [64] quantised 16 to 18 log-area-ratio coefficients at 60 to 80 bits/frame using non-uniform, PDF-optimised scalar quantisers, in their real-time wideband CELP coder. Lefebvre *et al.* [94] used a split vector quantiser with seven part splitting $(2, 2, 2, 2, 2, 3, 3)$ to quantise 16 LSFs at 48 bits/frame in their TCX wideband coder. Chen *et al.* [29] also used a seven-part split vector quantiser, with the same subvector sizes as in [94] operating at 49 bits/frame to quantise 16 LSF parameters in their transform predictive coder. Transparent results were reported by Biundo *et al.* [24] for a four and five part split vector quantiser at 45 bits/frame.

Because successive LSF frames are highly correlated [61], better quantisation can be achieved by exploiting the interframe correlation. Roy and Kabal [148] quantised 16 LSFs at 50 to 60 bits/frame using non-uniform differential scalar quantisation. They noted that because the LSFs were related to the formant positions, then the lower frequency LSFs were perceptually more important, hence more bits should be allocated to them. Combescure *et al.* [30] quantised 12 LSFs from a decimated lower band using a predictive split multistage vector quantiser at 33 bits/frame. Ubale *et al.* [190] used a seven-stage tree-searched multistage vector quantiser [93] with a moving average (MA) predictor at 28 bits/frame, while Biundo *et al.* [24] reported transparent results using an MA predictive split multistage vector quantiser (S-MSVQ) at 42 bits/frame. Guibé *et al.* [61] achieved transparent coding using a safety-net vector quantiser at 38 bits/frame, while the Adaptive Multirate wideband (AMR-WB) speech codec [19, 3] uses an S-MSVQ with MA predictor at 46 bits/frame to quantise ISPs.

Other quantisation schemes recently reported include the predictive Trellis-coded quantiser [160] and the HMM-based recursive quantiser [42] which achieve a spectral distortion of 1 dB at 34 and 40 bits/frame, respectively.

6.4.3 Experimental Setup

The TIMIT database was used in the training and testing of the quantisation experiments, where speech is sampled at 16 kHz. We have used the lower band pre-processing and LPC analysis of the 3GPP implementation of the AMR-WB speech codec (floating point version) [3, 4] to produce linear prediction coefficients which are then converted to line spectral frequency (LSF) representation [75] and immittance spectral pairs (ISP) representation [23] (quantisation is performed on the frequency form of ISPs, ie. ISFs).

The training set consists of 333789 vectors while the evaluation set, consisting of speech not contained in the training, has 85353 vectors. Unless specified otherwise, unweighted mean squared error is used as the distance measure for quantiser design and spectral distortion is used for evaluating quantisation performance. In narrowband speech coding, the conditions for transparent speech from LPC parameter quantisation are [123]:

1. The average spectral distortion (SD) is approximately 1 dB;
2. there is no outlier frame having more than 4 dB of spectral distortion; and
3. less than 2% of outlier frames are within the range of 2–4 dB.

According to Guibé *et al.* [61], listening tests have shown that these conditions for transparency also apply to the wideband case. Therefore, we have adopted these conditions in our analysis.

6.4.4 PDF-Optimised Scalar Quantisers with Non-Uniform Bit Allocation

Figure 6.5 shows the histogram of ISFs and LSFs from the test set of the TIMIT database. We can see that the distribution of the low to medium frequency parameters tend to be multimodal while high frequency ones are unimodal. Therefore, the best strategy is to quantise each LPC parameter using a scalar quantiser that is optimised for each specific probability density function (PDF). In this experiment, we have designed non-uniform

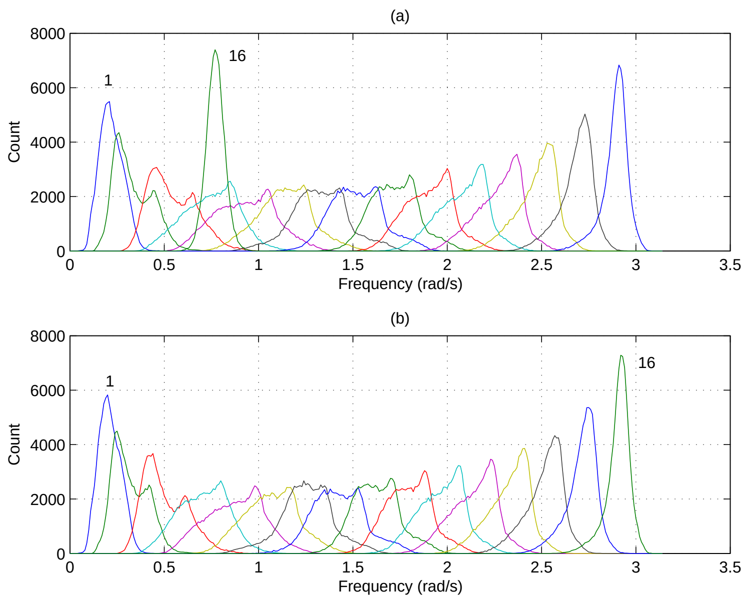


Figure 6.5: Histograms of LPC parameters from the TIMIT database (the first and last components are labelled): (a) immittance spectral frequencies (ISFs); (b) line spectral frequencies (LSFs)

scalar quantisers using the generalised Lloyd algorithm for each LPC parameter. Bit allocation was performed using a greedy algorithm that is similar to that described by Soong and Juang [175], where each bit is given to the scalar quantiser which results in the largest distortion improvement. This simple algorithm results in a locally optimal allocation of bits. To reduce the computational complexity of the bit allocation procedure, we have used mean squared error as the distortion measure rather than spectral distortion.

Table 6.1 shows the spectral distortion performance of non-uniform, PDF-optimised scalar quantisers on immittance spectral frequency vectors from the TIMIT database. We can see that a spectral distortion of 1 dB can be achieved at a bitrate of 58 bits/frame. In Table 6.2, which shows the spectral distortion performance of non-uniform scalar quantisation of line spectral frequency vectors, we can see that 1 dB spectral distortion is achieved at 59 bits/frame. We note that these results reflect the narrowband speech coding results given by Bistritz and Peller [23], where ISPs were observed to also perform better than LSFs by 1 bit/frame, in differential scalar quantisation.

Table 6.1: Average spectral distortion of the PDF-optimised scalar quantisers as a function of bitrate on wideband ISF vectors from the TIMIT database

Bits/frame	Avg. SD (dB)	Outliers (in %)	
		2–4 dB	> 4 dB
61	0.911	0.85	0.00
60	0.947	0.97	0.00
59	0.961	1.00	0.01
58	1.016	1.22	0.01
57	1.060	1.51	0.01
56	1.137	2.19	0.01
55	1.181	2.63	0.01

Table 6.2: Average spectral distortion of the PDF-optimised scalar quantisers as a function of bitrate on wideband LSF vectors from the TIMIT database

Bits/frame	Avg. SD (dB)	Outliers (in %)	
		2–4 dB	> 4 dB
61	0.918	0.82	0.00
60	0.970	0.95	0.00
59	1.011	1.18	0.01
58	1.080	1.64	0.01
57	1.120	1.88	0.01
56	1.162	2.26	0.01
55	1.219	2.98	0.03

6.4.5 Unconstrained Vector Quantisers and an Informal Lower Bound for Transparent Coding

In theory, unconstrained vector quantisers can achieve the lowest distortion of any quantisation scheme at a given bitrate and dimension. However, the exponentially growing complexity and storage, with respect to increasing bitrate and dimensionality, inhibits their use in practical schemes that require high bitrates. However, they can be used to provide an informal lower bound on the spectral distortion via extrapolation to higher bitrates, similar to that reported in [125] for narrowband LSF quantisation.

We have applied unconstrained vector quantisers to the task of quantising ISF and LSF vectors from the TIMIT database and their spectral distortion performance is shown in Tables 6.3 and 6.4, respectively. Unweighted mean squared error is used in the design and encoding phases of the vector quantiser. Due to computational constraints, we have only been able to train VQ codebooks up to 16 bits. For higher bitrates (18 bits/frame), we have used a VQ codebook that consists of randomly selected vectors from the training set. The resulting codebook will perform suboptimally, hence the spectral distortion will serve as an upper bound.

We notice from Tables 6.3 and 6.4 that for vector quantisation, LSFs produce slightly lower spectral distortion than ISFs, which is contrary to the results for scalar quantisation in the previous section. This is probably due to the inclusion of the 16th immittance spectral frequency (which is related to the reflection coefficient), in the joint quantisation of the vector. Because this parameter is not really a ‘frequency’, unlike the first 15 ISFs, and possesses different quantisation properties and sensitivity characteristics, then using the unweighted mean squared error (which assumes all vector components are similar) as a distance measure may not be optimal. The VQ code-points for the 16th ISF become dependent on (and their locations are more or less constrained by) the other 15 ISFs. Therefore, independent quantisation of the 16th ISF, as is the case with the PDF-optimised scalar quantisers, will generally result in lower spectral distortion.

In order to highlight the different quantisation characteristics of the 16th ISF, Figures 6.6(a) and (b) show the original and reconstructed spectral envelope estimates for the LSF and ISF representation, where the last (16th) parameter has been shifted by 142 Hz. We note the spectral localisation of the distortion caused by a shift of the 16th LSF. On the

Table 6.3: Average spectral distortion of the unconstrained vector quantiser as a function of bitrate on wideband ISF vectors from the TIMIT database

Bits/frame	Avg. SD (dB)	Outliers (in %)	
		2–4 dB	> 4 dB
16	2.445	70.36	1.86
15	2.521	72.09	2.69
14	2.609	73.16	4.01
13	2.705	74.07	5.53
12	2.814	73.96	7.96
11	2.934	72.60	11.30
10	3.067	70.50	15.56
9	3.218	66.79	21.20
8	3.385	61.96	27.80
7	3.577	55.96	35.50

Table 6.4: Average spectral distortion of the unconstrained vector quantiser as a function of bitrate on wideband LSF vectors from the TIMIT database

Bits/frame	Avg. SD (dB)	Outliers (in %)	
		2–4 dB	> 4 dB
16	2.420	70.22	1.54
15	2.498	72.09	2.29
14	2.585	73.58	3.28
13	2.686	74.46	4.92
12	2.794	74.31	7.31
11	2.916	73.28	10.63
10	3.051	71.06	14.96
9	3.203	67.20	20.67
8	3.370	62.23	27.41
7	3.563	56.44	35.02

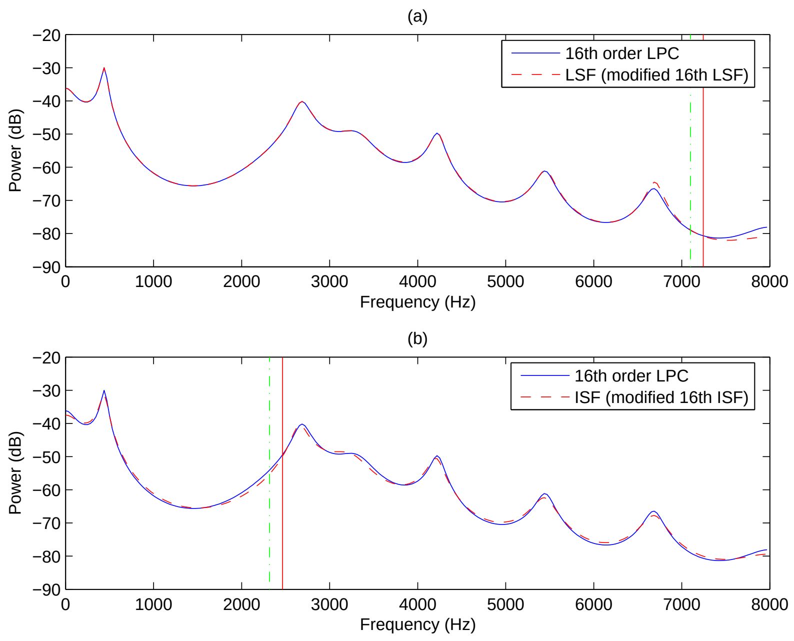


Figure 6.6: Original and reconstructed power spectral envelope estimates for 16th order LPC analysis: (a) Shifting the 16th LSF by 142 Hz (SD=0.583 dB); (b) Shifting the 16th ISF by 142 Hz (SD=0.6838 dB). The solid and dashed vertical lines show the original and shifted parameters (LSF and ISF), respectively.

other hand, because the 16th ISF is not really a 'frequency', but is essentially related to the 16th reflection coefficient, a shift of 142 Hz results in distortion appearing throughout the entire spectrum. The average spectral distortion of the modified LSF power spectral density is also less than that of the modified ISF power spectral density (0.583 cf. 0.684 dB).

Figures 6.7(a) and (b) shows the operating distortion-rate (D-R) curves of the unconstrained vector quantisation of LSFs and ISFs, respectively. The squares represent the performance of the vector quantiser that is properly trained using the LBG algorithm while the triangles represent the performance of the vector quantiser with a codebook formed from randomly picked training vectors. We can see that at low bitrates, the D-R curve is exponential-like, while at high bitrates, the curve is more linear. If we make the loose assumption of a linear D-R curve from 13 bits/frame and onwards, then a least-squares linear regression (the line in Figure 6.7) shows that single frame, vector quantisers need at least 35 bits/frame and 36 bits/frame to achieve a spectral distortion of 1 dB, for LSFs and ISFs, respectively. We should note that this is by no means a tight lower bound

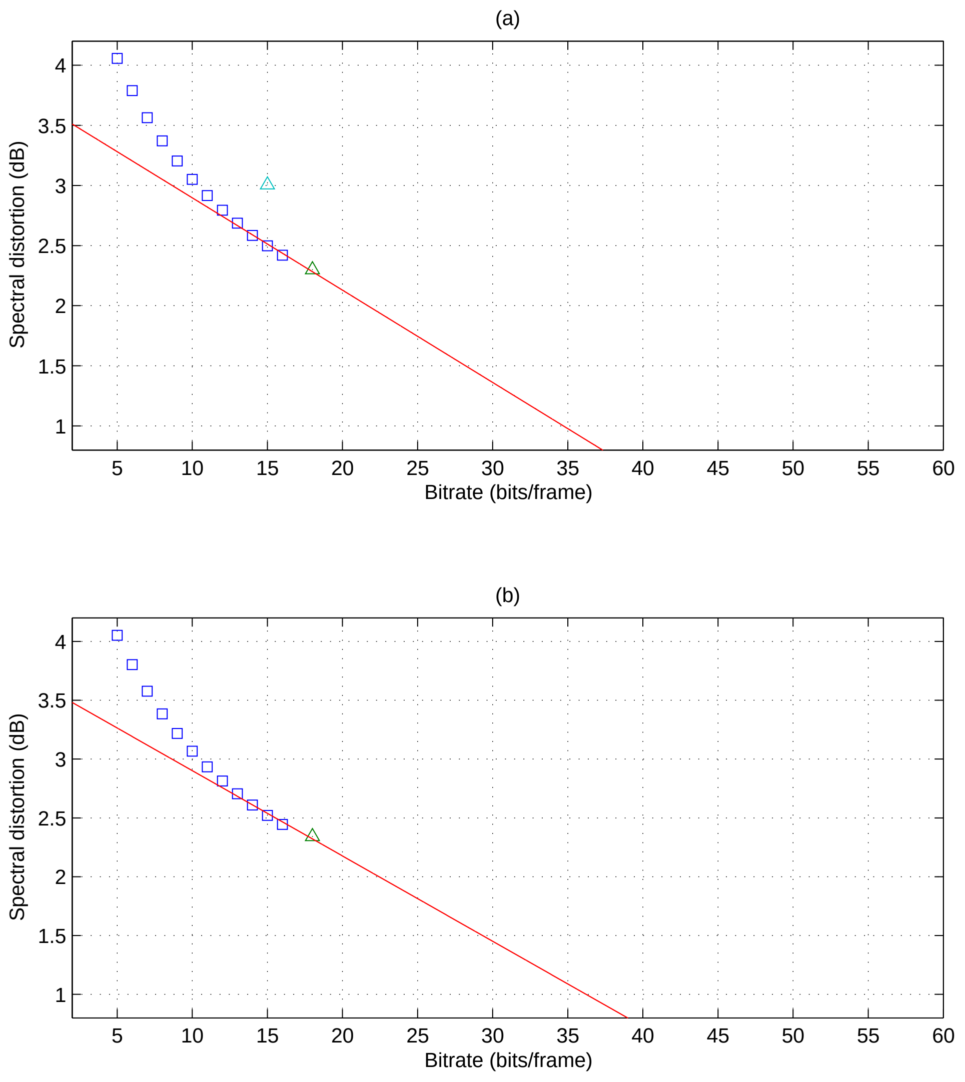


Figure 6.7: Extrapolating from the operating distortion-rate curve of the unconstrained vector quantisers to approximate a lower bound for transparent coding, when using: (a) line spectral frequencies (wideband); and (b) immittance spectral frequencies (wideband). □'s indicate the performance of a VQ codebook trained using LBG, while △'s indicate the performance of a VQ codebook consisting of random vectors from the training set.

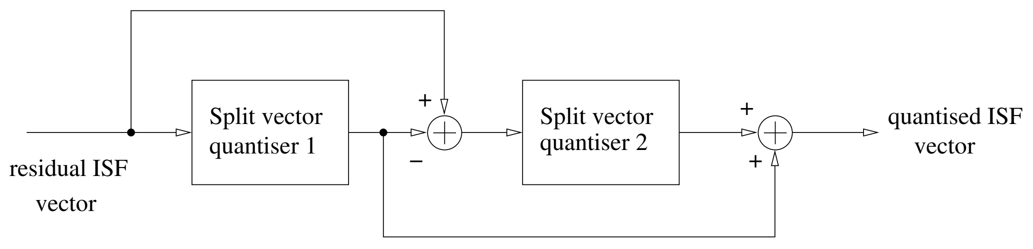


Figure 6.8: Block diagram of the split-multistage vector quantiser (after [24])

for several reasons. Firstly, the high rate linearity assumption of the D-R curve is rather loose as there are not enough operating points to determine such a trend. Also, due to the finite and limited number of training vectors, the vector quantiser becomes more and more 'over-trained'. Furthermore, the LBG algorithm generally produces codebooks that are locally optimal, depending on the initialisation used. Finally, the distortion performance of a VQ codebook consisting of randomly selected training vectors, should generally be worse than an LBG trained one. This is demonstrated in Figure 6.7(a) at 15 bits/frame, where there is a 0.5 dB spectral distortion difference between the randomly selected codebook and LBG trained codebook. Therefore, 35 bits/frame and 36 bits/frame can be said to be informal lower bounds only, for the transparent coding of LSFs and ISFs, respectively. A better approach would be that of Hedelin and Skoglund [66], where a Gaussian mixture model (GMM) with bounded support is used to estimate the LSF vector source. This GMM can then be used to generate a set of artificial training vectors which are used to train VQ codebooks using the LBG algorithm. This avoids the 'over-training' issue due to a limited size training set.

6.4.6 Split-Multistage Vector Quantisers with MA Predictor

The split-multistage vector quantiser (S-MSVQ), which is shown in Figure 6.8, is used in the AMR-WB speech coder for coding immittance spectral frequencies. It can be seen that it is essentially a modified multistage vector quantiser with split vector quantiser stages. This combination of product code vector quantisers allows for a large reduction in computational complexity. During residual ISF encoding, the S-MSVQ is searched using the efficient M-L tree search algorithm [93], where M surviving paths are determined and the path which minimises the distortion is selected.

For the 46 bits/frame S-MSVQ in the AMR-WB speech coder, residual ISF vectors

Table 6.5: Average spectral distortion as a function of bitrate of the AMR-WB (ITU-T G.722.2) split-multistage vector quantiser with MA prediction on wideband ISF vectors from the TIMIT database

Bits/frame	Avg. SD (dB)	Outliers (in %)	
		2–4 dB	> 4 dB
46	0.894	0.76	0.01
36	1.304	5.94	0.03

are split into two subvectors of dimension 9 and 7, respectively. Each subvector is then quantised using a two-stage multistage vector quantiser, where in the first stage, each subvector is quantised using 8 bits, and in the second stage, the two residual subvectors are further split into 3 and 2 subvectors, respectively. The bit allocation for each subvector in the second stage are $(6, 7, 7)$ bits and $(5, 5)$ bits [3].

Table 6.5 shows the spectral distortion performance of the S-MSVQ with MA predictor on ISF vectors from the TIMIT database. The S-MSVQ was adapted from the 3GPP implementation of the AMR-WB speech coder [4] and operates at 36 and 46 bits/frame. We can see that the S-MSVQ at 46 bits/frame, which is used in all bitrate modes except the lowest, achieves a spectral distortion that is lower than 1 dB, though there is a small percentage of outlier frames having a spectral distortion of greater than 4 dB. At 36 bits/frame, which is used in the lowest bitrate mode, we observe a higher spectral distortion and percentage of outlier frames, hence the coding is speech is less likely to be transparent.

6.4.7 GMM-Based Block Quantisers

Subramaniam and Rao [183] applied the GMM-based block quantiser to quantise narrowband speech LSFs. The distortion measure used for the minimum distortion block quantisation was spectral distortion. We can see in Figure 6.5 that higher frequency ISFs and LSFs tend to be unimodal, while the lower frequency ones have a multimodal distribution. Therefore, the GMM-based block quantiser is expected to perform well because of the ability to estimate and quantise the multimodal PDFs of the lower frequency LSFs, which are perceptually more important. Also, because of the use of decorrelating transforms, the GMM-based block quantiser is expected to achieve lower spectral distortions than the independent PDF-optimised scalar quantisers at a given bitrate.

Table 6.6: Average spectral distortion performance of the 16 cluster memoryless, fixed rate GMM-based block quantiser using spectral distortion criterion at different bitrates on wideband ISF vectors from the TIMIT database

Bits/frame	Avg. SD	Outliers (in %)	
	(in dB)	2–4 dB	> 4 dB
46	0.856	0.30	0.00
42	1.001	0.91	0.00
41	1.040	1.14	0.00
40	1.080	1.46	0.00
36	1.254	4.08	0.01

Table 6.7: Average spectral distortion performance of the 16 cluster memoryless, fixed rate GMM-based block quantiser using spectral distortion criterion at different bitrates on wideband LSF vectors from the TIMIT database

Bits/frame	Avg. SD	Outliers (in %)	
	(in dB)	2–4 dB	> 4 dB
46	0.844	0.22	0.00
42	0.985	0.66	0.01
41	1.025	0.88	0.01
40	1.064	1.18	0.01
36	1.240	3.51	0.01

Tables 6.6 and 6.7 show the spectral distortion performance of the fixed-rate GMM-based block quantiser on ISF and LSF vectors, respectively, from the TIMIT database. The GMM consists of 16 clusters and during the training process, we have used 20 iterations of the EM algorithm. We can observe that a spectral distortion of 1 dB is achieved at 41 bits/frame when using the LSF representation and 42 bits/frame when using the ISF representation. This difference in performance is similar to that seen in the vector quantiser from the previous section, and is contrary to the scalar quantiser result. Like the vector quantiser, the GMM-based block quantiser exploits correlation within each vector and tends to lose performance if some of the vector components are rather lightly correlated. Because the 16th ISF is a different type of parameter from the first 15 ISFs, this fact may explain the performance loss of using the ISF representation in joint vector and block quantisation schemes, compared with independent scalar quantisation schemes, where ISFs were experimentally determined to be more superior. More specifically, the 16th ISF is not really a 'frequency', hence it does not possess the spectral localisation properties of the first 15 ISFs.

Comparing Table 6.7 with 6.2, we can see that the 41 bits/frame GMM-based block quantiser is comparable in spectral distortion performance to a 59 bits/frame PDF optimised scalar quantiser. Both schemes utilise scalar codebooks that are optimised for the PDF of each vector component. Therefore, it is apparent that the saving of up to 18 bits/frame by using the GMM-based block quantiser is mostly due to the exploitation of correlation within LSF frame by the multiple Karhunen-Loève transforms.

Comparing Table 6.6 with 6.5, we can see that the GMM-based block quantiser achieves slightly less spectral distortion and outlier frames than the S-MSVQ with MA predictor at 46 and 36 bits/frame. Because the GMM-based block quantiser is a single frame scheme and does not exploit interframe dependencies, then we can conclude that it is more efficient, in the rate distortion sense, than the S-MSVQ (without MA predictor).

We determined an informal lower bound of 35 bits/frame for the transparent coding of LSFs in Section 6.4.5. Therefore, we can say that the GMM-based block quantiser performs about 6 bits/frame worse than the unconstrained vector quantiser, when transparently coding LSFs, though it must be stressed that the lower bound is by no means a tight one.

Table 6.8 shows the bitrate independent computational complexity and memory requirements of the GMM-based block quantiser for different numbers of clusters. The

Table 6.8: Bitrate independent computational complexity (in kflops/frame) and memory requirements (ROM) of the GMM-based block quantiser using spectral distortion-based quantiser selection as a function of the number of clusters for wideband speech coding

m	kflops/frame	ROM (floats)
4	66.4	1472
8	132.9	2688
16	265.8	5120
32	531.5	9984

Table 6.9: Average spectral distortion (SD), computational complexity, and memory requirements (ROM) of the five-part switched split vector quantiser using unweighted MSE as a function of bitrate and number of switch directions of wideband LSF vectors from the TIMIT database

m	Total bits/frame	Avg. SD	Outliers (in %)		kflops/	ROM
	$(b_1 + b_2 + b_3 + b_4 + b_5 + b_m)$	(in dB)	2–4 dB	> 4 dB	frame	(floats)
8	46 (8+8+9+9+9+3)	0.919	0.54	0.00	27.13	53376
	45 (8+8+8+9+9+3)	0.953	0.64	0.00	24.06	47232
	44 (8+8+8+8+9+3)	0.984	0.79	0.00	20.99	41088
	43 (7+8+8+8+9+3)	1.018	0.90	0.00	19.45	38016
	42 (7+7+8+8+9+3)	1.066	1.37	0.00	15.35	34944
16	46 (8+8+8+9+9+3)	0.903	0.48	0.00	24.57	94464
	45 (8+8+8+8+9+3)	0.932	0.60	0.00	21.50	82176
	44 (7+8+8+8+9+3)	0.964	0.73	0.00	19.96	76032
	43 (7+7+8+8+9+3)	1.007	0.97	0.00	18.43	69888
	42 (6+7+8+8+9+3)	1.050	1.19	0.01	17.66	66816

spectral distortion calculation accounts for a large part of the complexity, requiring at least 15.3 kflops/frame (assuming a 256 point FFT). However, the memory requirements of the GMM-based block quantiser are relativity low.

6.4.8 Switched Split Vector Quantisers

Using the Unweighted Mean-Squared-Error as the Distance Measure

Table 6.9 shows the average spectral distortion, computational complexity, and memory requirements of the five-part switched split vector quantiser (SSVQ) on wideband LSF vectors from the TIMIT database. Also shown are the bit allocations used, which were determined experimentally to result in minimal spectral distortion, while maintaining moderate complexity, with b_m being the number of bits that were given to the switch vector quantiser. The 16 dimensional vectors are split into 5 subvectors of sizes $(3, 3, 3, 3, 4)$. An unweighted mean-squared-error was used as the distance measure for codebook training and searching. We can see that the 8 switch, five-part SSVQ can achieve transparent coding at 43 bits/frame with a moderate computational complexity (19.45 kflops/frame). By using more switches (16 switches), the SSVQ achieves slightly lower spectral distortions and complexity, which is offset by a large increase in memory requirements.

Comparing the performance of the SSVQ with that of the S-MSVQ with MA predictor in Table 6.5, we can see that at the SSVQ achieves slightly higher spectral distortion (0.919 cf. 0.894 dB) at 46 bits/frame, though the number of outlier frames having a spectral distortion of 2 and 4 dB is less (0.54 cf. 0.76%) than that of the S-MSVQ. These results are to be expected since the S-MSVQ scheme is an interframe scheme which uses an MA predictor, while the SSVQ operates on single frames only and does not exploit interframe correlation. Prediction-based schemes tend to achieve lower distortion at a given bitrate with a higher percentage of outlier frames, due to the inability of predictors to capture rapid changes [61].

Comparing Tables 6.9 and 6.7, we can see that the GMM-based block quantiser achieves lower spectral distortion than the SSVQ, which is in contrast to narrowband LSF quantisation, where the SSVQ was found to be better. This is because of the larger dimension and bitrates in wideband LSF quantisation. That is, in order to keep the complexity manageable, the SSVQ splits vectors into five subvectors. Splitting vectors into more and more parts, while dramatically reducing the computational complexity and memory requirements, also reduces all three vector quantiser advantages (space filling, shape, and memory) [103], which penalises the rate-distortion performance of the vector quantisation scheme. However, comparing the computational complexity of the SSVQ with that of the GMM-based block quantiser (in Table 6.8), we can see that the former requires only 7.3% of the complexity of the latter, at 43 bits/frame. Therefore, the SSVQ is more computationally efficient than the GMM-based block quantiser.

Table 6.10 shows the average spectral distortion of the five-part switched split vector quantiser on wideband ISF vectors from the TIMIT database. The same vector splitting $(3, 3, 3, 3, 4)$ was applied and an unweighted mean-squared-error was used as the distance measure. We can see that the 8 switch, five-part SSVQ achieves transparent coding at 44 bits/frame for wideband ISFs. Comparing the spectral distortions with those in Table

Table 6.10: Average spectral distortion (SD), computational complexity, and memory requirements (ROM) of the five-part switched split vector quantiser using unweighted MSE as a function of bitrate and number of switch directions of wideband ISF vectors from the TIMIT database

m	Total bits/frame	Avg. SD	Outliers (in %)		kflops/	ROM
	$(b_1 + b_2 + b_3 + b_4 + b_5 + b_m)$	(in dB)	2–4 dB	> 4 dB	frame	(floats)
8	46 (8+8+9+9+9+3)	0.931	0.53	0.00	27.13	53376
	45 (8+8+8+9+9+3)	0.968	0.86	0.00	24.06	47232
	44 (8+8+8+8+9+3)	0.999	1.07	0.00	20.99	41088
	43 (7+8+8+8+9+3)	1.037	1.21	0.00	19.45	38016
	42 (7+7+8+8+9+3)	1.080	1.68	0.00	15.35	34944
16	46 (8+8+8+9+9+3)	0.920	0.63	0.00	24.57	94464
	45 (8+8+8+8+9+3)	0.948	0.77	0.00	21.50	82176
	44 (7+8+8+8+9+3)	0.983	0.87	0.00	19.96	76032
	43 (7+7+8+8+9+3)	1.032	1.30	0.00	18.43	69888
	42 (6+7+8+8+9+3)	1.078	1.57	0.01	17.66	66816

6.9, we notice that, as observed previously with the GMM-based block quantiser and unconstrained vector quantiser, ISFs perform slightly worser than LSFs, which can amount to a 1 bit/frame difference.

Using the Weighted Mean-Squared-Error as the Distance Measure

It is well known that each quantised LSF[2] influences the reconstructed power spectral envelope in different ways, depending on its location. Figure 6.9 shows the original and reconstructed spectral envelopes when two different LSFs have been shifted. The 15th LSF falls on top of a formant while the 4th LSF is located in a spectral valley. We can see from the Figure that a shift in the 15th LSF results in a shifting of the formant that is more dramatic than the distortion caused by a similar shift to the 4th LSF. Therefore, we can say that 'not all LSFs are considered equal', and a weighted distance measure, that finely quantises the LSFs that are located in the vicinity of a spectral peak, should result in less spectral distortion, as shown in Figure 6.9.

We have adopted a partial implementation of the weighted mean-squared-error measure introduced by Paliwal and Atal [123] in narrowband LSF quantisation, which applies a

[2]We have observed the first 15 ISFs to possess the same properties, hence a similar weighted Euclidean distance measure can be applied. However, the 16th ISF does not possess spectral error localisation properties, but rather, shifting this 'frequency' affects the entire spectrum. Therefore, direct application of a weighted Euclidean distance measure is not as straightforward.

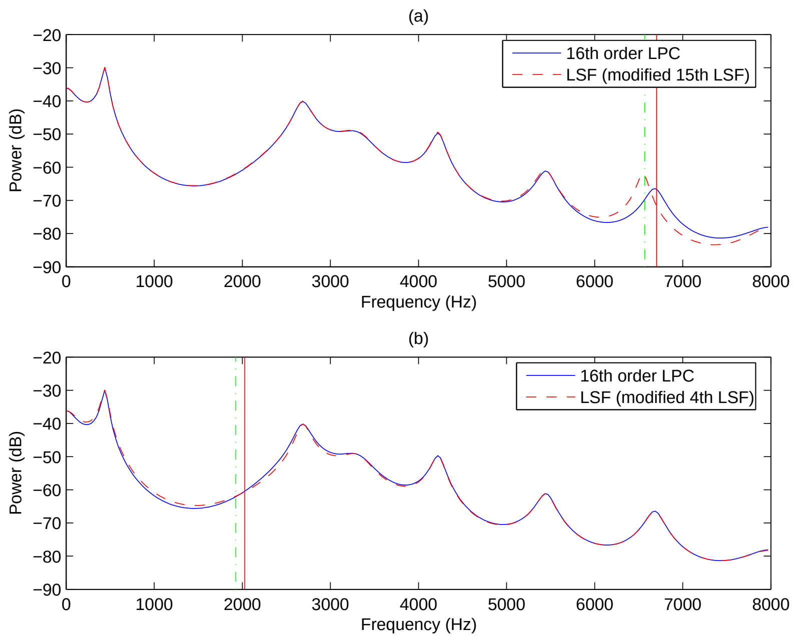


Figure 6.9: Original and reconstructed spectral envelope for a 16th order LPC analysis: (a) shifting the 15th LSF (SD=1.7475 dB); (b) shifting the 4th LSF (SD=0.5441 dB). The solid and dashed vertical lines show the original and shifted LSFs, respectively.

dynamic weighting (that emphasises LSFs in strong regions of the power spectral density) as well as a fixed weighting (which accounts for the difference in sensitivity of the human ear). In this work, our weighted distance measure implements only the dynamic weighting and is used in both the training and searching of the VQ codebooks.

The weighted distance measure, $d_w(\boldsymbol{f}, \hat{\boldsymbol{f}})$, between the original vector, $\boldsymbol{f}$, and the approximated vector, $\hat{\boldsymbol{f}}$, is defined as [123]:

$$d_w(\boldsymbol{f}, \hat{\boldsymbol{f}}) = \sum_{i=1}^{16} \left[w_i (f_i - \hat{f}_i) \right]^2 \tag{6.6}$$

where f_i and $\hat{f}_i$ are the ith LSF in the original and approximated vector respectively. The dynamic weights, $\{w_i\}$ are given by [123]:

$$w_i = [P(f_i)]^r \tag{6.7}$$

where $P(f)$ is the LPC power spectral density and r is a constant (typical value used is 0.15).

Table 6.11: Average spectral distortion (SD) of the five-part switched split vector quantiser using weighted MSE as a function of bitrate and number of switch directions of wideband LSF vectors from the TIMIT database

m	Total bits/frame $(b_1 + b_2 + b_3 + b_4 + b_5 + b_m)$	Avg. SD (in dB)	Outliers (in %) 2–4 dB	Outliers (in %) > 4 dB
8	46 (8+8+9+9+9+3)	0.889	0.33	0.00
	45 (8+8+8+9+9+3)	0.922	0.43	0.00
	44 (8+8+8+8+9+3)	0.953	0.57	0.00
	43 (7+8+8+8+9+3)	0.986	0.66	0.00
	42 (7+7+8+8+9+3)	1.037	1.05	0.00
16	46 (8+8+8+9+9+3)	0.878	0.34	0.00
	45 (8+8+8+8+9+3)	0.906	0.44	0.00
	44 (7+8+8+8+9+3)	0.936	0.50	0.00
	43 (7+7+8+8+9+3)	0.975	0.64	0.00
	42 (6+7+8+8+9+3)	1.018	0.83	0.00

Table 6.11 shows the average spectral distortion performance of the five-part SSVQ using a weighted MSE. By comparing these results with the unweighted MSE-based scheme (Table 6.9), we can see that by using a weighted distance measure, which emphasises specific LSFs that are located near the formant peaks, the SSVQ results in lower spectral distortions and percentages of outlier frames. The SSVQ with weighted MSE can achieve transparent coding at 42 bits/frame. In comparison, the memoryless, five-part split vector quantiser reported by Biundo *et al.* [24] required 45 bits/frame for transparent coding.

Comparing Tables 6.11 and 6.5, we can see that the SSVQ with weighted MSE, which is a memoryless scheme, achieves comparable spectral distortion performance to the S-MSVQ with MA predictor scheme from the AMR-WB speech coder, at 46 bits/frame. In addition, the SSVQ with weighted MSE has produced only half the number of outlier frames than the S-MSVQ, which is to be expected, since the latter has a predictive component.

6.4.9 Multi-Frame GMM-based Block Quantisers

Spectral Distortion Performance when Using 16 Clusters

Table 6.12 shows the average spectral distortion of the 16 cluster, multi-frame GMM-based block quantiser at varying bitrates and number of concatenated frames, p. Unweighted mean-squared-error is used for the cluster quantiser selection. Table 6.14 shows the bitrate independent computational complexity and memory requirements of the multi-frame

Table 6.12: Average spectral distortion as a function of bitrate and number of concatenated frames, p, of the 16 cluster multi-frame GMM-based block quantiser on wideband LSF vectors from the TIMIT database

p	Bits/frame	Avg. SD (dB)	Outliers (in %)	
			2–4 dB	> 4 dB
2	46	0.754	0.09	0.00
	42	0.881	0.29	0.00
	40	0.951	0.52	0.00
	39	0.991	0.69	0.00
	38	1.028	0.91	0.00
	37	1.067	1.23	0.00
3	46	0.725	0.07	0.00
	42	0.845	0.18	0.00
	40	0.911	0.37	0.00
	39	0.946	0.49	0.00
	38	0.983	0.65	0.00
	37	1.021	0.84	0.00
	36	1.060	1.15	0.00
4	46	0.713	0.05	0.00
	42	0.831	0.14	0.00
	40	0.897	0.26	0.00
	39	0.931	0.36	0.00
	38	0.967	0.47	0.00
	37	1.004	0.61	0.00
	36	1.042	0.86	0.00
5	46	0.711	0.02	0.00
	42	0.830	0.10	0.00
	40	0.895	0.18	0.00
	39	0.930	0.28	0.00

GMM-based block quantiser. Comparing this with Table 6.8, we can see that despite the larger dimensionality, the multi-frame GMM-based block quantiser is still computationally more efficient than the single frame GMM-based block quantiser of [183]. This is due to the replacement of the spectral distortion calculation in the cluster quantiser selection with the mean-squared-error, which is computationally less complex.

When quantising 2 frames jointly ($p = 2$), transparent coding is achieved at 39 bits/frame. By comparing this with the memoryless GMM-based block quantiser in Table 6.7, where transparent coding was achieved at 41 bits/frame, the saving of 2 bits/frame by the former may be attributed to the exploitation of correlation between successive pairs of frames. Also, there is a drop in the percentage of outlier frames having spectral distortion between 2 and 4 dB. The $p = 3$ scheme has a moderate tradeoff between distortion and complexity, as shown in Table 6.14, where transparent coding is achieved at 37 bits/frame. As more frames are concatenated, the average spectral distortions and number of outliers decrease, though the benefit of joint quantisation starts to diminish for $p > 4$.

Table 6.13 shows the average spectral distortion performance of the multi-frame GMM-based block quantiser on wideband ISF vectors. It can be observed that the spectral distortions are slightly higher than those in the LSF quantiser. For $p = 2$ and $p = 3$, 40 bits/frame and 38 bits/frame are required for transparent coding of ISFs, which is 1 bit/frame more than for LSFs.

Comparing Tables 6.13 and 6.5, we can see that the multi-frame GMM-based block quantiser outperforms the S-MSVQ from the AMR-WB speech coder in all bitrates. It is particularly interesting to compare the $p = 2$ multi-frame GMM-based block quantiser with S-MSVQ with MA predictor since both these schemes exploit memory across two consecutive frames, where we can see that the former achieves a spectral distortion that is 0.11 dB lower than the latter at 46 bits/frame.

Spectral Distortion Performance when Using 32 Clusters

Table 6.15 shows the average spectral distortion for the 32 cluster, multi-frame GMM-based block quantiser on wideband LSF vectors. Comparing with Table 6.12, we note that the spectral distortion and percentage of outliers are lower. This may be attributed to more accurate modelling of the PDF by using more clusters in the GMM. As we can see from Table 6.14, the computational and memory requirements of the 32 cluster scheme are much higher than those of the 16 cluster one.

6.5 Chapter Summary

This chapter began with the definition of wideband speech and described its advantages over toll-quality narrowband speech, such as improved naturalness and the ability to distinguish between fricatives, as well as provide better presence of the speaker, all of which can alleviate listener fatigue. We have also shown through the visual inspection of LPC-based spectral envelopes that, due to the extra bandwidth, a higher order LPC analysis is required to capture most of the short-term correlation information in the speech. Following

Table 6.13: Average spectral distortion as a function of bitrate and number of concatenated frames, p, of the 16 cluster multi-frame GMM-based block quantiser on wideband ISF vectors from the TIMIT database

p	Bits/frame	Avg. SD (dB)	Outliers (in %)	
			2–4 dB	> 4 dB
2	46	0.781	0.16	0.00
	42	0.910	0.45	0.00
	40	0.983	0.80	0.00
	39	1.021	1.02	0.00
	38	1.060	1.30	0.00
	37	1.100	1.77	0.00
3	46	0.753	0.11	0.00
	42	0.879	0.32	0.00
	40	0.946	0.56	0.00
	39	0.982	0.78	0.00
	38	1.019	1.01	0.00
	37	1.058	1.35	0.00
	36	1.097	1.74	0.00
4	46	0.743	0.07	0.00
	42	0.868	0.24	0.00
	40	0.935	0.46	0.00
	39	0.972	0.56	0.00
	38	1.010	0.81	0.00
	37	1.049	1.08	0.00
	36	1.087	1.40	0.00
5	46	0.744	0.06	0.00
	42	0.867	0.20	0.00
	40	0.934	0.34	0.00
	39	0.970	0.51	0.00

Table 6.14: Bitrate independent computational complexity (in kflops/frame) and memory requirements (ROM) of multi-frame GMM-based block quantiser as a function of the number of concatenated frames, p and number of clusters, m

m	p	kflops/frame	ROM (floats)
16	1	22.29	5120
	2	38.66	18176
	3	55.05	39424
	4	71.43	68864
	5	87.81	106496
32	1	44.58	9984
	2	77.33	36096
	3	110.1	78593
	4	142.9	137472
	5	175.6	212736

Table 6.15: Average spectral distortion as a function of bitrate and number of concatenated frames, p, of the 32 cluster multi-frame GMM-based block quantiser on wideband LSF vectors from the TIMIT database

p	Bits/frame	Avg. SD (dB)	Outliers (in %)	
			2–4 dB	> 4 dB
2	46	0.728	0.07	0.00
	42	0.850	0.21	0.00
	40	0.919	0.34	0.00
	39	0.955	0.44	0.00
	38	0.992	0.62	0.00
	36	1.069	1.13	0.00
3	46	0.700	0.02	0.00
	42	0.817	0.12	0.00
	40	0.882	0.22	0.00
	39	0.916	0.31	0.00
	38	0.951	0.40	0.00
	37	0.987	0.54	0.00
	36	1.026	0.77	0.00
4	46	0.693	0.02	0.00
	42	0.810	0.09	0.00
	40	0.873	0.18	0.00
	39	0.910	0.28	0.00
	38	0.942	0.35	0.00
	37	0.979	0.48	0.00
	36	1.015	0.62	0.00

this, we have given a review of the state-of-the-art coding schemes for wideband speech as well as the industry standard coders such as the ITU-T G.722 (subband/ADPCM coder) and ITU-T G.722.2 (AMR-WB ACELP coder).

Since the focus of this chapter is primarily on spectral quantisation for wideband LPC-based speech coders such as CELP, we provided a review of quantisation schemes that have been reported in the wideband speech coding literature. We have also evaluated some of these schemes such as PDF-optimised scalar quantisers, vector quantisers, and the GMM-based block quantiser on the two competing LPC parameter representations: line spectral frequencies (LSFs) and immittance spectral pairs (ISPs). Our experimental results have shown that ISPs are superior to LSFs by 1 bit/frame in independent quantisation schemes, such as scalar quantisers; while LSFs are the superior representation in joint vector schemes, such as the vector quantiser and GMM-based block quantiser. Through the extrapolation of the operating distortion-rate curve of unconstrained vector quantisation, we also derived an informal lower bound of 35 bits/frame and 36 bits/frame, for the transparent coding of wideband LSFs and ISPs, respectively. We speculate that this may be due to the fact that the last ISP parameter, which is not really a 'frequency', is not correlated with the other ISPs and hence impacts on the memory advantage of block and vector quantisation schemes, which aim to minimise the unweighted MSE for each vector as a whole. Furthermore, because this parameter is a reflection coefficient, it does not possess the error localisation properties of LSFs, but rather propagates errors throughout the entire spectrum. Therefore, additional measures may need to be taken, such as independent quantisation of the last ISP, or use of a weighted distance measure, when vector quantising ISPs.

Finally, we presented and discussed the results of the switched split vector quantiser (SSVQ) and the multi-frame GMM-based block quantiser, for coding wideband LSF and ISF vectors. The SSVQ was able to achieve transparent coding at 43 bits/frame and 44 bits/frame, when using an unweighted mean-squared-error (MSE) on LSFs and ISFs, respectively. The spectral distortion performance of the SSVQ on LSFs was improved by using a weighted MSE that emphasised LSFs located near peaks in the power spectral density. The resulting scheme was transparent at 42 bits/frame. The multi-frame GMM-based block quantiser was able to achieve transparent coding at 37 bits/frame and 38 bits/frame with a moderate computational complexity for LSFs and ISFs, respectively. These two quantisation experiments again confirm our finding that LSFs are superior to ISFs by about 1 bit/frame in joint vector quantisation schemes.

Chapter 7

MFCC Quantisation in Distributed Speech Recognition

7.1 Abstract

This chapter investigates the application of the multi-frame GMM-based block quantisation scheme to MFCC quantisation in distributed speech recognition and examines how it compares with other schemes. The advantage of the multi-frame GMM-based block quantiser is: superior recognition performance at low bitrates, which is comparable with vector quantisation; fixed and relatively low computational and memory complexity that is independent of bitrate; and bitrate scalability, where the bitrate can be dynamically altered without requiring codebook re-training.

We begin the chapter with some background theory on speech recognition, which covers the basic ideas of feature extraction and pattern recognition using hidden Markov models (HMMs). Following this, we provide a general review of client/server-based speech recognition systems and the various types of modes (NSR and DSR) that have been proposed and reported in the literature. We also briefly describe the Aurora-2 DSR experimental framework, which will be used extensively to evaluate the performance and robustness to noise of the various DSR schemes. The second half of the chapter is dedicated to presenting and discussing results of different quantisation schemes applied to a common DSR framework. The schemes investigated include the multi-frame GMM-based block quantiser, the memoryless GMM-based block quantiser, the non-uniform (Lloyd-Max) scalar

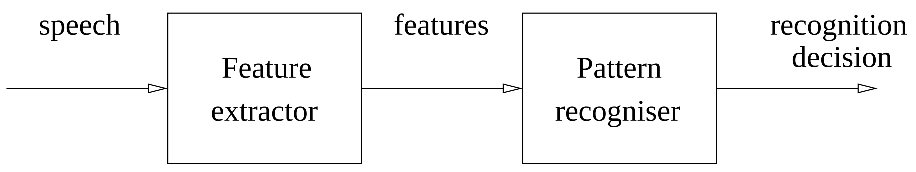


Figure 7.1: A typical speech recognition system (after [178])

quantiser, and vector quantiser. Two sets of experimental results are presented. The first set compares the recognition performance of each quantisation scheme as a function of bitrate in clean and matched conditions. The second set compares the recognition performance of each scheme as a function of SNR in noisy, mismatched conditions.

Publications resulting from this research: [131, 174]

7.2 Preliminaries of Speech Recognition

Figure 7.1 shows a block diagram of a speech recognition system, highlighting the main components in general. In this section, we give only a brief review of each of these components rather than a comprehensive coverage of the algorithms used in modern recognition systems, as the scope of this chapter is focused on the efficient quantisation of MFCC features for distributed speech recognition.

7.2.1 Speech Production

Speech sounds can be broadly classified as either *voiced* or *unvoiced.* Voiced sounds, such as /*iy*/ (as in s*ee*), are periodic and have a harmonic structure that is not present in unvoiced sounds, such as /*s*/, which are aperiodic and noise-like. These are best visualised in Figure 7.2, which shows the waveform and spectrogram of the sentence, *she had your dark suit in greasy wash-water all year*, and highlights the voiced and unvoiced sections in the first word, *she.* Notice that the spectrum for /*sh*/ is flat, similar to that of noise, while the spectrum of /*iy*/ shows a harmonic structure, as characterised by the alternating bands.

Voiced sounds are produced when air from the lungs is excited by vibrating vocal folds in the larynx. A glottal wave, with a fundamental frequency of f_0 and harmonics

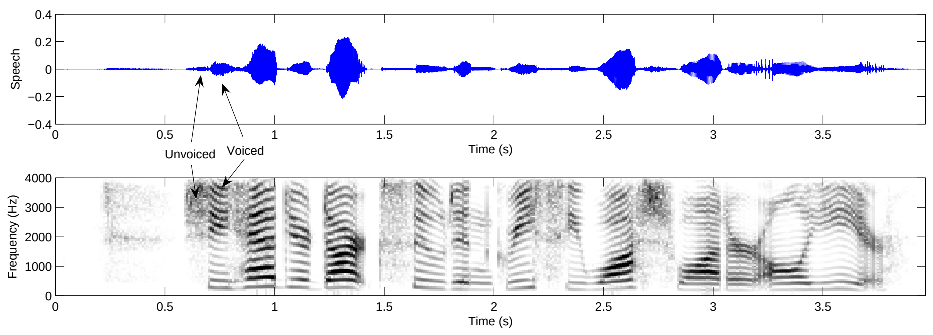


Figure 7.2: Waveform and spectrogram of the sentence, *she had your dark suit in greasy wash-water all year*, highlighting the unvoiced /*s*/ and voiced /*iy*/ sounds in *she*.

at multiples of the fundamental frequency, is generated and this wave passes through the vocal tract, which can be viewed as an acoustic tube that starts at the larynx and terminates at the lips. This tube changes shape to create resonances and anti-resonances that emphasise and de-emphasise certain parts of the spectrum, respectively. *Formants* occur where the spectrum has been emphasised by the resonances of the vocal tract. Along with changes in the articulators (the lips, tongue, jaws, and teeth), different quasi-periodic sounds can be produced [73, 151]. The vocal folds do not vibrate for unvoiced sounds but instead, the vocal tract is constricted by the articulators and air passes through rapidly to produce a noise-like sound [151].

Speech production can be modelled as consisting of a *source* and *filter* component. The source component represents the excitation (which is aperiodic noise for unvoiced sounds and periodic for voiced sounds) while the filter component emphasises parts of the spectrum, just like the vocal tract with its resonances and anti-resonances [151].

7.2.2 Feature Extraction

The role of feature extraction is to compactly represent speech in a way that preserves information that is relevant and important for subsequent recognition [73]. Speech is initially passed through a pre-emphasis filter:

$$H(z) = 1 - \alpha z^{-1} \tag{7.1}$$

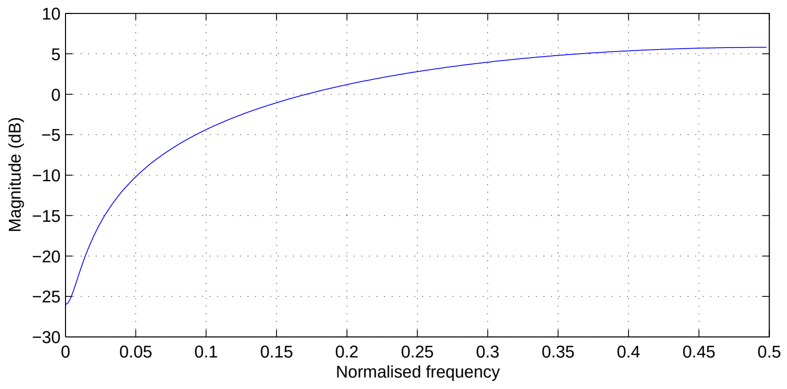


Figure 7.3: Magnitude and phase response of the pre-emphasis filter, $H(z) = 1 - 0.95z^{-1}$

where $\alpha = 0.95$. The magnitude response of this filter is shown in Figure 7.3. The role of the pre-emphasis filter is to remove the spectral tilt (ie. flatten the spectrum) [138], as shown in Figure 7.4(b), where the first formant has been shifted down while the higher frequency formants have been shifted up, allowing them to be analysed at the same level.

Assuming that speech is sampled at 8 kHz, the pre-emphasised speech is then windowed into overlapping segments of 25 ms with 10 ms shift before analysis. A tapered window function, such as the Hamming window, is used to reduce the effects of spectral leakage caused by the blocking process. Acoustic information in speech (eg. formants) manifests itself in the frequency domain[1] (as shown in Figure 7.4), hence the most popular parametric feature sets for speech recognition are spectral-based and can be categorised into two classes: *linear prediction-based* and *Fourier transform-based* [35].

Linear Prediction-Based Features

In linear prediction-based feature extraction, feature vectors are derived from the LPC spectra of speech, which models the vocal tract. Examples include the linear prediction coefficients themselves, reflection coefficients [35], line spectral frequencies [62], etc. In pre-HMM speech recognition systems, the Itakura minimum prediction residual distance measure [76], which calculates the residual error when the tested speech frame is filtered through the reference LPC filter, is used as a distance measure for features based on the LPC spectra. Alternatively, features can be derived from the LPC cepstra and these

[1]The temporal movement of formants, as shown in Figure 7.2, is also useful for speech recognition and is expressed in the delta and acceleration features, which are the first and second derivatives of the spectral-based features [73].

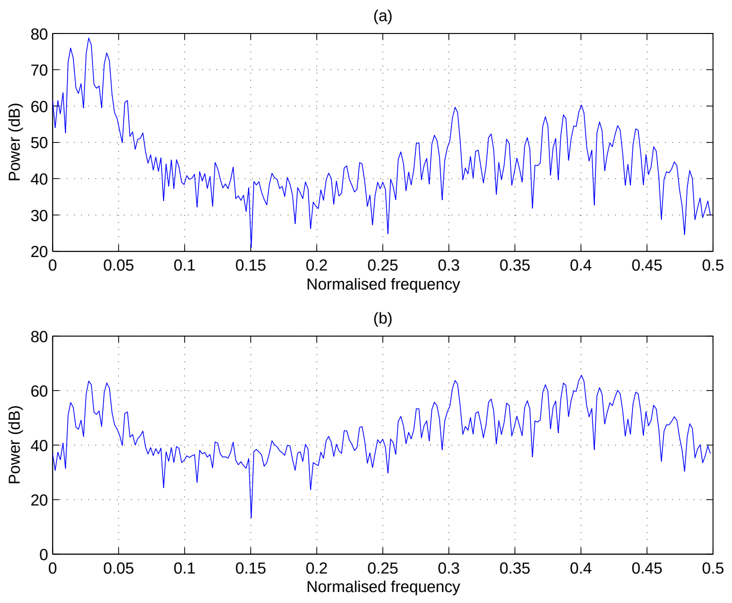


Figure 7.4: The effect of pre-emphasis on the power spectral density of speech: (a) Original PSD of speech; (b) PSD of speech filtered with pre-emphasis filter.

include the linear prediction-based cepstral coefficients (LPCCs) and perceptual linear prediction (PLP) coefficients [67]. The advantage of cepstral-based features is that, because they are derived from an orthogonal basis, simpler Euclidean distance measures can be used [35]. Through a recursive equation, LPCCs are calculated directly from the linear prediction coefficients which represent the speech spectrum in terms of linear frequency. PLP coefficients are calculated in a similar way, where the only difference is that the autocorrelation coefficients used for the linear prediction analysis are derived from the inverse Fourier transform of the Bark frequency-warped power spectral density, obtained using the short-time Fourier transform [73]. The advantage of PLPs over LPCCs is the non-linear frequency scale which matches more closely to the perceptual response of the human auditory system.

Mel Frequency-Warped Cepstral Coefficients

The other class of features are Fourier transform-based and include the Mel frequency-warped cepstral coefficients (MFCCs) and Bark frequency-warped cepstral coefficients (BFCCs), though the former is more commonly used. The discrete Fourier transform is applied to each windowed segment of speech, where the magnitude spectrum is squared to obtain the short-time power spectral density (PSD) or power spectrum of the speech. The PSD is then filtered by a series of M overlapping triangular-shaped filters that are centred on the Mel scale[2]. The filters are overlapped in such a way that the starting and ending frequencies fall on the centres of the previous and next filter, respectively, in order to simulate critical bandwidths [151]. Figure 7.5 shows 20 Mel frequency-warped triangular-shaped filters.

The energy from each filter is accumulated and compressed non-linearly using the natural logarithm to reduce the dynamic range, resulting in a vector of M coefficients for each frame. It has been noted in previous studies that most of the variation in speech can be compactly represented by the first few eigenvectors, whose directional cosines are similar to a cosine series expansion [135, 35]. Hence PCA can be approximated by the discrete cosine transform (DCT), which is applied to the vector of log energies to give the final MFCC vector. The decorrelation aspect of the DCT is a desirable characteristic because of the use of hidden Markov models (HMM) in the subsequent pattern recognition stage. The mixture of Gaussians used in each state of the HMMs use diagonal covariance matrices because of the difficulty in re-estimating off-diagonal components when only a finite training data set is available [138].

The first cepstral coefficient, c_0, may be replaced with the log energy, $\log E$, of the speech frame and this captures the changes in recording level. Also *cepstral mean subtraction* (CMS) is often applied to reduce the effect of convolutional distortion due to changes in the microphone, its distance from the speaker, and the acoustics of the room. In CMS, the mean is removed from the MFCCs [73]. In order to capture temporal changes in the spectra, the approximated first and second derivatives of the MFCC feature vectors are calculated to give the delta and acceleration coefficients, respectively, which are concatenated with the mean-removed MFCCs to give a final feature vector of dimension $3M$ [73]. Typically, speech recognition systems use $M = 13$ MFCC coefficients. Therefore, after concatenating with the delta and acceleration coefficients, the final feature vector has a dimension of 39.

It has been shown in various studies (for example, in [35]) that MFCC features perform

[2]The Melody scale, or more commonly known as the Mel scale, is a non-linear and perceptually-motivated frequency scale. It was derived through perception experiments by Stevens and Volkman [179], where the test subject was asked to manually adjust a stimulus tone so that it perceptually had half the pitch of a given reference tone [157].

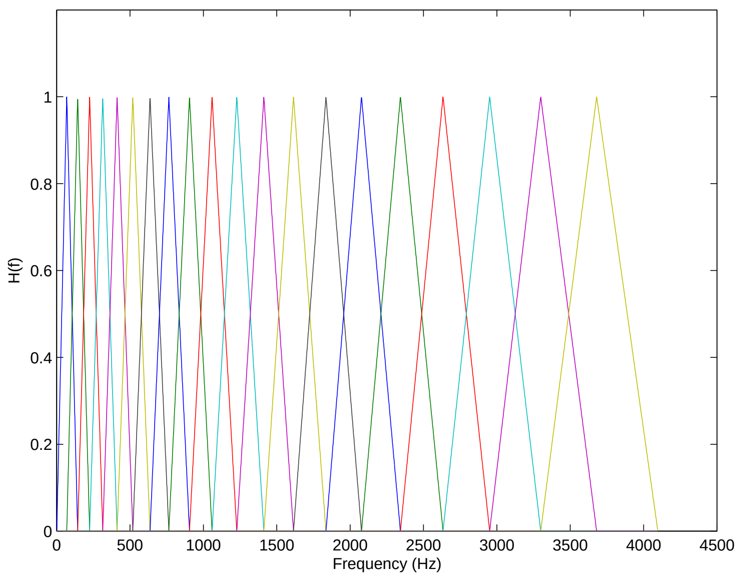


Figure 7.5: Filterbank of Mel frequency-warped triangular-shaped filters used for MFCC calculation from 8 kHz speech ($M = 20$)

better in speech recognition than LPCCs or other linear prediction-based feature sets. Therefore, it is to no surprise that MFCCs are the predominant feature set in modern speech recognition systems.

7.2.3 Pattern Recogniser

The role of the pattern recogniser is to determine which reference template matches the current test feature vector. Of all the pattern recognisers available, the hidden Markov model (HMM) has had the most success in speech recognition. The HMM is a stochastic signal model that consists of a number of probabilistic states [138]. At each instant of time, the model changes from the current state to another state (or, the same state), known as state transition, that is governed by an unobservable or hidden stochastic process. Within each state, there is another stochastic process that produces an output symbol based on a probability distribution, which is usually represented by a Gaussian mixture model.

Typical HMM parameters include the number of states, N, the number of observation vectors per state (for discrete HMMs), M, state transition probability matrix, $\boldsymbol{A}$, output probability matrix, $\boldsymbol{B}$, and initial state distribution, $\boldsymbol{\pi}$ [138]. An HMM is designed from training feature vectors using the Baum-Welch algorithm (also known as the Forward-

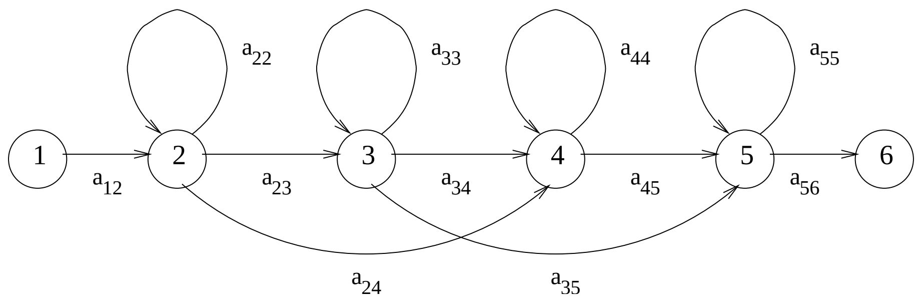


Figure 7.6: A typical 6 state, left-to-right hidden Markov model (HMM) (after [203])

Backward algorithm) which is based on a similar principle to the EM algorithm for GMM estimation [73]. In order to calculate the likelihood, $P(\lambda|\boldsymbol{O})$, of an HMM, λ, generating a given observation sequence, $\boldsymbol{O}$, the Viterbi algorithm is used to select the optimum state sequence that maximises the probability [73]. Therefore, $P(\lambda|\boldsymbol{O})$ can be considered a similarity measure between the given observation sequence and the reference sequence used train the HMM.

In a speech recognition system, an N-state HMM with left-to-right topology, as shown in Figure 7.6, is typically designed for each speech unit (phoneme or word) based on its feature vectors. During the recognition phase, probabilities or likelihoods are calculated for each stored HMM and the one which has the highest likelihood is deemed as matching the given test feature vector. In earlier speech recognition systems, which used discrete HMMs, where each state has a finite set of output symbols, feature vectors are quantised using a vector quantiser before they are matched with all the stored HMM models [138]. In modern speech recognition systems, which use continuous HMMs, the vector quantisation stage is not required [73].

The detailed operation of the HMM is beyond the scope of this work and the reader should refer to Rabiner's HMM tutorial [138] for details.

7.3 Client/Server-Based Speech Recognition

With the increase in popularity of remote and wireless devices such as personal digital assistants (PDAs) and cellular phones, there has been a growing interest in applying automatic speech recognition (ASR) technology in the context of mobile communication systems. Speech recognition can facilitate consumers in performing common tasks, which have traditionally been accomplished via buttons or pointing devices, such as making a call through voice dialing or entering data into their PDAs via spoken commands and sentences. Some of the issues that arise when implementing ASR on mobile devices include: computational and memory constraints of the mobile device; network bandwidth utilisation; and robustness to noisy operating conditions.

Mobile devices generally have limited storage and processing ability which makes implementing a full on-board ASR system impractical. The solution to this problem is to perform the complex speech recognition task on a remote server that is accessible via the network. Various modes of this client/server approach have been proposed and reported in the literature. The most common ones are shown in Figure 7.7 and are discussed in the following subsections.

7.3.1 Network Speech Recognition

In the *Network Speech Recognition* (NSR) mode [85], the user's speech is compressed using conventional speech coders (such as the GSM speech coder) and transmitted to the server which performs the recognition task. In speech-based NSR (Figure 7.7(a)), the server calculates ASR features from the decoded speech to perform the recognition. In bitstream-based NSR (Figure 7.7(b)), the server uses ASR features that are derived from linear predictive coding (LPC) parameters taken directly from the bitstream. Numerous studies have been reported in the literature evaluating and comparing the performance of these two forms of NSR [48, 69, 74, 83, 99, 140, 189, 51].

Literature Review of Speech-Based NSR

Euler and Zinke [48] investigated the effect of three CELP-based speech coders, LD-CELP, RPE-LTP, and TETRA-CELP at 16, 13, and 4.8 kbps, respectively, on isolated word recognition and speaker verification. Narrowband speech was coded and decoded using the CELP coders, and 12 LPCCs and their delta coefficients extracted from the decoded speech. They found that the speech coders operating at 13 kbps and lower decreased the recognition performance in matched and mismatched conditions.

Lilly and Paliwal [99] examined the influence of six speech coders at bitrates ranging from 40 kbps to 4.8 kbps. The tandeming of speech coders and its effect was also investigated. High bitrate ADPCM coders were found to have minimal effect on the recognition while low bitrate CELP coders achieved the lowest recognition accuracy. The same trend was also observed when tandeming. This may be due to the influence of the quantisation noise shaping in CELP coders, which is designed to improve perceptual quality. Spectral information that is important for recognition purposes, is de-emphasised by the noise shaping filter, which exploits spectral masking[3]. MFCCs were also found to be more robust to the effects of speech coders than LPCCs.

Digalakis *et al.* [39] compared the effects of G.721 ADCPM, GSM RTE-LTP, and mu-law on speech recognition performance. Similar to what was observed in [99], the G.721 ADPCM coder incurred minimal degradation while mu-law and GSM achieved the lowest accuracy.

Turunen and Vlaj [189] examined five speech coders and through comparisons and tandeming experiments, identified features that affected the recognition performance. Their first observation was the degrading effect on recognition performance of *postfiltering*, which is applied to smooth the decoded speech and improve its subjective quality. Secondly, the accuracy of the vocal tract model, as represented by the LPC parameters, plays an important role in recognition results. The G.728 LD-CELP does not transmit LPC envelope information but uses a 50th-order backward all-pole predictor. It achieves about 2% less recognition accuracy than the G.729 CS-ACELP coder, which explicitly transmits LPC parameters in the bitstream. Also, other vocal tract models, such as that in the G.727 ADPCM coder which uses a pole-zero model, performed as good as G.729 [189].

Hirsch [69] investigated the influence of the AMR-NB coders at various bitrates and compared the recognition performance with full-rate and half-rate GSM on speech corrupted with noise. Feature extraction was performed using the standard ETSI Aurora frontend and also the advanced noise-robust Aurora frontend. As expected, degradation in recognition performance was observed for coded speech. The performance of the noise-robust Aurora frontend, however, was about 16% higher than when using the standard frontend.

[3]In spectral masking, noise that is in the presence of strong tones will tend to be masked. Therefore, we can afford to increase the amount of noise due to quantisation in the formant regions without affecting the perceptual quality. Noise shaping filters tend to increase the quantisation error in these regions while decreasing, by a similar amount, errors in the spectral valleys [15].

Literature Review of Bitstream-Based NSR

Kim and Cox [83] extracted LPCCs from the spectral envelope information contained in the bitstream. Because speech recognition systems operate on frames sampled at 100 Hz while speech coders process frames at 50 Hz, LSFs from the bitstream were interpolated to the higher frame rate before the LPCCs were extracted. Cepstral liftering was applied and the log energy coefficient calculated from the residual signal. In order to improve the recognition performance further, voiced/unvoiced information from the speech coder was added to the feature.

Huerta and Stern [74] reported their study which examined the derivation of cepstral feature vectors from various parts of the GSM speech coder bitstream and compared the recognition performance with speech-based NSR. One method involved converting the LAR coefficients to LP coefficients and deriving cepstral coefficients (LPCCs). Another involved deriving the cepstrals from the residual signal, as represented by the RPE-LTP parameters. Though the residual signal usually contains only speaker dependent information such as pitch, periodicity, and global waveform information, it still carries some information relevant to speaker independent speech recognition because of the low order (8th) of the LPC analysis [74]. Their results showed the LAR-derived cepstral features to achieve similar performance to speech-based NSR, which had a degraded performance compared with baseline MFCC-based recognition of the original speech. The residual-derived cepstral features did not perform as well though. However, when the LAR-derived and residual-derived cepstral coefficients were concatenated or added to form new features, the recognition accuracy surpassed that of speech-based NSR and was nearly identical to the baseline MFCC-based performance.

Raj *et al.* [140] used a more principled method of combining LPC-derived and residual-derived cepstral coefficients and reported their results for the GSM, CELP and LPC coders. The spectral envelope parameters (LAR coefficients for GSM and LPC or LSFs for CELP), were converted to LP coefficients and LPCCs were derived. The log power spectrum of the residual signal was also calculated and represented as a 32-dimensional vector. The LPCCs and residual log spectra features were concatenated and the extended feature vector reduced in dimensionality using linear discriminant analysis (LDA), whose classes were similar to phoneme classes. The new features achieved the same recognition performance as the baseline system (for GSM and CELP) and were better than speech-based NSR and LPCC-derived features in all cases.

Gallardo-Antolin *et al.* [51] investigated another bitstream-based NSR scheme for GSM speech that was robust to various bit errors such as random errors, burst errors, and frame substitutions. They derived their feature vectors by converting LAR coefficients to LP coefficients and from these, an LPC power spectrum was calculated. MFCCs were then derived from the LPC power spectrum, with the log energy coefficients calculated from analysis of the decoded speech. For error-free speech, the bitstream-derived features performed similarly to those derived from decoded speech. However, as the bit error rate (BER) was increased, the former maintained respectable recognition scores while the latter's performance drops significantly.

7.3.2 Distributed Speech Recognition

In *Distributed Speech Recognition* (DSR), shown in Figure 7.7(c), the ASR system is distributed between the client and server. Here, the feature extraction of speech is performed at the client. These ASR features are compressed and transmitted to the server via a dedicated channel, where they are decoded and input into the ASR backend. Studies have shown that DSR generally performs better than NSR [85] because, in the latter model, speech is processed for optimal perceptual quality and this does not necessarily result in optimal recognition performance [178]. Various schemes for compressing the ASR features have been proposed in the literature. The disadvantage is that DSR requires some modifications to the existing mobile communications infrastructure, such as the addition of a dedicated channel for the transmission of the compressed MFCC feature bitstream.

Digalakis *et al.* in [39] evaluated the use of uniform and non-uniform scalar quantisers as well as product code vector quantisers (split vector quantiser) for compressing MFCCs between 1.2 and 10.4 kbps. They used a greedy-based bit allocation algorithm, where bits were added to each component and the word error rate (WER) was evaluated. The component which resulted in the largest improvement in recognition performance was chosen to receive the allocated bit. This procedure was continued until all bits had been allocated [39]. They concluded that split vector quantisers achieved word error rates (WER) similar to that of scalar quantisers while requiring less bits. Also, PDF-optimised non-uniform scalar quantisers performed better than uniform scalar quantisers, which suggested that the PDF of MFCCs were far from being uniformly distributed. Also,

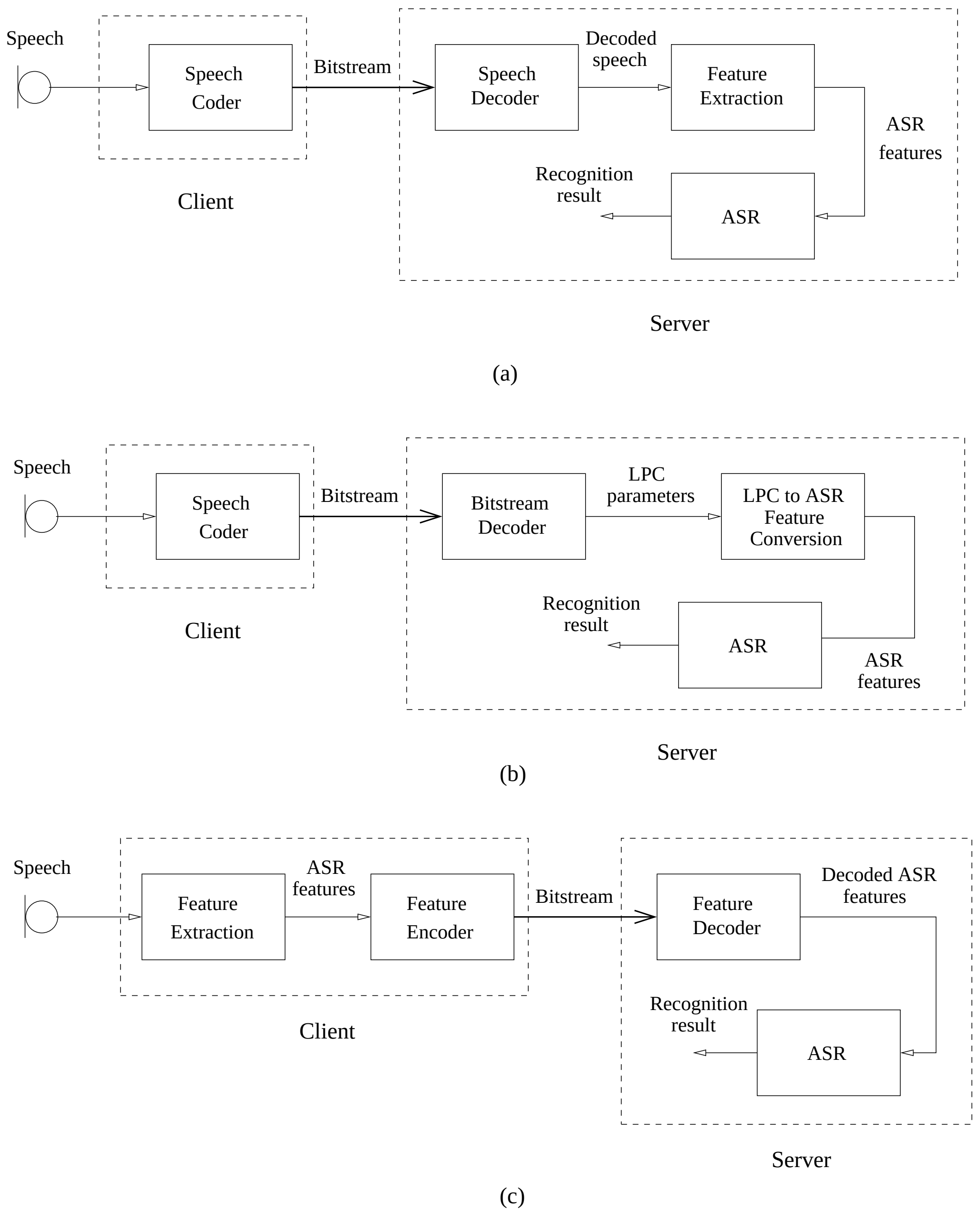


Figure 7.7: Client-server-based speech recognition modes: (a) Speech-based network speech recognition (NSR); (b) Bitstream-based network speech recognition; (c) Distributed speech recognition (DSR)

PDF-optimised scalar quantisation with non-uniform bit allocation performed significantly better than one with uniform bit allocation. They concluded that 2 kbps (20 bits/frame) was the required bitrate for split vector quantisation to achieve unquantised recognition performance.

Ramaswamy and Gopalakrishnan [144] investigated the application of tree-searched multistage vector quantisers with first-order linear prediction operating at 4 kbps (40 bits/frame). The current MFCC feature vector was subtracted from the previous quantised frame to give a residual vector. The first 12 coefficients of the residual vector were then quantised using a two-stage multistage vector quantiser, while the last coefficient, representing c_0, was scalar quantised. Their system achieved near identical recognition performance as the unquantised baseline system, with only minor degradation.

Transform coding, based on the discrete cosine transform (DCT), was investigated in [86] at 4.2 kbps. In this scheme, feature vectors of dimension 14 (13 MFCCs plus the energy coefficient) were processed. For each cepstral coefficient, eight temporally consecutive coefficients were grouped together and processed by the DCT, which exploited temporal correlation between the MFCC frames. The first DCT coefficient was quantised using PCM (12 bits) or DPCM (6 bits) with first order predictor, while the rest of the DCT coefficients were quantised using PCM (18 bits). The energy coefficient was encoded using PCM (12 bits) or DPCM (3 bits). The DPCM coding of the transform coefficients made the scheme insensitive to environmental variations [86].

Zhu and Alwan [206] used a two-dimensional DCT, where 12 successive MFCC frames were stacked together to form a block of 12×12. Zonal sampling was performed, where a fraction of the lowest energy components were set to zero and the remaining transform coefficients were scalar quantised and entropy coded with runlength and Huffman coding. This scheme is similar to the JPEG scheme for image coding. The advantage of this scheme, compared with that of [86], is that both intraframe as well as interframe correlation are exploited by the 2D-DCT. This leads to better energy compaction and hence allow for more data reduction. Noise-robust feature sets, such as peak isolated MFCC (MFCCP) [180] and variable frame-rate peak isolated MFCCs (VFR_MFCCP) [207] were also tested. Their results showed that, firstly, the quantised MFCCs always performed slightly worse than the unquantised MFCCs at all SNR levels. Secondly, the quantised noise-robust features at 624 bps resulted in recognition accuracies that even surpassed the unquantised

MFCCs at low SNRs.

The ETSI DSR standard [47] uses split vector quantisers to compress the MFCC vectors at 4.4 kbps (44 bits/frame). Feature vectors of dimension 14 (13 MFCCs and $\log E$) are split into pairs of subvectors, with the energy parameters, c_0 and $\log E$ belonging to the same pair. A weighted Euclidean distance measure is used for the energy parameter subvector.

Srinivasamurthy *et al.* [178] exploited correlation across consecutive MFCC features by using a DPCM scheme followed by entropy coding. Their scheme is a scalable one, where the bitstream is multiresolution or embedded. That is, a coarsely quantised, *base layer* is transmitted. If higher recognition performance is required, the client can transmit further *enhancement layers* which are combined with the base layer by the server to obtain higher quality features [178].

Bitrate Scalability

Even though vector quantisers generally give better recognition performance using less bits, they are not scalable in bitrate when compared with scalar quantiser-based schemes, such as DPCM and transform coders. In other words, the vector quantiser is designed to operate at a specific bitrate only and will need to be re-trained for other bitrates. *Bitrate scalability* is a desirable feature in DSR applications, since one may need to adjust the bitrate adaptively, depending on the network conditions. For instance, if the communications network is heavily congested, then it may be more acceptable to sacrifice some recognition performance by operating at a lower bitrate in order to offset long response times. In addition to this, the computational complexity of vector quantisers can be quite high, when compared with scalar quantiser-based schemes. This form of scalability contrasts with that mentioned by Sriniwasamurthy *et al.* [178], where the bitstream is embedded with a base (coarse) layer, followed by successive enhancement layers. In order to distinguish between the two forms of scalability, we term this latter form as *bitstream scalability.* In this study, we investigate *bitrate scalable* quantisation schemes only.

7.4 The ETSI Aurora-2 Experimental Framework

The purpose of the ETSI Aurora-2 experiment is to provide a common framework for evaluating noise-robust speech recognition systems. It consists of a clean speech database, a noise database, a standard MFCC-based frontend, and scripts for performing the various training and test sets. The recognition engine that is used is the HMM Toolkit (HTK) software [203].

The TIDigits database [96] forms the basis of the clean speech database, where the original 20 kHz speech was downsampled to 8 kHz and filtered using the frequency characteristic of ITU G.712 (300–3400 Hz). Aurora-2 also provides a database of eight background noises, which were deemed to be commonly encountered in real-life operating conditions for DSR. These noises were recorded at the following places [70]:

- Suburban train (subway)
- Crowd of people (babble)
- Car
- Exhibition hall (exhibition)
- Restaurant
- Street
- Airport
- Train station

This noise is added to the filtered clean speech at various SNRs to simulate noise corruption.

There are two training modes: training with clean speech[4] only and training with clean and noisy (multicondition) speech [70]. In multicondition training, the noises added are subway, babble, car, and exhibition. When training with clean speech only, the best recognition performance is achieved in matched conditions, ie. when testing with clean speech as well. However, when the speech to be tested has background noise, then multicondition training is desirable, as it includes the distorted speech in the training data [70].

[4]Note that all clean speech is filtered using the G.712 frequency characteristic before training.

For the testing, there are three test sets, known as test set A, B, and C. In test set A and B, 4004 test utterances from the TIDigits database are divided into four subsets of 1001 utterances each and four different types of noises are added to each subset at varying levels of SNRs (∞, 20, 15, 10, 5, 0, -5 dB)[5]. Therefore, there are a total of $4 \times 7 = 28$ recognition accuracies reported in test set A and B. In test set C, only two subsets of 1001 utterances and two noises are used, giving a total of 14 recognition accuracies.

In test set A, the subway, babble, car, and exhibition noises are added to each subset and these are the same noises used in multicondition training, hence test set A evaluates the system in matched conditions. In test set B, the other four noises, namely restaurant, street, airport, and train station, are used instead. Because these noises were not present in the multicondition training, then test set B evaluates the system in mismatched conditions (mismatched noise). Test set C contains two utterance subsets only (of the four) with the noises, subway and street, added. Both the speech and noise are filtered using the MIRS frequency characteristic before they are added, hence test set C evaluates the system in mismatched conditions (mismatched frequency characteristic) [70].

Whole word HMMs are used for modelling the digits with the following parameters [69]:

- 16 states per word (with 2 dummy states at beginning and end);
- left-to-right topology without skips over states;
- 3 Gaussian mixtures per state; and
- diagonal covariance matrices

For more details on the HTK reference recogniser and Aurora frontend, the reader should refer to [70, 47].

7.5 Setup of Experiments for Evaluating Different Quantisation Schemes in a DSR Framework

We have evaluated the recognition performance of various quantisation schemes using the publicly available HMM Toolkit (HTK) 3.2 software on the ETSI Aurora-2 database

[5]An SNR of ∞ dB means that no noise is added and we are testing with clean speech.

[70]. Training was done on clean data only (no multicondition training) and testing was performed using test set A. In order to see the recognition performance as a function of bitrate, we focus on the results of testing on *clean speech* (SNR of ∞ dB), where the four word recognition accuracies for each type of noise are averaged to give the final score for the specific quantisation scheme. In addition to this, the effect of different types of noise at varying levels of SNR on the recognition performance is also investigated at the bitrates of 1.2 kbps and 0.6 kbps for each quantisation scheme.

The ETSI DSR standard Aurora frontend [47] was used for the MFCC feature extraction. As a slight departure from the ETSI DSR standard, we have used 12 MFCCs (excluding the zeroth cepstral coefficient, c_0, and logarithmic frame energy, $\log E$) as the feature vectors to be quantised. This was done to maintain consistency in the bitrate-scalable GMM-based block quantisation scheme by avoiding arbitrary bit allocation, as c_0 and $\log E$ are sensitive to changes in recording level of a speech utterance and are generally coded independently [47, 86, 144].

It is well known that lower order cepstral coefficients are particularly sensitive to undesirable variations caused by factors such as transmission, speaker characteristics, and vocal efforts, etc. [80]. As bits are distributed on the basis of the variance of each MFCC, the bit allocations will be particularly sensitive to these spectral variations. In our scalar quantiser experiments, we have found this to degrade the performance of the recognition as too many bits are given to the lower order MFCCs. The bit allocation formula is derived through constrained minimisation of the MSE. However, quantisation based on the reduction of MSE between the original and quantised MFCC feature vector does not necessarily correlate to an improvement in recognition performance. Therefore, in order to reduce the effect of these variations on bit allocation, we have applied the cepstral liftering technique of [80] to the MFCCs using the following lifter window function, $w(n)$ [80]:

$$w(n) = 1 + \frac{L}{2}\sin\left(\frac{\pi n}{L}\right) \quad \text{where } n = 1, 2, \ldots, L \tag{7.2}$$

where L is the feature length. This cepstral liftering procedure is analogous to the weighted distance measure used in LSF quantisation for speech coding. That is, the components of the vector that play a larger role in affecting the final result, which in our case is recognition performance, are emphasised by the lifter window. Cepstral mean subtraction

Table 7.1: Average word recognition accuracy as a function of bitrate and number of clusters for the memoryless GMM-based block quantiser (baseline accuracy = 98.01%)

Bitrate (kbps)	Recognition accuracy (in %)				
	2 cluster	4 cluster	8 cluster	16 cluster	32 cluster
0.3	23.46	19.99	16.69	8.06	8.06
0.4	43.52	53.25	57.67	23.25	9.07
0.6	68.66	79.73	85.72	87.59	82.03
0.8	86.24	90.32	91.45	93.70	94.48
1.0	90.53	94.18	95.03	95.49	96.05
1.2	93.88	95.85	95.94	96.40	96.68
1.5	95.96	96.46	96.96	97.17	97.23
1.7	97.02	96.98	97.16	97.28	97.38
2.0	97.34	97.17	97.54	97.58	97.69
2.2	97.58	97.33	97.62	97.70	97.69
2.4	97.58	97.54	97.69	97.90	97.74
3.0	97.90	97.81	97.87	97.83	97.93
4.4	97.99	97.99	98.09	98.04	98.03

(CMS) is applied to the decoded 12 MFCC features, which are concatenated with their corresponding delta and acceleration coefficients, giving the final feature vector dimension of 36 for the ASR system. The HTK parameter type is `MFCC_D_A_Z`. The baseline average recognition accuracy using unquantised MFCC features is 98.01%.

In the training of the single frame and multi-frame GMM-based block quantiser, 20 iterations of the EM algorithm were used to generate a 16 and 32 cluster GMM.

7.6 Recognition Performance of the Memoryless GMM-Based Block Quantiser

Table 7.1 shows the recognition accuracy for the memoryless GMM-based block quantiser at various bitrates and number of clusters. At 2 kbps, the recognition accuracy is roughly the same as the unquantised scheme. Between 2 kbps and 800 bps, the recognition performance gradually decreases where it can be seen that the higher cluster schemes maintain a higher accuracy. This may be attributed to the more accurate modelling of the source PDF by using more clusters. Since it has been shown in other studies [128] that using more clusters generally will reduce the quantisation distortion incurred at a fixed bitrate, it would be expected to indirectly lead to better recognition.

Table 7.2: Average word recognition accuracy as a function of bitrate and number of frames for 16 cluster multi-frame GMM-based block quantiser (baseline accuracy = 98.01%)

Bitrate (kbps)	Recognition accuracy (in %)			
	2 frames	3 frames	4 frames	5 frames
0.3	78.26	89.59	91.30	92.96
0.4	91.07	94.32	95.05	95.36
0.6	95.52	96.62	97.05	96.78
0.8	96.92	97.27	97.41	97.52
1.0	97.40	97.61	97.74	97.71
1.2	97.56	97.74	97.75	97.89
1.5	97.78	97.80	97.86	97.83
1.7	97.80	97.96	97.99	97.96
2.0	97.99	97.89	98.06	97.97
2.2	98.03	97.97	97.94	98.04

At bitrates below 800 bps, the recognition performance drops dramatically, where we have the situation of higher clusters leading to steeper decreases. For MFCC frames containing no information (all zero), the recognition accuracy is 8.06%. This situation may be explained by the shortage of bits to be allocated to all clusters. A 16 cluster quantiser, for instance, requires at least 4 bits in total to be able to uniquely identify each cluster (assuming a uniform allocation of levels) while a 32 cluster block quantiser requires at least 5 bits[6]. Therefore, the single frame GMM-based block quantiser performs poorly when the number of bits approaches $\log_2 m$, where m is the number of clusters.

7.7 Recognition Performance of the Multi-Frame GMM-Based Block Quantiser

Table 7.2 shows the average word recognition accuracy of the 16 cluster multi-frame GMM-based block quantiser for different bitrates and number of frames. It can be observed that this quantiser achieves an accuracy close to the unquantised, baseline system at 1 kbps or 10 bits/frame, which is half the bitrate of the single-frame GMM-based block quantiser. For bitrates lower than 600 bps, the performance gradually rolls off.

[6]In reality, quantiser levels are non-uniformly allocated in this scheme, so most of the available quantiser levels will be allocated to only a fraction of the cluster block quantisers. The decoder will be able to determine which cluster block quantisers are operational since it performs an identical bit allocation using the same stored models as the encoder.

Table 7.3: Average word recognition accuracy as a function of bitrate and number of clusters for 5 frame multi-frame GMM-based block quantiser (baseline accuracy = 98.01%)

Bitrate (kbps)	Recognition accuracy (in %)	
	16 clusters	32 clusters
0.2	82.94	87.70
0.3	92.96	94.20
0.4	95.36	96.03
0.6	96.78	97.06
0.8	97.52	97.58
1.0	97.71	97.57
1.2	97.89	97.89
1.5	97.83	97.93
1.7	97.96	97.96
2.0	97.97	97.95

In terms of quantiser distortion, the multi-frame GMM-based block quantiser generally performs better as more frames are concatenated together because interframe memory can be exploited by the KLT. Also, because the dimensionality of the vectors is high, the block quantisers operate at a higher rate. Comparing Table 7.2 with the 16 cluster column of Table 7.1, it can be observed that there is a trend between using more frames to reduce MFCC frame distortion and improving the recognition accuracy, at low bitrates. In other words, the average recognition accuracy gets progressively better as more and more frames are jointly quantised. At 300 bps, the recognition accuracy of jointly quantising five frames is roughly 14% higher than quantising 2 frames only. However, this comes at the expense of higher delay, computational, and memory requirements.

Compared with the results of the single frame GMM-based block quantiser in Table 7.1, the multi-frame scheme does not suffer from a dramatic drop in recognition accuracy at low bitrates. Unlike the single frame scheme, where there was a shortage of bits to distribute among clusters, the multi-frame GMM-based block quantiser is able to provide enough bits, thanks to the increased dimensionality of the vectors. For example, at 300 bps, a 16 cluster, single frame GMM-based block quantiser has a total bit budget of 3 bits. On the other hand, a 16 cluster, 2 frame multi-frame GMM-based block quantiser has 6 bits while a 3 and 4 frame scheme has 9 and 12 bits, respectively. Therefore, the multi-frame GMM-based block quantiser can operate at lower bitrates while maintaining good recognition performance.

Table 7.4: Average word recognition accuracy as a function of bitrate for non-uniform scalar quantiser (baseline accuracy = 98.01%)

Bitrate (kbps)	Recognition accuracy (in %)
0.6	38.17
0.8	72.31
1.0	86.68
1.2	93.27
1.5	95.45
1.7	96.17
2.0	96.97
2.2	97.21
2.4	97.40
3.0	97.76
4.4	97.96

Table 7.3 shows the average word recognition accuracy of a 16 cluster and 32 cluster multi-frame GMM-based block quantiser, where the number of frames is fixed at 5. As expected, using more clusters to reduce the quantised MFCC distortion has led to an improvement in recognition accuracy, at the cost of an increase in complexity and memory.

7.8 Comparison with the Recognition Performance of the Non-Uniform Scalar Quantiser

For the scalar quantisation experiment, each MFCC was quantised using a non-uniform Gaussian Lloyd-Max scalar quantiser whose bit allocation was calculated using the high resolution formula of (2.43). We have chosen this method over the WER-based greedy algorithm of [39] because of its computational simplicity and this allows us to scale any bitrate with ease.

Table 7.4 shows the average recognition accuracy of the non-uniform scalar quantiser. It can be seen that the accuracy decreases linearly in the range of 4.4 to 1.2 kbps and drops rapidly below this range. Comparing Table 7.4 with Tables 7.1 and 7.2, the GMM-based block quantisers use less bits than the non-uniform scalar quantiser to achieve a certain level of recognition accuracy. This may be attributed to the effectiveness of the KLT as a decorrelator and energy compactor, as well as the better modelling of the source PDF.

Table 7.5: Average word recognition accuracy, computational complexity (in kflops/frame), and memory requirements (ROM) as a function of bitrate for vector quantiser (baseline accuracy = 98.01%)

Bitrate (kbps)	Recognition accuracy (in %)	kflops/frame	ROM (in floats)
0.4	76.94	0.77	192
0.6	91.83	3.07	768
0.8	95.65	12.29	3072
1.0	96.85	49.51	12288
1.2	97.01	196.7	49152

7.9 Comparison with the Recognition Performance of the Unconstrained Vector Quantiser

An unconstrained, full-search vector quantiser was used to quantise single MFCC frames. In terms of minimising quantiser distortion, the vector quantiser is considered the optimum coding scheme [55], hence it will serve as an informal upper recognition bound for single frame quantisation and highlight the effectiveness of the multi-frame GMM-based block quantiser in exploiting interframe memory. Table 7.5 shows the average recognition accuracies at several bitrates as well as the computational and memory requirements of the vector quantiser. Comparing this with Table 7.1, the vector quantisation scheme achieves higher recognition than the single frame GMM-based block quantiser for all bitrates, which is consistent with the fact that the vector quantiser will always incur the least distortion of all quantisation schemes for a given dimension. Comparing with the performance of the multi-frame GMM-based block quantiser in Table 7.2, it can be observed that this scheme gives higher recognition accuracies than the vector quantiser for all bitrates considered. Even the 2 frame, multi-frame GMM-based block quantiser does better than the vector quantiser. Hence this shows that there is a considerable amount of correlation between MFCC frames that can be exploited by quantisation schemes.

As can be seen from Table 7.5, the computational complexity and memory requirements of the vector quantiser are dependent on the bitrate and can be quite high at medium bitrates like 1.2 kbps. On the other hand, the complexity of the GMM-based block quantiser, as shown in Table 7.6, is constant for all bitrates. Also of note is that, unlike the GMM-based block quantiser, the vector quantiser is *not bitrate scalable*.

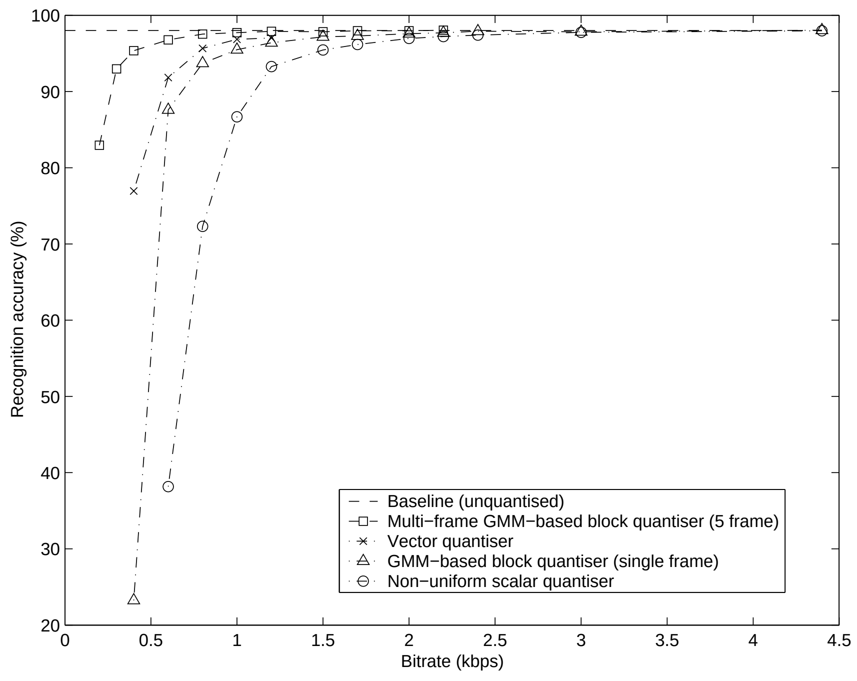


Figure 7.8: Summary of average word recognition accuracies for all quantisation schemes considered

Table 7.6: Bitrate independent computational complexity (in kflops/frame) and memory requirements (ROM) of the multi-frame GMM-based block quantiser as a function of number of concatenated vectors, p, and number of clusters, m

m	p	kflops/frame	ROM (floats)
16	1	13.65	3136
	2	22.86	10624
	3	32.07	22720
	4	41.28	39424
	5	50.50	60736
32	1	27.30	4416
	2	45.71	14976
	3	64.14	31936
	4	82.57	55296
	5	101.0	121216

7.10 Effect of Additive Noise on Recognition Performance

The effect of undesirable noise on the recognition performance is important and relevant to DSR systems, since the operator will most likely be immersed in background environmental noise that will also be captured by his/her mobile device. The Aurora-2 recognition task provides various types of background noise that is added to the clean speech at various SNR levels ($20, 15, 10, 5, 0, -5$ dB). In test set A, the four noises added are suburban train (subway), babble, car, and exhibition hall [70].

In this section, we evaluate the word recognition accuracy for all quantisation schemes, on speech corrupted with additive noise, as a function of SNR. The recognition models are trained on *clean speech* only (no multicondition training). The bitrates tested are 1.2 kbps (12 bits/frame) and 0.6 kbps (or 6 bits/frame) for all quantisation schemes. The notation we have used to abbreviate the quantisation schemes are as follows:

- GMM-5 is the five frame, multi-frame GMM-based block quantiser;
- GMM-1 is the memoryless GMM-based block quantiser;
- VQ is the unconstrained vector quantiser; and
- SQ is the non-uniform scalar quantiser.

Tables 7.7, 7.8, 7.9, and 7.10 shows the word recognition accuracy at 1.2 kbps when speech is corrupted with subway, babble, car, and exhibition noise, respectively. The results for the original, unquantised scheme are given for comparative purposes. We can see that the multi-frame GMM-based block quantiser (GMM-5) generally achieves the highest recognition accuracies of all quantisation schemes with the scalar quantiser performing the worst. The vector quantiser (VQ) is theoretically the best quantiser for a given dimension for minimising distortion, and this correlates generally to recognition performance, where we observe it outperforming the memoryless GMM-based block quantiser (GMM-1). Figure 7.9 shows the recognition accuracy for each quantisation scheme operating at 1.2 kbps, plotted against SNR for each of the noises. We can see that the quantisation schemes which exploit memory (GMM-5, GMM-1 and VQ) maintain a recognition that is close to the baseline, as opposed to the memoryless scalar quantiser which degrades rapidly as noise is added (particularly at SNRs of 20 dB and 15 dB).

Table 7.7: Word recognition accuracy for speech corrupted with subway noise at varying SNRs (in dB) at 1.2 kbps.

Quantisation	Recognition accuracy (in %)						
scheme	∞ dB	20 dB	15 dB	10 dB	5 dB	0 dB	−5 dB
Unquantised	98.07	94.14	86.67	66.17	38.62	23.43	16.12
GMM-5	97.64	92.48	82.68	58.89	32.61	21.86	15.23
VQ	97.11	92.26	81.30	59.32	31.62	19.65	14.03
GMM-1	96.44	89.13	77.40	50.78	27.11	19.37	13.82
SQ	92.85	68.47	48.42	30.61	22.35	17.38	12.56

Table 7.8: Word recognition accuracy for speech corrupted with babble noise at varying SNRs (in dB) at 1.2 kbps

Quantisation	Recognition accuracy (in %)						
scheme	∞ dB	20 dB	15 dB	10 dB	5 dB	0 dB	−5 dB
Unquantised	98.07	95.92	90.69	74.94	45.56	22.91	12.64
GMM-5	98.13	94.98	87.30	65.36	36.73	20.80	12.12
VQ	97.16	92.93	85.01	65.11	36.19	19.62	10.67
GMM-1	96.58	91.48	81.92	60.94	34.61	19.35	10.94
SQ	93.80	67.87	47.64	29.87	21.31	16.51	10.28

Table 7.9: Word recognition accuracy for speech corrupted with car noise at varying SNRs (in dB) at 1.2 kbps

Quantisation	Recognition accuracy (in %)						
scheme	∞ dB	20 dB	15 dB	10 dB	5 dB	0 dB	−5 dB
Unquantised	97.97	95.59	88.88	68.42	36.09	20.61	13.30
GMM-5	97.88	93.80	83.21	55.92	29.38	18.52	12.73
VQ	97.02	93.14	83.12	54.94	27.02	19.27	11.78
GMM-1	96.39	91.05	78.08	49.42	25.17	18.01	11.24
SQ	93.44	70.44	45.87	27.86	22.19	16.94	11.72

Table 7.10: Word recognition accuracy for speech corrupted with exhibition noise at varying SNRs (in dB) at 1.2 kbps

Quantisation	Recognition accuracy (in %)						
scheme	∞ dB	20 dB	15 dB	10 dB	5 dB	0 dB	−5 dB
Unquantised	97.93	93.34	85.56	62.79	33.42	19.01	10.74
GMM-5	97.90	92.84	81.70	54.83	28.60	18.94	10.86
VQ	96.73	91.36	77.63	50.45	26.87	16.94	10.77
GMM-1	96.24	89.45	75.84	45.63	25.42	17.46	11.66
SQ	92.97	72.79	48.32	28.63	20.73	13.82	8.89

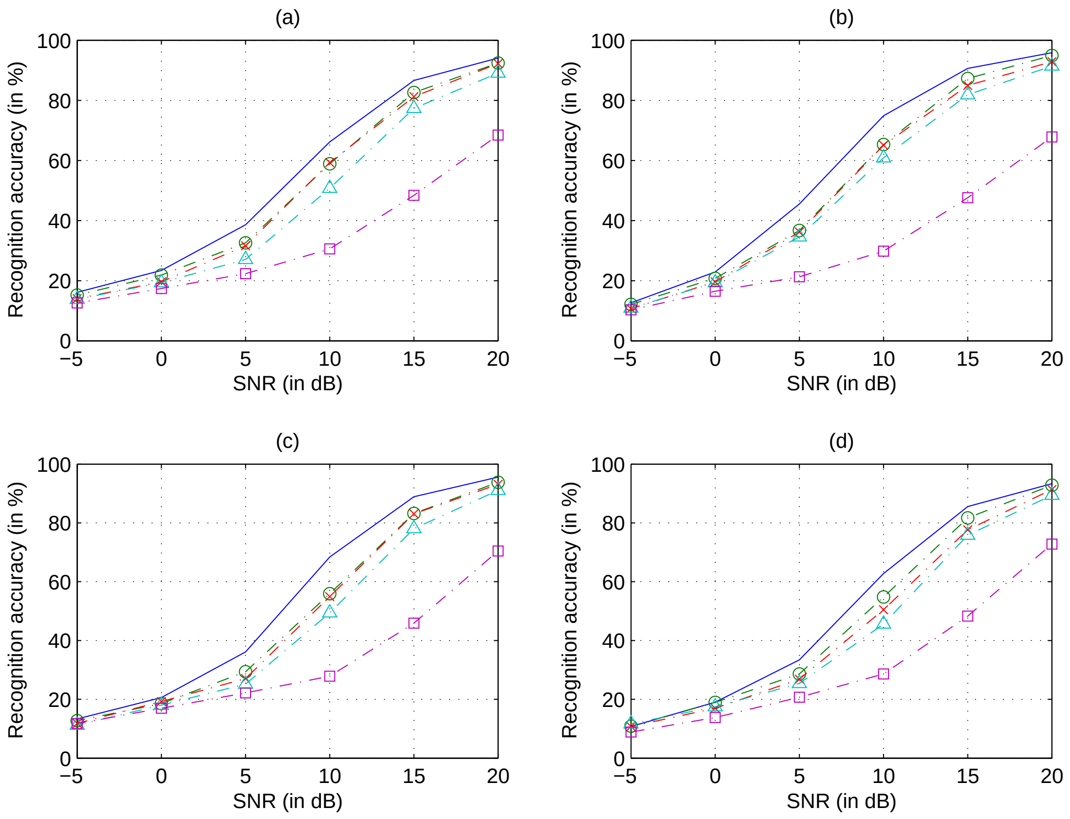


Figure 7.9: Plot of recognition accuracy versus SNR for all quantisation schemes at 1.2 kbps: (a) subway noise; (b) babble noise; (c) car noise; and (d) exhibition noise. (Solid lines are unquantised, circles are GMM-5, crosses are VQ, triangles are GMM-1, squares are SQ)

Table 7.11: Word recognition accuracy for speech corrupted with subway noise at varying SNRs (in dB) at 0.6 kbps.

Quantisation	Recognition accuracy (in %)						
scheme	∞ dB	20 dB	15 dB	10 dB	5 dB	0 dB	-5 dB
Unquantised	98.07	94.14	86.67	66.17	38.62	23.43	16.12
GMM-5	96.38	88.73	74.33	48.17	26.93	18.73	13.57
VQ	94.40	82.22	71.29	48.30	26.44	15.84	11.21
GMM-1	84.41	77.56	64.14	44.24	25.15	16.43	11.36
SQ	8.32	8.29	8.29	8.26	8.14	8.11	8.07

Table 7.12: Word recognition accuracy for speech corrupted with babble noise at varying SNRs (in dB) at 0.6 kbps

Quantisation	Recognition accuracy (in %)						
scheme	∞ dB	20 dB	15 dB	10 dB	5 dB	0 dB	-5 dB
Unquantised	98.07	95.92	90.69	74.94	45.56	22.91	12.64
GMM-5	97.10	91.54	77.90	53.78	30.05	18.02	11.25
VQ	91.66	83.25	72.79	52.39	29.96	16.35	10.34
GMM-1	89.94	76.12	62.61	43.02	25.94	15.90	10.13
SQ	8.25	8.22	8.16	8.13	8.16	8.13	8.13

Tables 7.11, 7.12, 7.13, and 7.14 shows the word recognition accuracy at 0.6 kbps when speech is corrupted with subway, babble, car, and exhibition noise, respectively. Similar to the previous case at the higher bitrate of 1.2 kbps, the multi-frame GMM-based block quantiser generally achieves higher recognition accuracies than the other schemes, with the scalar quantiser being the worst. There are cases where the GMM-1 scheme achieves a slightly higher recognition performance, such as for an SNR of 0 dB of subway noise, but this discrepancy is insignificant. Figure 7.10 shows the recognition accuracy at 0.6 kbps as a function of SNR. It is particularly interesting to note that the difference in recognition performance between the multi-frame and memoryless GMM-based block quantiser at 0.6 kbps is larger than that observed at 1.2 kbps (in Figure 7.9), for medium to high SNRs. For instance, when speech is corrupted with babble noise with the SNR at 15 dB, the multi-frame GMM-based block quantiser achieves a recognition accuracy that is about 15% higher than the memoryless GMM-based block quantiser at 0.6 kbps, while at 1.2 kbps, that difference is only 5%. This shows that there is an advantage in using more efficient quantisation schemes at moderately high SNRs and low bitrates.

Another interesting observation, especially from the low bitrate results, is the dimin-

Table 7.13: Word recognition accuracy for speech corrupted with car noise at varying SNRs (in dB) at 0.6 kbps

Quantisation	Recognition accuracy (in %)						
scheme	∞ dB	20 dB	15 dB	10 dB	5 dB	0 dB	−5 dB
Unquantised	97.97	95.59	88.88	68.42	36.09	20.61	13.30
GMM-5	96.51	89.02	72.89	44.92	24.22	17.00	11.03
VQ	91.92	84.22	70.00	44.02	23.11	15.81	10.86
GMM-1	87.59	76.86	59.86	36.92	21.00	14.70	11.09
SQ	8.29	8.23	8.29	8.26	8.23	8.23	8.23

Table 7.14: Word recognition accuracy for speech corrupted with exhibition noise at varying SNRs (in dB) at 0.6 kbps

Quantisation	Recognition accuracy (in %)						
scheme	∞ dB	20 dB	15 dB	10 dB	5 dB	0 dB	−5 dB
Unquantised	97.93	93.34	85.56	62.79	33.42	19.01	10.74
GMM-5	97.13	89.60	73.25	43.04	24.13	16.94	9.60
VQ	92.35	84.60	70.01	42.58	22.80	14.47	10.86
GMM-1	87.41	78.96	59.86	34.16	20.18	12.74	9.16
SQ	7.93	7.87	7.87	7.84	7.81	7.81	7.81

ishing advantage, as the SNR degrades, of the more efficient quantisation schemes, like GMM-5, over the less efficient ones, such as GMM-1. For example, testing on clean speech and speech with an SNR of 20 dB, the GMM-5 scheme at 0.6 kbps scheme mostly achieves roughly 10% or better recognition performance over GMM-1 at the same bitrate. This advantage diminishes as more and more noise is added, with the difference reduced to roughly 2% or less at an SNR of −5 dB. A similar trend can also be seen in Figure 7.10, when comparing VQ with GMM-1, where the advantages of VQ, in terms of lower MSE distortion, diminish as more noise is added. We can therefore conclude that, firstly, the advantages of using more efficient quantisation schemes manifest themselves more at SNRs of 10 dB and higher. Also for DSR in noisy environments where the SNR is very low, the noise robustness of the underlying speech recognition system becomes the dominant factor, rather than MFCC quantisation efficiency, when it comes to recognition performance. This is, in fact, consistent with the results from [206], where the recognition performance as a result of using quantised MFCCs, was always worse than when using the unquantised MFCCs, for all levels of SNR. By quantising more noise-robust features, such as those used in [206] (MFCCPs and VFR_MFCCPs), better recognition accuracy can be achieved.

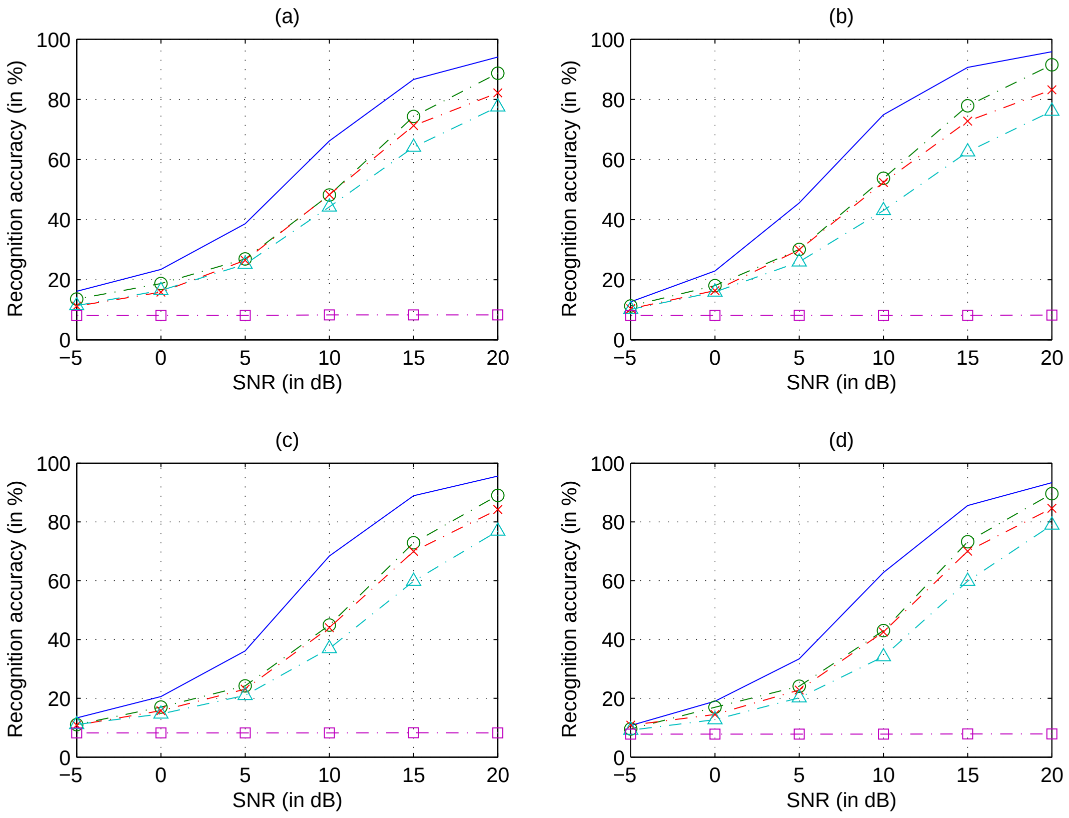


Figure 7.10: Plot of recognition accuracy versus SNR for all quantisation schemes (excluding SQ) at 0.6 kbps: (a) subway noise; (b) babble noise; (c) car noise; and (d) exhibition noise. (Solid lines are unquantised, circles are GMM-5, crosses are VQ, triangles are GMM-1, squares are SQ)

7.11 Chapter Summary

In this chapter, we provided a brief review of automatic speech recognition with particular emphasis on the speech features such as the Mel frequency-warped cepstral coefficients. Following this, we reviewed the literature that investigated various modes of client/server-based speech recognition systems, such as network speech recognition and distributed speech recognition. The experimental database used for evaluating the performance of our MFCC quantisation schemes as well as the parameters for the recognition task were described given in detail. Next, we presented our results on MFCC quantisation in a DSR framework using multi-frame GMM-based block quantisers and compared its performance against the memoryless GMM-based block quantiser, the non-uniform scalar quantiser, and the unconstrained vector quantiser. The multi-frame GMM-based block quantiser achieved better recognition at lower bitrates, exhibiting negligible degradation of 1% (WER of 2.5%) in recognition performance over the baseline system at 800 bps and 5% (WER of 7%) at 300 bps. Unlike vector quantisation schemes, the multi-frame GMM-based block quantiser is scalable in bitrate and has a complexity that is independent of bitrate. The performance of the multi-frame GMM-based block quantiser in the presence of noise was also evaluated. It was found that the recognition performance of relatively high SNRs was influenced mostly by the quantisation scheme. However, at low SNRs, the effect of quantisation efficiency diminishes and recognition performance is dependent on the noise robustness of the underlying features.

Chapter 8

Conclusions and Future Research

8.1 Chapter Summary and Conclusions

This dissertation has examined block and vector quantisation schemes that are efficient, in terms of rate-distortion and computational requirements, and reported on their performance in four unique application areas. In this section, we give a summary as well as present the findings and conclusions of each chapter.

8.1.1 Chapter 2: Efficient Block Quantisation

This chapter provided a general introduction to block quantisation, which is an example of a transform coder. The decorrelating properties of the Karhunen-Loève transform and its role in block quantisation were described. We also reviewed the discrete cosine transform as a useful alternative transform to the KLT. For sources which have Gauss-Markov properties, the DCTs decorrelating ability is similar to that of the KLT, hence the DCT is popularly used in image coding.

We provided a literature review of adaptive transform coding schemes, which resolve the problems of data non-stationarity by partitioning the vector space into local regions and designing transforms adapted to the statistics of each region. Additionally, a simple scheme using K-means clustering and local block quantisers was described and this formed a useful baseline for evaluating the recent schemes that utilise Gaussian mixture models for estimating the PDF. Following this, we gave a detailed summary of the GMM-based block quantisation scheme of [183].

We presented our modification to the GMM-based block quantiser, that replaces the KLT with a DCT. Due to the data independence property and fixed orthogonal bases of the DCT, the complexity of the new GMM-DCT-based block quantiser is considerably lowered. This modified scheme is expected to be competitive with the KLT-based GMM-based block quantiser for image coding, since images tend to have Gauss-Markov statistics and are highly correlated. We also described our multi-frame GMM-based block quantiser, that exploits interframe correlation using the KLT by concatenating successive frames into larger ones.

A new bit encoding technique was introduced that allows the use and encoding of fractional bits in a fixed-rate block quantiser. This scheme uses the concept of a generalised positional number system and is simple in implementation. To complement this fractional bit technique, we also described some heuristic algorithms for dealing with bit allocation issues.

8.1.2 Chapter 3: Efficient Vector Quantisation

This chapter provided a general review of vector quantisation, its advantages over the scalar quantiser, and its limitations, with regards to its exponential growth of complexity as a function of the number of bits and dimensionality. Product code vector quantisers, such as the split and multistage vector quantiser, alleviate the complexity issue by dividing the quantisation process into codebooks of lower dimensionality, or sequential and independent stages, respectively. These structural constraints though cause suboptimal quantisation performance. We have also identified and analysed the main source of suboptimality in the split vector quantiser (SVQ), namely the vector splitting which degrades the memory advantage, the shape advantage, and the space-filling advantage. In order to address at least two of these suboptimalities, we have introduced a new type of product code vector quantiser called the switched split vector quantiser (SSVQ), which consists of a hybrid of a full-dimension, unconstrained switch vector quantiser and numerous split vector quantisers. The first stage (ie. switch vector quantiser) allows the SSVQ to exploit global statistical dependencies as well as match the marginal PDF shape of the data, which would otherwise have not been exploited by normal SVQ. Also, the tree structured characteristic of the switch vector quantiser provides a dramatic reduction in search complexity. We have shown via computer simulations of 2-D vector quantisation how SSVQ is superior to SVQ in terms of quantisation performance and computational complexity. The only disadvantage of SSVQ is the increase in memory requirements.

8.1.3 Chapter 4: Lossy Image Coding

In this chapter, we have presented a comprehensive literature review of image coding techniques, which includes vector quantisation, transform coding, and subband and wavelet-based coding. Fundamental to image subband coding and image processing in general, is the non-expansive filtering of data with a finite length. Similar to the procedure given in [106], the symmetric extension method was examined in depth with examples provided for even and odd tapped filters as well as filters with unequal lengths.

The remainder of the chapter was dedicated to the results and discussion of various quantisation schemes, such as the block quantiser based on the KLT and DCT and the GMM-based block quantiser. It was shown that the GMM-based block quantiser achieved higher PSNRs and better subjective quality than the traditional fixed-rate block quantiser/transform coder at a given bitrate, which demonstrates the advantages of accurate source PDF estimation and the use of multiple decorrelating transforms. Because images are highly correlated and have Gauss-Markov properties, replacing the KLT with the data dependent DCT should result in comparable performance. Through PSNRs and visual inspection, we showed that the GMM-DCT-based block quantiser is comparable in quantisation performance, with only a fraction of the complexity. Next, a novel and low complexity method of encoding fractional bits in a fixed-rate framework and heuristic algorithms for compensating quantiser levels in bit allocation were evaluated and shown to improve the PSNR slightly. Finally, we presented a method of pre-processing an image using the wavelet transform before block quantisation that reduces block artifacts and improves the image quality.

8.1.4 Chapter 5: LPC Parameter Quantisation in Narrowband Speech Coding

In this chapter, we first reviewed the basics of speech coding, such as speech production and the modelling of speech using linear prediction analysis. The operation of various speech coders was also described and this highlighted the role and importance of LPC quantisation. Different LPC parameter representations, that are both robust to quantisation and provide simple checks for filter stability, were covered. The line spectral frequencies are one of the more popular representations and were thus used in our evaluation of various quantisation schemes.

The first quantisation scheme that we evaluated was the multi-frame GMM-based block quantiser, which has the advantage of bitrate scalability and bitrate independent complexity. By extending the decorrelating transform to exploit the linear dependencies between multiple frames, the multi-frame GMM-based block quantiser was able to achieve transparent coding at bitrates as low as 21 bits/frame, though the computational complexity and memory requirements become an issue. This quantisation scheme was compared with scalar quantisers, the split vector quantiser, the multistage vector quantiser, and the single-frame GMM-based block quantiser, and was generally found to perform better in terms of spectral distortion, bitrate, and complexity.

The switched split vector quantiser was also evaluated as an LSF quantiser. Transparent coding was achieved at bitrates as low as 22 bits/frame, though the memory requirements of the two-part SSVQ were relatively high. It was determined that the three-part SSVQ, with transparent coding at 23 bits/frame, was well-balanced in terms of quantisation performance and complexity. Compared with other single-frame quantisers, the SSVQ achieved generally better spectral distortion performance. One aspect that the SSVQ excelled was the low computational complexity.

8.1.5 Chapter 6: LPC Parameter Quantisation in Wideband Speech Coding

This chapter began with the definition of wideband speech and described its advantages over toll-quality narrowband speech, such as improved naturalness and the ability to distinguish between fricatives, as well as provide better presence of the speaker, all of which can alleviate listener fatigue. We have also shown through the visual inspection of LPC-based spectral envelopes that, due to the extra bandwidth, a higher order LPC analysis is required to capture most of the short-term correlation information in the speech. Following this, we have given a review of the state-of-the-art coding schemes for wideband speech as well as the industry standard coders such as the ITU-T G.722 (subband/ADPCM coder) and ITU-T G.722.2 (AMR-WB ACELP coder).

Since the focus of this chapter is primarily on spectral quantisation for wideband LPC-based speech coders such as CELP, we provided a review of quantisation schemes that have been reported in the wideband speech coding literature. We have also evaluated some of these schemes such as PDF-optimised scalar quantisers, the unconstrained vector quantiser, and the GMM-based block quantiser on the two competing LPC parameter representations: line spectral frequencies (LSFs) and immittance spectral pairs (ISPs). Our experimental results have shown that ISPs are superior to LSFs by 1 bit/frame in independent quantisation schemes, such as scalar quantisers; while LSFs are the superior representation in joint vector schemes, such as the vector quantiser and GMM-based block quantiser. Through the extrapolation of the operating distortion-rate curve of unconstrained vector quantisation, we also derived an informal lower bound of 35 bits/frame and 36 bits/frame, for the transparent coding of wideband LSFs and ISPs, respectively. We speculate that this may be due to the fact that the last ISP parameter, which is not really a 'frequency', is not correlated with the other ISPs and hence impacts on the memory advantage of block and vector quantisation schemes, which aim to minimise the unweighted MSE for each vector as a whole. Furthermore, because this parameter is a reflection coefficient, it does not possess the error localisation properties of LSFs, but rather propagates errors throughout the entire spectrum. Therefore, additional measures may need to be taken, such as independent quantisation of the last ISP, or use of a weighted distance measure, when vector quantising ISPs.

Finally, we presented and discussed the results of the switched split vector quantiser (SSVQ) and the multi-frame GMM-based block quantiser, for coding wideband LSF and ISF vectors. The SSVQ was able to achieve transparent coding at 43 bits/frame and 44 bits/frame, when using an unweighted mean-squared-error (MSE) on LSFs and ISFs, respectively. The spectral distortion performance of the SSVQ on LSFs was improved by using a weighted MSE that emphasised LSFs that were located near peaks in the power spectrum. The resulting scheme was transparent at 42 bits/frame. The multi-frame GMM-based block quantiser was able to achieve transparent coding at 37 bits/frame and 38 bits/frame with a moderate computational complexity for LSFs and ISFs, respectively. These two quantisation experiments again confirm our finding that LSFs are superior to ISFs by about 1 bit/frame in joint vector quantisation schemes.

8.1.6 Chapter 7: MFCC Quantisation in Distributed Speech Recognition

In this chapter, we provided a brief review of automatic speech recognition with particular emphasis on the speech features such as the Mel frequency-warped cepstral coefficients. Following this, we reviewed the literature that investigated various modes of client/server-based speech recognition system, such as network speech recognitions and distributed speech recognition. The experimental database used for evaluating the performance of our MFCC quantisation schemes as well as the parameters for the recognition task were described in detail. Next, we presented our results on MFCC quantisation in a DSR framework using multi-frame GMM-based block quantisers and compared its performance against the memoryless GMM-based block quantiser, the non-uniform scalar quantiser, and the unconstrained vector quantiser. The multi-frame GMM-based block quantiser achieved better recognition at lower bitrates, exhibiting negligible degradation of 1% (WER of 2.5%) in recognition performance over the baseline system at 800 bps and 5% (WER of 7%) at 300 bps. Unlike vector quantisation schemes, the multi-frame GMM-based block quantiser is scalable in bitrate and has a complexity that is independent of bitrate. The performance of the multi-frame GMM-based block quantiser in the presence of noise was also evaluated. It was found that the recognition performance of relatively high SNRs was influenced mostly by the quantisation scheme. However, at low SNRs, the effect of quantisation efficiency diminishes and recognition performance is dependent on the noise robustness of the underlying features.

8.2 Suggestions for Future Research

This dissertation has examined block and vector quantisation schemes that are efficient, in terms of rate-distortion and computational requirements. Improvements in the rate-distortion efficiency have been derived from compensating the suboptimalities of each quantisation scheme, whether it be through accurate estimation of the source via parametric modelling (in the case of the GMM-based block quantiser), or exploitation of dependencies before applying constrained quantisation (in the case of the switched split vector quantiser). We have evaluated these schemes in four different applications, where efficient quantisation is required. In order to continue this line of research, this section lists some possible directions for further investigations.

In Chapter 2, two modern transform coding paradigms were discussed. They can be classified as either hard or soft clustering. The hard clustering paradigm takes the form of adaptive transform coders, where the vector space is partitioned into disjoint regions and a local transform is designed. Archer and Leen [13] developed the optimal adaptive transform coder, where the partitioning of the vector space, the local transform design, and quantiser design are performed jointly, in order to minimise distortion. The soft clustering paradigm involves modelling the source of the vectors using a mixture of individual and overlapping Gaussian sources, and designing optimal block quantisers for each source. It would be interesting to compare and contrast these two transform coding paradigms.

In Chapter 3, we identified the sources of suboptimality in the split vector quantiser and proposed the switched split vector quantiser, which compensates for the losses in the memory and shape advantages, by using a switch vector quantiser. As we have discussed in this chapter, the switched vector quantiser aims to exploit global dependencies in the vector space initially, which would otherwise have been neglected by an initial vector split. A further step would be to exploit the dependencies within each local cluster of vectors to increase the efficiency of the local split vector quantiser. Therefore, one possible path that warrants further research is to use different sized splitting for each of the local split vector quantisers. This variable splitting algorithm should adapt to the unique statistics of each cluster in order to minimise distortion. Another possible path is to develop a method of decorrelating the subvectors within each local split vector quantiser. That is, we can improve the efficiency of the split vector quantiser by removing correlation between each of the subvectors. A transform that can perform this *partial decorrelation* was discovered during the course of this research and this could be applied to the SSVQ.

In Chapter 5, we evaluated various quantisation schemes for LPC parameter coding used in narrowband speech coding. The training and test speech were assumed to be clean and noise-free. However, in a real-life scenario, the negative effects of additive background noise (such as babble and car noises) on the output quality of a speech coder cannot be neglected. In addition to this, the transmission channel was assumed to be perfect and lossless. However, this is not always possible in practice. Therefore, it is necessary to further investigate the effects of background noise and frame erasure conditions on the spectral distortion performance for each of the quantisation schemes considered in this dissertation.

In Chapter 6, we compared the relative performance of line spectral frequencies (LSFs) and immittance spectral pairs (ISPs) in quantisation experiments for wideband speech. We showed that ISPs outperformed LSFs in independent scalar quantisation while the trend was reversed for joint block and vector quantisation schemes. The unique nature of the last ISP warrants further investigation into how joint vector quantisation schemes can be applied in an optimal way. Similar to the case in MFCC quantisation, where the logarithmic energy coefficient, $\log E$, is quantised separately, separate quantisation of the last ISP may be required, which can be handled in a $(15, 1)$ split vector quantisation scheme. A weighted Euclidean distance measure needs to be developed for this type of quantisation scheme that takes into account the nature of the last ISP and how deviations affect the reconstructed power spectrum. In this chapter, we have also derived an informal lower bound on the number of bits required to transparently code LSFs and ISPs, by extrapolating the operating distortion-rate curve of the vector quantiser. We pointed out, however, that issues with 'over-training' were present that affected the tightness of this bound. Further work in determining a lower bound would involve undertaking the process outlined by Hedelin and Skoglund [66], where a GMM with bounded support is derived from training data, and used to generate artificial vectors. These vectors can then be used to train the vector quantiser. Lastly, as we have pointed out earlier, the effects of background noise and frame erasure conditions need to be investigated as well.

In Chapter 7, we evaluated the recognition performance of the various quantisation schemes in the task of quantising Mel frequency-warped cepstral coefficients (MFCCs). The distance measure that was used to design and search the quantiser codebook was mean-squared-error that is weighted with a fixed lifter window. It would be interesting to derive a weighted distance measure, that incorporates both fixed and dynamic weights, to emphasise parts of the speech that may be beneficial for recognition. Also, evaluating these quantisation schemes on feature sets that have been determined to be more robust to noise deserves further attention. The Aurora-2 connected-digits recognition task has a relatively small vocabulary and thus is not a very complex recognition task. It is proposed that the DSR evaluation be extended to databases with a larger vocabulary, such as the Aurora-3 and the DARPA Resource Management (RM) databases.

Bibliography

[1] "IEEE transactions, journals, and letters: information for authors", IEEE Periodicals, Transactions/Journals Department, 2000, pp. 4–5.

[2] "3rd generation partnership project; Technical specification group services and system aspects; Mandatory speech codec speech processing functions; Adaptive multi-rate (AMR) speech codec; Transcoding functions (Release 5)", Technical Specification TS 26.090, 3rd Generation Partnership Project (3GPP), June 2002.

[3] "3rd generation partnership project; Technical specification group services and system aspects; speech codec speech processing functions; AMR wideband speech codec; Transcoding functions (Release 5)", Technical Specification TS 26.190, 3rd Generation Partnership Project (3GPP), Dec 2001.

[4] "3rd generation partnership project; Technical specification group services and system aspects; speech codec speech processing functions; AMR wideband speech codec; Transcoding functions (Release 5)", Technical Specification TS 26.190, 3rd Generation Partnership Project (3GPP), Dec 2001.

[5] J.-P. Adoul and R. Lefebvre, "Wideband speech coding", in *Speech Coding and Synthesis*, W.B. Kleijn and K.K. Paliwal, Ed. Amsterdam: Elsevier, 1995, pp. 289–309.

[6] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies, "Image coding using vector quantization in the wavelet domain", in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, 1990, pp. 2297–2300.

[7] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies, "Image coding using wavelet transform", *IEEE Trans. Image Processing*, vol. 1, no. 2, pp. 205–220, Apr. 1992.

[8] M. Antonini, T. Gaidon, P. Mathieu, and M. Barlaud, "Wavelet transform and image coding", in *Wavelets in Image Communication*, M. Barlaud, Ed. Amsterdam: Elsevier, 1994, pp. 65–188.

[9] C. Archer and T.K. Leen, "Optimal dimension reduction and transform coding with mixture principal components", in *Proceedings of International Joint Conference on Neural Networks*, July 1999.

[10] C. Archer and T.K. Leen, "From mixtures of mixtures to adaptive transform coding", in *Proceedings of International Joint Conference on Neural Networks*, July 1999, pp. 925–931.

[11] C. Archer and T.K. Leen, "Adaptive transform coding as constrained vector quantization", in *Proceedings of the IEEE Workshop*, Dec. 2000.

[12] C. Archer and T.K. Leen, "The coding-optimal transform", in *Proceedings of the Data Compression Conference*, IEEE Computer Society Press, 2001.

[13] C. Archer and T.K. Leen, "A generalized Lloyd-type algorithm for adaptive transform coder design", *IEEE Trans. Signal Processing*, vol. 52, no. 1, pp. 255–264, Jan. 2004.

[14] B.S. Atal and S.L. Hanauer, "Speech analysis and synthesis by linear prediction of the speech wave", *J. Acoust. Soc. Amer.*, vol. 50, pp. 637–655, Aug. 1971.

[15] B.S. Atal and M.R. Schroeder, "Predictive coding of speech signals and subjective error criteria", *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-27, no. 3, pp. 247–254, Jun. 1979

[16] B.S. Atal and J.R. Remde, "A new model of LPC excitation for producing natural-sounding speech at low bit rates", in *Proc. IEEE. Int. Conf. Acoust., Speech, Signal Processing*, May 1982, pp. 614–617.

[17] B.S. Atal, R.V. Cox, and P. Kroon, "Spectral quantization and interpolation for CELP coders", in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, Glasgow, Scotland, May 1989, pp. 69–72.

[18] A. Bernard and A. Alwan, "Low-bitrate distributed speech recognition for packet-based and wireless communication", *IEEE. Trans. Speech Audio Processing*, vol. 10, no. 8, pp. 570–579, Nov. 2002.

[19] B. Bessette, R. Salami, R. Lefebvre, M. Jelfnek, J. Rotola-Pukkila, J. Vainio, H. Mikkola, and K. Jarvinen, “The adaptive multirate wideband speech codec (AMR-WB)”, *IEEE Trans. Speech Audio Processing*, vol. 10, no. 8, pp. 620–636, Nov. 2002.

[20] V. Bhaskaran and K. Konstantinides, *Image & Video Compression Standards*, 2nd Edition, Kluwer International Series, Boston, 1997.

[21] B. Bhattacharya, W.P. LeBlanc, S.A. Mahmoud, and V. Cuperman, “Tree searched multi-stage vector quantization of LPC parameters for 4 kb/s speech coding”, in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, 1992, pp. I-105–I-108.

[22] J.A. Bilmes, “A gentle tutorial of the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models”, Technical Report, University of Califorina, Berkeley, ICSI-TR-97-021, 1997.

[23] Y. Bistritz and S Pellerm, “Immittance spectral pairs (ISP) for speech encoding”, in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing*, 1993, pp. II-9–II-12.

[24] G. Biundo, S. Grassi, M. Ansorge, F. Pellandini and P.A. Farine, “Design techniques for spectral quantization in wideband speech coding”, in *Proc. of 3rd COST 276 Workshop on Information and Knowledge Management for Integrated Media Communication*, Budapest, Oct. 2002, pp. 114-119.

[25] P.J. Burt and E.H. Adelson, “The Laplacian pyramid as a compact image code”, *IEEE Trans. Commun.*, vol. 31(4), pp. 532–540, Apr. 1983.

[26] A. Buzo, A.H. Gray Jr., R.M. Gray, and J.D. Markel, “Speech coding based upon vector quantization”, *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 28, pp. 562–574, Oct. 1980.

[27] J.P. Campbell Jr., V.C. Welch, and T.E. Tremain, “An expandable error-protected 4800 bps CELP coder (U.S. federal standard 4800 bps voice coder)”, in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, Glasgow, U.K., 1989, pp. 735–738.

[28] W. Chen and C.H. Smith, “Adaptive coding of monochrome and color images”, *IEEE Trans. Commun.*, vol. COM-25(11), pp. 1285–1292, Nov. 1977.

[29] J.H. Chen and D. Wang, “Transform predictive coding of wideband speech signals”, in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, 1996, pp. 275–278.

[30] P. Combescure, J. Schnitzler, K. Fischer, R. Kirchherr, C. Lamblin, A. le Guyader, D. Massaloux, C. Quinquis, J Stegmann, P. Vary, "A 16, 24, 32 kbit/s wideband speech codec based on ATCELP", in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, 1999, pp. 5–8.

[31] R.V. Cox, "Speech Coding Standards", in *Speech Coding and Synthesis*, W.B. Kleijn and K.K. Paliwal, Ed. Amsterdam: Elsevier, 1995, pp. 49–78.

[32] R.E. Crochiere, S.A. Webber, and J.L. Flanagan, "Digital coding of speech in subbands", *Bell System Technical Journal*, vol. 55, pp. 1069–1085, Oct. 1976.

[33] A. Crossman, "A variable bit rate audio coder for videoconferencing", in *IEEE Workshop on Speech Coding for Telecommunications*, pp. 7–8, 1993.

[34] G. Davis and A. Nosratinia, "Wavelet-based image coding: an overview", *Applied and Computational Control, Signals, and Circuits*, vol. 1, no. 1, pp. 205–269, 1998.

[35] S.B. Davis and P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences", *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-28, no. 4, pp. 357–366, Aug. 1980.

[36] J. Dejener, "Digital speech compression: putting the GSM 06.10 RPE-LTP algorithm to work", 1994. Available: http://www.ddj.com/documents/s=1012/ddj9412b/

[37] A.P. Dempster, N.M. Laird and D.B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm", *J. Royal Stat. Soc.*, vol. 39, pp. 1–38, 1977.

[38] R.A. DeVore and B.J. Lucier, "Wavelets", Technical Report, University of South Carolina.

[39] V.V. Digalakis, L.G. Neumeyer and M. Perakakis, "Quantization of cepstral parameters for speech recognition over the world wide web", *IEEE J. Select. Areas Commun.*, vol. 17, no. 1, pp. 82–90, Jan 1999.

[40] R. Dony and S. Haykin, "Optimally adaptive transform coding", *IEEE Trans. Image Processing*, vol. 4, no. 10, pp. 1358–1370, 1995.

[41] H. Dudley, "The vocoder", *Bell Labs Rec.*, 19, pp. 122, 1939.

[42] E.R. Duni, A.D. Subramaniam, and B.D. Rao, "Improved quantization structures using generalised HMM modelling with application to wideband speech coding", in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, May 2004, pp. 161-164.

[43] M. Effros, P. Chou, and R.M. Gray, "Weighted universal image compression", *IEEE Trans. Image Processing*, vol. 8, no. 10, pp. 1317–1328, 1999.

[44] M. Effros, H. Feng, and K. Zeger, "Suboptimality of the Karhunen-Loève transform for transform coding", *IEEE Trans. Inform. Theory*, vol. 50, no. 8, pp. 1605–1619, Aug. 2004.

[45] T. Eriksson, J. Lindén, and J. Skoglund, "Exploiting interframe correlation in spectral quantization–a study of different memory VQ schemes", in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, 1996, pp. 765–768.

[46] D. Esteban and C. Galand, 'Application of quadrature mirror filters to split band voice coding schemes', in *Proc. IEEE Int. Conf. Acoustic, Speech, Signal Processing*, 1977, pp.191-195.

[47] "Speech processing, transmission and quality aspects (STQ); Distributed speech recognition; Front-end feature extraction algorithm; Compression algorithms", Tech. Rep. Standard ES 201 108 v1.1.3, European Telecommunications Standards Institute (ETSI), September 11 2003.

[48] S. Euler and J. Zinke, "The influence of speech coding algorithms on automatic speech recognition", in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, Adelaide, Australia, Apr. 1994, pp. I-621–I-624.

[49] A. Fingerhut, *U.S. Department of Defense LPC-10 Voice Coder*, Release 1.5, Washington University, Oct 1997. Available: http://www.arl.wustl.edu/ jaf/lpc/lpc10-1.5.tar.gz

[50] D. Gabor, "Theory of communication", *Proc. IEE*, 1936.

[51] A. Gallardo-Antolin, F. Diaz-de-Maria and F. Valverde-Albacete, "Recognition from GSM digital speech", in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, 1998, pp. 1443–1446.

[52] W.R. Gardner and B.D. Rao, "Theoretical analysis of the high-rate vector quantization of LPC parameters", *IEEE Trans. Speech Audio Processing*, vol. 3, pp. 367–381, Sept. 1995.

[53] A. Gersho, "Asymptotic optimal block quantization", *IEEE Trans. Inform. Theory*, vol. IT-25, pp. 373–380, July 1979.

[54] A. Gersho and B. Ramamurthi, "Image coding using vector quantization", in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, 1982, pp. 428–431.

[55] A. Gersho and R.M. Gray, *Vector Quantization and Signal Compression*, Massachusetts: Kluwer, 1992.

[56] V.K. Goyal, "Theoretical foundations of transform coding", *IEEE Signal Processing Mag.*, vol. 18, no. 5, Sept. 2001.

[57] V.K. Goyal, "Transform coding with backward adaptive updates", *IEEE Trans. Inform. Theory*, vol. 46, no. 4, pp. 1623–1633, July 2000,

[58] A. Graps, "An introduction to wavelets', *IEEE Computational Science and Engineering*, vol. 2, no. 2, pp. 50–61, 1995.

[59] R.M. Gray and D.L. Neuhoff, "Quantization", *IEEE Trans. Inform. Theory*, Vol. 44, No. 6, pp. 2325–2383, Oct. 1998.

[60] A. Gray and J. Markel, "Quantization and bit allocation in speech processing", *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-24, pp. 459–473, 1976.

[61] G. Guibé, H.T. How and L. Hanzo, "Speech spectral quantizers for wideband speech coding", *European Transactions on Telecommunications*, 12(6), pp. 535–545, 2001.

[62] F.S. Gurgen, S. Sagayama, and S. Furui, "Line spectrum frequency-based distance measures for speech recognition", in *Proc. Int. Conf. Spoken Language Processing*, Kobe, Japan, Nov. 1990, pp. 521–524.

[63] W. Fisher, V. Zue, J. Bernstein, and D. Pallet, "An acoustic-phoenetic data base", *J. Acoust. Soc. Am.*, vol. 81, Suppl. 1, 1987

[64] E. Harborg, J.E. Knudsen, A. Fuldseth and F.T. Johansen, "A real-time wideband CELP coder for a videophone application", in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, 1994, pp. 121–124.

[65] M.H. Hayes, *Statistical Digital Signal Processing and Modeling*, New York: Wiley, 1996, p. 40.

[66] P. Hedelin and J. Skoglund, "Vector quantization based on Gaussian mixture models", *IEEE Trans. Speech Audio Processing*, Vol. 8, No. 4, pp. 385–401, July 2000.

[67] H. Hermansky, "Perceptual linear predictive (PLP) analysis of speech", *J. Acoust. Soc. Am.*, 87(4), pp. 1738–1752, Apr. 1990.

[68] M.L. Hilton, B.D. Jawerth and Ayan Sengupta, "Compressing still and moving images with wavelets', *Multimedia Systems*, vol. 2, no. 3, 1994.

[69] H.G. Hirsch, "The influence of speech coding on recognition performance in telecommunication networks", in *Proc. Int. Conf. Spoken Language Processing*, Denver, USA, Sept. 1998.

[70] H.G. Hirsch and D. Pearce, "The Aurora experimental framework for the performance evaluation of speech recognition systems under noisy conditions", *ISCA ITRW ASR2000*, Paris, France, Sept. 2000.

[71] H. Hotelling, "Analysis of a complex of statistical variables into principal components", *J. Educ. Psychology*, vol. 24, pp. 417–441, 498–520, 1933.

[72] J.J.Y. Huang and P.M. Schultheiss, "Block quantization of correlated Gaussian random variables", *IEEE Trans. Commun. Syst.*, vol. CS-11, pp. 289–296, Sept. 1963.

[73] X. Huang, A. Acero, and H. Hon, *Spoken Language Processing: A Guide to Theory, Algorithm, and System Development*, New Jersey: Prentice Hall, 2001.

[74] J.M. Huerta and R.M. Stern, "Speech recognition from GSM codec parameters", in *Proc. Int. Conf. Spoken Language Processing*, vol. 4, 1998, pp.1463–1466.

[75] F. Itakura, "Line spectrum representation of linear predictive coefficients of speech signals", *J. Acoust. Soc. Amer.*, vol. 57, p. S35, Apr. 1975.

[76] F. Itakura, "Minimum prediction residual principle applied to speech recognition", *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-23, no. 1, pp. 67–72, Feb. 1975.

[77] F. Itakura and S. Saito, "Speech analysis-synthesis based on the partial autocorrelation coefficient", *Proc. JSA*, pp. 199–200, 1969.

[78] A.K. Jain, "Image data compression: a review", *Proc. IEEE*, vol. 69(3), pp. 349–389, Mar. 1981.

[79] J.D. Johnston, "A filter family designed for use in quadrature mirror filters banks", in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, 1980, pp. 291-294.

[80] B.H. Juang and A.H. Gray, Jr., "Multiple stage vector quantisation for speech coding', in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, vol. 1, 1982, pp. 597-600.

[81] B.H. Juang, L.R. Rabiner and J.G. Wilpon, "On the use of bandpass liftering in speech recognition", *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 35, pp. 947–954, July 1987.

[82] H. Karhunen, "Uber lineare methoden in der wahrscheinlichkeitsrechnung", *Ann. Acad. Sci. Fenn.*, Ser. A.I. 34, Helsinki, 1947.

[83] H.K. Kim and R.V. Cox, "A bitstream-based front-end for wireless speech recognition on IS-136 communications system", *IEEE Trans. Speech Audio Processing*, vol. 9, no. 5, pp. 558–568, July 2001.

[84] H. Kiya, K. Nishikawa, and M. Sagawa, "Property of circular convolution for subband image coding", *IEICE Trans. Fundamentals*, vol. E75-A, no. 7, pp. 852–860, July 1992.

[85] I. Kiss, "A comparison of distributed and network speech recognition for mobile communication systems", in *Proc. Int. Conf. Spoken Language Processing*, 2000.

[86] I. Kiss and P. Kapanen, "Robust feature vector compression algorithm for distributed speech recognition", in *Proc. Eurospeech*, 1999, pp. 2183–2186.

[87] N.P. Koestoer, "Robust linear prediction analysis for speech coding", PhD dissertation, Griffith University, 2002.

[88] B. Kolman, *Introductory Linear Algebra with Applications*, New York: Macmillan, 1990, pp. 439–444.

[89] K.P. Kramer and M.V. Mathews, "A linear coding for transmitting a set of correlated signals", *IRE Trans. Inform. Theory* (Corresp), vol. IT-17, pp. 751–752, Nov. 1971.

[90] V. Krishnan, D.V. Anderson and K.K. Truong, "Optimal multistage vector quantization of LPC parameters over noisy channels", *IEEE Trans. Speech Audio Processing*, Vol. 12, No. 1, pp. 1–8, Jan. 2004.

[91] P. Kroon and W.B. Kleijn, "Linear-prediction based analysis-by-synthesis coding" in *Speech Coding and Synthesis*, W.B. Kleijn and K.K. Paliwal, Ed. Amsterdam: Elsevier, 1995, pp. 79–119.

[92] C. Laflamme, J.-P. Adoul, R. Salami, S. Morissette, and P. Mabilleau, "16 kbps wideband speech coding technique based on algebraic CELP", in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, 1991, pp. 13–16.

[93] W.P. LeBlanc, B. Bhattacharya, S.A. Mahmoud, and V. Cuperman, "Efficient search and design procedures for robust multi-stage VQ for LPC parameters for 4 kb/s speech coding", *IEEE Trans. Speech Audio Processing*, vol. 1, no. 4, pp. 373–385, Oct. 1993.

[94] R. Lefebvre, R. Salami, C. Laflamme, and J.-P. Adoul, "8 kbit/s coding of speech with 6 ms frame-length", in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, 1993, pp. II-612–II-615.

[95] R. Lefebvre, R. Salami, C. Laflamme, and J.-P. Adoul, "High quality coding of wideband audio signals using transform coded excitation (TCX)", in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, 1994, pp. I-193–I-196.

[96] R.G. Leonard, "A database for speaker-independent digit recognition", in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, 1984, vol. 3, pp. 328–331.

[97] J. Le Roux and C. Gueguen, "A fixed point computation of partial correlation coefficients", *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 27, no. 3, pp. 257–259, June 1977.

[98] A.S. Lewis and G. Knowles, "Image compression using the 2-D wavelet transform", *IEEE Trans. Image Processing*, vol. 1, no. 2, pp. 244–250, Apr. 1992.

[99] B.T. Lilly and K.K. Paliwal, "Effect of speech coders on speech recognition performance", in *Proc. Int. Conf. Spoken Language Processing*, 1996, vol. 4, pp. 2344–2347.

[100] Y. Linde, A. Buzo, and R.M. Gray, "An algorithm for vector quantizer design", *IEEE Trans. Commun.*, vol. COM-28, no. 1, pp. 84–95, Jan. 1980.

[101] S.P. Lloyd, "Least square quantization in PCM", *IEEE Trans. Inform. Theory*, vol. IT-28, no. 2, pp. 129–137, Mar. 1982.

[102] M. Loéve, "Fontions aléatoires de seconde ordre", in P. Levý, *Processus Stochastiques et Mouvement Brownien*, Paris, France: Hermann, 1948.

[103] T.D. Lookabaugh and R.M. Gray, "High-resolution quantization theory and the vector quantizer advantage", *IEEE Trans. Inform. Theory*, vol. 35, no. 5, pp. 1020–1033, Sept 1989.

[104] J. MacQueen, "Some methods for classification and analysis of multivariate observations", *Proc. of the Fifth Berkeley Symposium on Math., Stat., and Prob.*, vol. 1, 1967, pp. 281–296.

[105] P.C. Mahalanobis, "On the generalized distance in statistics", in *Proc. Indian Nat. Inst. Sci.* (Calcutta), 1936, vol. 2, pp. 49–55.

[106] S.A. Martucci, "Signal extension and noncausal filtering for subband coding of images", *SPIE: Visual Communications and Image Processing*, vol. 1605, pp.137–148, 1991.

[107] J. Makhoul, "Linear prediction: a tutorial review", *Proc. IEEE*, vol. 63, no. 4, pp. 561–580, Apr. 1975.

[108] J. Makhoul, S. Roucos, and H. Gish, "Vector quantization in speech coding", *Proc. IEEE*, vol. 73, pp. 1551–1588, Nov. 1985.

[109] S. Mallat, "A theory of multiresolution signal decomposition: the wavelet representation", *IEEE Trans. Pattern Anal. Mach. Intel.*, vol. 11, no. 7, pp. 674–693, July 1989.

[110] H.S. Malvar and D.H. Staelin, "The LOT: transform coding without blocking effects", *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 37(4), pp. 553–559, Apr. 1989.

[111] J. Max, "Quantising for minimum distortion", *IRE Trans. Inform. Theory*, vol. IT-6, pp. 7–12, Mar. 1960.

[112] A. Mertins, *Signal Analysis: Wavelets, Filter Banks, Time-Frequency Transforms and Applications*, John Wiley and Sons, 1999.

[113] R. Meston, "GSM vocoders improve speech transmission", 2004. Available: http://www.eetasia.com/ARTICLES/2004NOV/B/2004NOV01_MSD_PD_TA.pdf

[114] W.B. Mikhael and V. Krishnan, "Energy-based split vector quantizer employing signal representation in multiple transform domains", *Digital Signal Processing*, vol. 11, no. 4, pp. 359-370, Oct. 2001.

[115] J.R. Movellan, "Tutorial on principal component analysis", Technical Report, University of California, San Diego, 2003. Available: http://mplab.ucsd.edu/tutorials/pdfs/PCA.pdf

[116] A.N. Netravali and J.O. Limb, "Picture coding: A review", *Proc. IEEE*, vol. 68, no. 3, pp. 366–406, Mar. 1980.

[117] P. Noll, "Digital audio coding for visual communications", *Proc. IEEE*, vol. 83, no. 6, pp. 925–943, June 1995.

[118] A. Ortega and M. Vetterli, "Adaptive scalar quantization without side information", *IEEE Trans. Image Proc.*, vol. 6, pp. 665–676, May 1997.

[119] A. Ortega and K. Ramchandran, "Rate-distortion methods for image and video compression', *IEEE Signal Processing Magazine*, vol. 15, no. 6, Nov 1998.

[120] M.D. Paez and T.H. Glisson, "Minimum mean-squared-error quantization in speech PCM and DPCM systems", *IEEE Trans. Commun.*, vol. COM-20, pp. 225–230, Apr. 1972.

[121] E. Paksoy, K. Srinivasan, and A. Gersho, "Variable rate speech coding with phonetic segmentation", in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, 1993, pp. 155–158.

[122] K.K. Paliwal and B.S. Atal, "Efficient vector quantisation of LPC parameters at 24 bits/frame", in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, 1991, pp. 661–664.

[123] K.K. Paliwal and B.S. Atal, "Efficient vector quantization of LPC parameters at 24 bits/frame", *IEEE Trans. Speech Audio Processing*, vol. 1, no. 1, pp. 3–14, Jan. 1993.

[124] K.K. Paliwal and W.B. Kleijn, "An introduction to speech coding" in *Speech Coding and Synthesis*, W.B. Kleijn and K.K. Paliwal, Ed. Amsterdam: Elsevier, 1995, pp. 1–47.

[125] K.K. Paliwal and W.B. Kleijn, "Quantization of LPC parameters" in *Speech Coding and Synthesis*, W.B. Kleijn and K.K. Paliwal, Ed. Amsterdam: Elsevier, 1995, pp. 443–466.

[126] K.K. Paliwal and S. So, "Low complexity GMM-based block quantisation of images using the discrete cosine transform", accepted by ICIP 2003.

[127] K.K. Paliwal and S. So, "Low complexity Gaussian mixture model-based block quantisation of images", in *Proc. Microelectronic Engineering Research Conference*, Brisbane, Australia, Nov. 2003.

[128] K.K. Paliwal and S. So, "A fractional bit encoding technique for the GMM-based block quantisation of images", *Digital Signal Processing*, vol. 15, pp. 255–275, May 2005.

[129] K.K. Paliwal and S. So, "Low complexity GMM-based block quantisation of images using the discrete cosine transform", *Signal Processing: Image Communication*, vol. 20, pp. 435–446, June 2005.

[130] K.K. Paliwal and S. So, "Multiple frame block quantisation of line spectral frequencies using Gaussian mixture models", in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, Montreal, Canada, 2004, pp. I-149–152.

[131] K.K. Paliwal and S. So, "Scalable distributed speech recognition using multi-frame GMM-based block quantization", in *Proc. Int. Conf. Spoken Language Processing*, Jeju, Korea, Oct. 2004.

[132] J. Pan and T.R. Fischer, "Vector quantization-lattice vector quantization of speech LPC coefficients", in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, vol. 1, 1994, pp. 513–516.

[133] J. Pan, "Two-stage vector quantization-pyramidal lattice vector quantization and application to speech LSP coding", in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, vol. 2, 1996, pp. 737–740.

[134] J.W. Paulus and J. Schnitzler, "16 kbit/s wideband speech coding based on unequal subbands", in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, 1996, pp. 255–258.

[135] L.C.W. Pols, "Spectral analysis and identification of Dutch vowels in monosyllabic words", Doctoral dissertation, Free University, Amsterdam, The Netherlands, 1966.

[136] J.G. Proakis and D.G. Manolakis, *Digital Signal Processing: Principles, Algorithms, and Applications*, 3rd Edition, New Jersey: Prentice-Hall, 1996.

[137] S. Quackenbush, "A 7 kHz bandwidth, 32 kbps speech coder for ISDN", in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, 1991, pp. 1–4.

[138] L.R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition", *Proc. IEEE*, vol. 77, no. 2, pp. 257–286.

[139] L.R. Rabiner and B.H. Juang, *Fundamentals of Speech Recognition*, New Jersey: Prentice Hall, 1993.

[140] B. Raj, J. Migdal and R. Singh, "Distributed speech recognition with codec parameters", in *Proc. ASRU*, Trento, Italy, Dec. 2001.

[141] B. Ramamurthi and A. Gersho, "Classified vector quantization of images", *IEEE Trans. Commun.*, vol. COM-34, pp. 1105–1115, Nov. 1986.

[142] T.A. Ramstad, S.O. Aase and J.H. Husøy, *Subband Compression of Images: Principles and Examples*, vol. 6, Amsterdam: Elsevier Science B.V., 1995.

[143] K.R. Rao and P. Yip, *Discrete Cosine Transform: Algorithms, Advantages, and Applications*, Academic Press, 1990.

[144] G.N. Ramaswamy and P.S. Gopalakrishnan, "Compression of acoustic features for speech recognition in network environments", in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, 1998, pp. 977–980.

[145] H.C. Reeve, III, and J.S. Lim, "Reduction of blocking effect in image coding", in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, 1983, pp. 1212–1215.

[146] D.A. Reynolds and R.C. Rose, "Robust text-independent speaker identification using Gaussian mixture models", *IEEE Trans. Speech Audio Processing*, vol. 3, no. 1, pp. 72–83, Jan. 1995.

[147] E.A. Riskin, "Optimal bit allocation via the generalized BFOS algorithm", *IEEE Trans. Inform. Theory*, 37(2), pp. 400–402, 1991.

[148] G. Roy and P Kabal, "Wideband CELP speech coding at 16 kbits/sec", in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, 1991, pp. 17–20.

[149] M.J. Sabin and R.M. Gray, "Product code vector quantizers for waveform and voice coding", *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-32, no. 3, pp. 474–488, June 1984.

[150] A. Said and W.A. Pearlman, 'A new fast and efficient image codec based on set partitioning in hierarchical trees', *IEEE Trans. Circuits and Systems for Video Technology*, vol. 6, June 1996.

[151] C. Sanderson, "Automatic person verification using speech and face information", PhD dissertation, Griffith University, 2002.

[152] K. Sayood, *Introduction to Data Compression*, San Francisco: Morgan Kaufmann Publishers, 1996.

[153] M.R. Schroeder and B.S. Atal, "Code-excited linear prediction (CELP): high quality speech at very low bit rates", in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, 1985, pp. 937–940.

[154] A. Segall, "Bit allocation and encoding for vector sources", *IEEE Trans. Inform. Theory*, vol. IT-22, no. 2, pp. 162–169, Mar. 1976.

[155] C.E. Shannon, "A mathematical theory of communication", *Bell Sys. Tech. J*, vol. 27, pp. 379-423, 625-656, 1948.

[156] C.E. Shannon, "Coding theorems for a discrete source with a fidelity criterion", *IRE National Convention Record*, part 4, pp. 142-163, 1959.

[157] B.J. Shannon and K.K. Paliwal, "A comparative study of filter bank spacking for speech recognition", in *Proc. Microelectronic Engineering Research Conference*, Brisbane, Australia, Nov. 2003.

[158] J.M. Shapiro, "Embedded image coding using zerotrees of wavelet coefficients", *IEEE Trans. Signal Processing*, vol. 41, no. 12, pp. 3445–3462, Dec. 1993.

[159] J.M. Shapiro, "An embedded wavelet hierarchical image coder', *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, vol. 4, March 1992.

[160] Y. Shin, S. Kang, T.R. Fischer, C. Son, and Y. Lee, "Low-complexity predictive trellis coded quantization of wideband speech LSF parameters", in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, 2004, pp. 145-148.

[161] J. Shlens, "A tutorial on principal component analysis", Technical Report, University of California, San Diego, 2003. Available: http://www.snl.salk.edu/ shlens/pub/notes/pca.pdf

[162] Y. Shoham, "Low-delay code-excited linear predictive coding of wideband speech at 32 kbps", in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, 1991, pp. 9–12.

[163] U. Sinervo, J. Nurminen, A. Heikkinen, and J. Saarinen, "Evaluation of split and multistage techniques in LSF quantization", in *Proc. Norsig 2001*, Trondheim, Norway, Oct. 2001, pp. 18–22.

[164] M.J.T. Smith and S.L. Eddins, "Analysis/synthesis techniques for subband image coding", *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 38, no. 8, pp. 1446–1456, Aug. 1990.

[165] S. So and K.K. Paliwal, "Efficient block coding of images using Gaussian mixture models", in *Proc. Fourth Australasian Workshop on Signal Processing and Applications 2002*, Brisbane, Australia, Sept. 2002, pp. 71–74.

[166] S. So and K.K. Paliwal, "Efficient vector quantisation of line spectral frequencies using the switched split vector quantiser", in *Proc. Int. Conf. Spoken Language Processing*, Jeju, Korea, Oct. 2004.

[167] S. So and K.K. Paliwal, "Multi-frame GMM-based block quantisation of line spectral frequencies", to appear in *Speech Commun.*, 2005.

[168] S. So and K.K. Paliwal, "Multi-frame GMM-based block quantisation of line spectral frequencies for wideband speech coding", in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, vol. I, Philadelphia, USA, 2005, pp. 121–124.

[169] S. So and K.K. Paliwal, “Efficient product code vector quantisation using the switched split vector quantiser”, submitted to *Digital Signal Processing*, Nov. 2004.

[170] S. So and K.K. Paliwal, “A comparative study of LPC parameter representations and quantisation schemes in wideband speech coding”, submitted to *Digital Signal Processing*, May 2005.

[171] S. So and K.K. Paliwal, “Comparison of LPC parameter representations for wideband speech coding”, to be submitted to *IEEE Signal Processing Lett.*, 2005.

[172] S. So and K.K. Paliwal, “Switched split vector quantisation of line spectral frequencies for wideband speech coding”, to appear in *Proc. European Conf. Speech Communication and Technology* (Eurospeech), Lisbon, Portugal, Sept 2005.

[173] S. So and K.K. Paliwal, “A comparison of LSF and ISP representations for wideband LPC parameter coding using the switched split vector quantiser”, to appear in *Proc. IEEE Int. Symp. Signal Processing and Applications* (ISSPA), Sydney, Australia, Aug 2005.

[174] S. So and K.K. Paliwal, “Scalable distributed speech recognition using Gaussian mixture model-based block quantisation”, submitted to *Speech Commun.*, 2005.

[175] F.K. Soong and B.H. Juang, “Line spectrum pair (LSP) and speech data compression”, in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, San Diego, California, Mar 1984, pp. 37–40.

[176] F.K. Soong and B.H. Juang, “Optimal quantization of LSP parameters”, in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, New York, pp. 394–397, Apr. 1988.

[177] T. Sporer, K. Brandenburg and B. Edler, “The use of multirate filterbanks for coding of high quality digital audio”, in *Proc. EUSIPCO*, vol. 1, 1992, pp. 211–214.

[178] N. Srinivasamurthy, A. Ortega and S. Narayanan, “Efficient scalable encoding for distributed speech recognition”, submitted to *IEEE Trans. Speech and Audio Processing*, 2003. Available: http://biron.usc.edu/~snaveen/papers/Scalable_DSR.pdf

[179] S.S. Stevens and J. Volkman, “The relation of pitch to frequency”, *Journal of Psychology*, 53, pp. 329, 1940

[180] B. Strope and A. Alwan, "A model of dynamic auditory perception and its application to robust word recognition", *IEEE Trans. Speech Audio Processing*, vol. 5, no. 2, pp. 451–464, Sept. 1997.

[181] J.K. Su and R.M. Mersereau, "Coding using Gaussian mixture and generalized Gaussian models", in *Proc. IEEE Int. Conf. Image Processing*, Lausanne, Switzerland, 1996, pp. 217–220.

[182] A.D. Subramaniam and B.D. Rao, "PDF optimized parametric vector quantization with applications to speech coding", *34th Asilomar Conference on Signals, Systems and Computers*, Pacific Grove, CA, Nov. 2000.

[183] A.D. Subramaniam and B.D. Rao, "PDF optimized parametric vector quantization of speech line spectral frequencies", *IEEE Trans. Speech Audio Processing*, vol. 11, no. 2, pp. 130–142, Mar. 2003.

[184] A.D. Subramaniam, "Gaussian mixture models in compression and communication", PhD dissertation, University of California, San Diego, CA, 2003.

[185] N. Sugamura and F. Itakura, "Speech analysis and synthesis methods developed at ECL in NTT–from LPC to LSP–", *Speech Commun.*, vol. 5, pp. 199–215, Jun. 1986.

[186] R.J. Tocci, *Digital Systems: Principles and Applications*, 6th Edition, Prentice Hall, New Jersey, pp. 6–9, 1995.

[187] C. Tsao and R.M. Gray, "Matrix quantizer design for LPC speech using the generalized Lloyd algorithm", *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 33, pp. 537–545, June 1985.

[188] A. Tucker, *A Unified Introduction to Linear Algebra: Models, Methods, and Theory*, New York: Maxwell-Macmillan, 1989, pp. 502–504.

[189] J. Turunen and D. Vlaj, "A study of speech coding parameters in speech recognition", in *Proc. Eurospeech*, 2001, pp. 2363–2366.

[190] A. Ubale and A. Gersho, "A multi-band CELP wideband speech coder", in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, 1994, pp. 1367-1370.

[191] C. Valens, "A really friendly guide to wavelets", Technical Report, 1999. Available: http://perso.wanadoo.fr/polyvalens/clemens/download/arfgtw.pdf

[192] M. Vetterli, “Multi-dimensional subband coding: some theory and algorithms”, *Signal Processing*, vol. 6, pp. 97-112, Apr. 1984.

[193] M. Vetterli and J. Kovacevic, *Wavelets and Subband Coding*, Prentice-Hall, New Jersey, 1995.

[194] M. Vetterli, “On Fourier and wavelets: representation, approximation, and compression”, presented at Wavelet and Multifractal Analysis 2004, Cargese, Corsica, July 2004. Available: http://www.inrialpes.fr/is2/people/pgoncalv/WAMA2004/lectures/Vetterli-lecture.pdf.

[195] R. Viswanathan and J. Makhoul, “Quantization properties of transmission parameters in linear predictive systems”, *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-23, pp. 309–321, 1975.

[196] G.K. Wallace, “The JPEG still picture compression standard”, *Communications of the ACM*, vol. 34, no. 4, pp. 30-44, Apr. 1991.

[197] S. Wang and A. Gersho, “Phonetically-based vector excitation coding of speech at 3.6 kbit/s”, in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, Glasgow, May, 1989, pp. I-349–352.

[198] P.A. Wintz, “Transform picture coding”, *Proc. IEEE*, vol. 60(7), pp. 809–820, July 1972.

[199] R.C. Wood, “On optimum quantization”, *IEEE Trans. Inform. Theory*, vol. IT-15, no. 2, pp. 248–252, Mar. 1969.

[200] J.W. Woods and S.D. O’Neil, “Sub-band coding of images”, in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, 1986, pp. 1005–1008.

[201] J.W. Woods and S.D. O’Neil, “Subband coding of images”, *IEEE Trans. Acoust., Speech, and Signal Processing*, vol. ASSP-34(5), pp. 1278-1288, Oct. 1986.

[202] T. Wuppermann and F. de Bont, “Feasibility study of 32 kb/s wideband speech and music coding with a low-delay filterbank”, in *IEEE Workshop on Speech Coding for Telecommunications*, pp. 11–12, 1993.

[203] S. Young, G. Evermann, T. Hain, D. Kershaw, G. Moore, J. Odell, D. Ollason, D. Povey, V. Valtchev, and P. Woodland, *The HTK Book (for HTK Version 3.2.1)*, Cambridge University Engineering Department, 2002.

[204] Y. Zhang and C.J.S. deSilva, "An isolated word recognizer using the EM algorithm for vector quantization", *IREECON 1991*, Sydney, Australia, pp. 289–292.

[205] J. Zhou, Y. Shoham and A. Akansu, "Simple fast vector quantization of the line spectral frequencies", in *Proc. Int. Conf. Spoken Language Processing*, Vol. 2, 1996, pp. 945–948.

[206] Q. Zhu and A. Alwan, "An efficient and scalable 2D DCT-based feature coding scheme for remote speech recognition", in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, vol. 1, Aug 2001, pp. 113-116.

[207] Q. Zhu and A. Alwan, "On the use of variable frame rate analysis in speech recognition", in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, 2000, vol. 3, pp. 1783–1786.