Development of Novel Image Compression Algorithms for Portable Multimedia Applications

Ranjan Kumar Senapati
[Roll No. 508EC107]

Department of Electronics and Communication Engineering
National Institute of Technology
Rourkela-769 008, Odisha, India
December 2012

Development of Novel Image Compression Algorithms for Portable Multimedia Applications

Thesis submitted in partial fulfillment
of the requirements for the degree of

Doctor of Philosophy

in

Electronics and Communication Engineering

by

Ranjan Kumar Senapati
(Roll: 508EC107)

under the guidance of

Dr. Umesh Chandra Pati

Prof. Kamala Kanta Mahapatra

Department of Electronics and Communiation Engineering
National Institute of Technology Rourkela
Rourkela-769 008, Odisha, India

Dedicated
to
my family

Department of Electronics and Communication Engg.
National Institute of Technology Rourkela
Rourkela-769 008, Odisha, India.

September 26, 2013

Certificate

This is to certify that the thesis titled ***Development of novel image compression algorithms for portable multimedia applications*** by ***Ranjan Kumar Senapati*** is a record of an original research work carried out under our supervision and guidance in partial fulfillment of the requirements for the award of the degree of Doctor of Philosophy in ***Electronics and Communication Engineering*** during the session 2011-2012. We believe that the thesis fulfills part of the requirements for the award of degree of Doctor of Philosophy. The results embodied in the thesis have not been submitted for award of any degree.

Dr. Umesh C. Pati	**Prof. Kamala K. Mahapatra**
Associate Professor	Professor
Dept. of Electronics & Comm. Engg.	Dept. of Electronics & Comm. Engg.
National Institute of Technology, Rourkela	National Institute of Technology, Rourkela
Odisha, India - 769 008	Odisha, India - 769 008

Acknowledgement

"The will of God will never take you where Grace of God will not protect you."
Thank you God for showing me the path...

I owe deep gratitude to the ones who have contributed greatly in completion of this thesis.

Foremost, I would like to express my sincere gratitude to my supervisors Dr. Umesh Chandra Pati and Prof. Kamala Kanta Mahapatra for providing me with a platform to work on challenging and mature area of Image Compression. Their profound insights and attention to details have been true inspirations to my research.

I am very much indebted to Prof. Sukadev Meher, Prof. Sarat Kumar Patra, Prof. Deviprasad Priyabrata Acharya, Prof. Banshidhar Majhi and Prof. K. Barada Mohanty for providing insightful comments at different stages of thesis that were indeed thought provoking.

My special thanks go to Prof. Upendra Kumar Sahoo and Prof. Ajit Kumar Sahoo for contributing towards enhancing the quality of the work in shaping this thesis.

I would like to thank Karuppanan, Dandpat sir, Bhasker, Ayaskanta sir, Aditya, Madhusmita, Nataraj, Deepak for their encouragement and understanding. Their help can never be penned with words. I acknowledge all staff, research scholars and juniors of ECE department, NIT Rourkela for helping me.

I specially thank the management and principal of GMR Institute of Technology-Rajam for granting study leave during my Ph.D program.

Most importantly, none of this would have been possible without the love and patience of my family. My parents, *Sri Purandara Senapati* and *Smt. Kunti lata Senapati*, my wife *Subhalaxmi* and my little son *Pradnesh* to whom this dissertation is dedicated to, have been a constant source of love, concern, support and strength all these years. I would like to express my heartfelt gratitude to them.

Ranjan Kumar Senapati

Abstract

Portable multimedia devices such as digital camera, mobile devices, personal digital assistants (PDAs), etc. have limited memory, battery life and processing power. Real time processing and transmission using these devices requires image compression algorithms that can compress efficiently with reduced complexity. Due to limited resources, it is not always possible to implement the best algorithms inside these devices. In uncompressed form, both raw and image data occupy an unreasonably large space. However, both raw and image data have a significant amount of statistical and visual redundancy. Consequently, the used storage space can be efficiently reduced by compression. In this thesis, some novel low complexity and embedded image compression algorithms are developed especially suitable for low bit rate image compression using these devices.

Despite the rapid progress in the Internet and multimedia technology, demand for data storage and data transmission bandwidth continues to outstrip the capabilities of available technology. The browsing of images over Internet from the image data sets using these devices requires fast encoding and decoding speed with better rate-distortion performance. With progressive picture build-up of the wavelet based coded images, the recent multimedia applications demand good quality images at the earlier stages of transmission. This is particularly important if the image is browsed over wireless lines where limited channel capacity, storage and computation are the deciding parameters. Unfortunately, the performance of JPEG codec degrades at low bit rates because of underlying block based DCT transforms. Although wavelet based codecs provide substantial improvements in progressive picture quality at lower bit rates, these coders do not fully exploit the coding performance at lower bit rates. It is evident from the statistics of transformed images that the number of significant coefficients having magnitude higher than earlier thresholds are very few. These wavelet based codecs code zero to each insignificant subband as it moves from coarsest to finest subbands. It is also demonstrated that there could be six to seven bit plane passes where wavelet coders encode many zeros as many subbands are likely to be insignificant with respect to early thresholds. Bits indicating insignificance of a coefficient or subband are required, but they don't code information that reduces distortion of the reconstructed image. This leads to reduction of zero distortion for an increase in non zero bit-rate.

Another problem associated with wavelet based coders such as Set partitioning in hierarchical trees (SPIHT), Set partitioning embedded block (SPECK), Wavelet

block-tree coding (WBTC) is because of the use of auxiliary lists. The size of list data structures increase exponentially as more and more elements are added, removed or moved in each bitplane pass. This increases the dynamic memory requirement of the codec, which is a less efficient feature for hardware implementations. Later, many listless variants of SPIHT and SPECK, e.g. No list SPIHT (NLS) and Listless SPECK (LSK) respectively are developed. However, these algorithms have similar rate distortion performances, like the list based coders. An improved LSK (ILSK) algorithm proposed in this dissertation that improves the low bit rate performance of LSK by encoding much lesser number of symbols (i.e. zeros) to several insignificant subbands. Further, the ILSK is combined with a block based transform known as discrete Tchebichef transform (DTT). The proposed new coder is named as Hierarchical listless DTT (HLDTT). DTT is chosen over DCT because of it's similar energy compaction property like discrete cosine transform (DCT). It is demonstrated that the decoded image quality using HLDTT has better visual performance (i.e., Mean Structural Similarity) than the images decoded using DCT based embedded coders in most of the bit rates.

The ILSK algorithm is also combined with Lift based wavelet transform to show the superiority over JPEG2000 at lower rates in terms of peak signal-to-noise ratio (PSNR). A full-scalable and random access decodable listless algorithm is also developed which is based on lift based ILSK. The proposed algorithm named as scalable listless embedded block partitioning (S-LEBP) generates bit stream that offer increasing signal-to-noise ratio and spatial resolution. These are very useful features for transmission of images in a heterogeneous network that optimally service each user according to available bandwidth and computing needs. Random access decoding is a very useful feature for extracting/manipulating certain area of an image with minimal decoding work. The idea used in ILSK is also extended to encode and decode color images. The proposed algorithm for coding color images is named as Color listless embedded block partitioning (CLEBP) algorithm. The coding efficiency of CLEBP is compared with Color SPIHT (CSPIHT) and color variant of WBTC algorithm. From the simulation results, it is shown that CLEBP exhibits a significant PSNR performance improvement over the later two algorithms on various types of images.

Although many modifications to NLS and LSK have been made, the listless modification to WBTC algorithm has not been reported in the literature. Therefore, a listless variant of WBTC (named as LBTC) algorithm is proposed. LBTC not only reduces the memory requirement by 88-89% but also increases the encoding and decoding speed, while preserving the rate-distortion performance at the same time.

Further, the combination of DCT with LBTC (named as DCT_LBT) and DTT with LBTC (named as Hierarchical listless DTT, HLBT_DTT) are compared with some state-of-the-art DCT based embedded coders. It is also shown that the proposed DCT_LBT and HLBT_DTT show significant PSNR improvements over almost all the embedded coders in most of the bit rates.

In some multimedia applications e.g., digital camera, camcorders etc., the images always need to have a fixed pre-determined high quality. The extra effort required for quality scalability is wasted. Therefore, non-embedded algorithms are best suited for these applications. The proposed algorithms can be made non-embedded by encoding a fixed set of bit planes at a time. Instead, a sparse orthogonal transform matrix is proposed, which can be integrated in a JEPG baseline coder. The proposed matrix promises a substantial reduction in hardware complexity with a marginal loss of image quality on a considerable range of bit rates than block based DCT or Integer DCT.

Contents

Certificate iii

Acknowledgement iv

List of Figures xii

List of Tables xvi

List of Abbreviations xix

1 Introduction **1**

1.1 Representation of Digital Images 1

1.1.1 Need of Image Compression 2

1.2 Image Compression Fundamentals 2

1.2.1 Measuring Image Information 4

1.2.2 Fidelity Criteria 5

1.2.3 Image Compression Model 6

1.2.4 JPEG Baseline Image Compression 7

1.2.5 JPEG2000 Image Coding Standard 10

1.2.6 Embedded Image Compression 11

1.3 Background and Scope 12

1.4 Motivation 16

1.5 Objective 17

1.6 Organization of the Dissertation 18

1.7 Conclusions 20

2 Compression Performance Assessment of Discrete Tchebichef Transform **21**

2.1 Introduction 22

2.2 Discrete Tchebichef Transform 23

2.3 Similar Properties between DTT and DCT 26

2.4 Application of DTT on JPEG baseline Standard 29

2.4.1 Simulation Results 30

2.4.1.1 Coding Performance for Grayscale Images 30
2.4.1.2 Coding Performance for Color Images 32
2.5 Application of DTT on SPIHT Embedded Coder 41
2.5.1 The Proposed DTT_SPIHT Embedded Coder 41
2.5.1.1 Rearrangement Algorithm of Transformed Coefficients 41
2.5.1.2 Human Visual System 42
2.5.1.3 SPIHT Algorithm 43
2.5.2 Simulation Results and Analysis 43
2.6 Development of 4 × 4 Zigzag Pruning DTT Algorithm 46
2.6.1 Proposed Zigzag Prune 4 × 4 DTT Algorithm 47
2.6.2 Complexity Analysis 48
2.6.3 Hardware Utilization 49
2.6.4 Results and Comparisons 50
2.7 Conclusions 51

3 Low Complexity Embedded Image Compression Algorithm Using Hierarchical Listless DTT 53

3.1 Introduction 53
3.2 The Proposed HLDTT Embedded Coder 55
3.2.1 Algorithm for Rearrangement of Transformed Coefficients . . . 56
3.2.2 Improved LSK Algorithm 56
3.2.3 The Pseudo code of ILSK Algorithm 59
3.2.3.1 Description of Functions and Parameters used in Pseudocode 60
3.2.4 Region of Interest Retrievability 62
3.2.4.1 Rate-Distortion optimization 65
3.2.4.2 Strengths of the proposed method compared to Maxshift method in JPEG2000 69
3.2.5 Memory Requirement 70
3.2.5.1 Dynamic Memory Comparison 71
3.2.6 Complexity Analysis 73
3.3 Comparison of DTT with ICT, DCT, IWT, Filter based DWT and Lift based DWT 74
3.4 Simulation Results and Analysis 77
3.4.1 Coding Performance 77
3.4.2 Statistical Analysis 86
3.4.3 Computational Complexity 93
3.4.4 Effect of Block Size 94

3.5 Conclusions 95

4 Reduced Memory Listless Scalable Embedded Image Compression Algorithms 97

4.1 Introduction 97
4.2 Listless Embedded Block Partitioning Algorithm for Gray Scale Images 99
    4.2.1 Comparison with LSK and NLS Algorithms 99
    4.2.2 Memory Allocation 99
        4.2.2.1 Dynamic Memory Comparison 100
4.3 Listless Embedded Block Partitioning Algorithm for Color Images . . 102
    4.3.1 Pseudocode of CLEBP Algorithm 104
    4.3.2 Complexity Analysis 105
4.4 Simulation Results and Analysis 106
    4.4.1 Coding Performance 107
        4.4.1.1 Coding Performance for Gray Scale Images 108
        4.4.1.2 Coding Performance for Color Images 110
        4.4.1.3 Coding Performances Between Gray scale and Color Images 111
        4.4.1.4 Progressive Transmission and Scalability Efficiency . 115
        4.4.1.5 Region of interest and Random access decoding . . . 118
    4.4.2 Effect of Block Size 120
        4.4.2.1 Impact of Coding Eficiency on Varying Block Sizes . 120
        4.4.2.2 Impact of Encoding and Decoding Times on Varying Block Sizes 123
    4.4.3 Computational Complexity 124
4.5 Conclusions 124

5 Listless Block-Tree Set Partitioning Algorithm for Very Low Bit Rate Embedded Image Compression 126

5.1 Introduction 126
5.2 The Proposed LBTC Algorithm 128
    5.2.1 Initialization 130
    5.2.2 The Pseudo-code of LBTC(2 × 2) Encoder Algorithm 132
5.3 The Proposed DCT_LBT Embedded Encoder 133
    5.3.1 Relation Between Transformed Coefficients 134
5.4 The proposed HLBT_DTT Embedded Coder 135
    5.4.1 Coefficient Scanning Order 135
5.5 Memory Allocation 136
    5.5.1 Dynamic Memory Requirement Comparison 137

5.6 Experimental Results and Performance Comparison 137
    5.6.1 Coding Efficiency of LBTC Embedded Wavelet Coder 137
    5.6.2 Coding Efficiency of DCT_LBT Embedded Coder 140
    5.6.3 Coding Efficiency of HLBT_DTT Embedded Coder 145
    5.6.4 Computational Complexity 151
5.7 Conclusions 153

6 A Low Complexity and Efficient Sparse 8×8 Orthogonal Transform Matrix for Image Compression 154

6.1 Introduction 154
6.2 Signed Discrete Cosine Transform 155
6.3 Proposed 8×8 Transform Matrix 157
6.4 Distributed Arithmetic based Algorithm for Fast Computation 158
6.5 Application to JPEG Color Image Compression 160
    6.5.1 Chroma Subsampling and Color Space Conversion 160
    6.5.2 Merge and Quantize 161
6.6 FPGA Synthesis Results of the Proposed Transform Matrix 161
6.7 Matlab Simulation Results and Discussion 163
6.8 Conclusion 165

7 Conclusions 169

7.1 Summary of Work Done 169
7.2 Contributions 171
7.3 Future Work 172

References 173

Dissemination 181

Appendix A 185

List of Figures

1.1 Alternative views of an image data 3

1.2 General image compression system 6

1.3 Baseline JPEG: (a) Compression, (b) Decompression 8

1.4 Parent-child relationship of wavelet coefficients of a 3-level wavelet decomposition pyramid. 11

2.1 The 8×8 basis images of 2D (a) DTT (b) DCT. 26

2.2 Plot of Tchebichef polynomials for $N = 20$ 27

2.3 The Energy Compaction Property on a 256×256 image using (a) DCT, and (b) DTT. 28

2.4 Implementation of FDTT on the 8×8 blocks of image data. 29

2.5 The Compression ratio vs scaling factor plots between DCT and DTT on a JPEG baseline codec (a) Lena, (b) Mandrill, and (c) Ruler images. 31

2.6 The rate-distortion comparison plots between DCT and DTT on a JPEG baseline codec (a) Fingerprint, and (b) Slope images. 33

2.7 The rate-distortion comparison plots between DCT and DTT on a JPEG baseline codec (a) Lena, and (d) Resolution chart images. . . 34

2.8 The rate-distortion comparison plots between DCT and DTT on a JPEG baseline codec (a) Ruler, and (b) Numbers images. 35

2.9 The rate-distortion comparison plots between DCT and DTT on a JPEG baseline codec (a) Barbara, and (b) Mandrill images. 36

2.10 The reconstructed Lena images at a scale factor of 5 using DCT and DTT 37

2.11 The reconstructed Ruler images at a scale factor of 5 using DCT and DTT 37

2.12 The reconstructed Mandrill images at a scale factor of 5 using DCT and DTT 38

2.13 The rate-distortion comparison plots between DCT and DTT on a JPEG baseline codec (a) f16 and (b) Lena color images. 38

2.14 Visual quality of the f16 decoded images between DCT and DTT on a JPEG baseline codec 39
2.15 Visual quality of the Lena decoded images between DCT and DTT on a JPEG baseline codec 40
2.16 Block diagram of HVS based DTT_SPIHT embedded image coder . . 41
2.17 Rearrangement algorithm of 8 × 8 transformed coefficients. 42
2.18 The reconstructed images of Lena using HVS based DTT_SPIHT at a bit-rate of 0.25 bpp 46

3.1 Block diagram of HLDTT embedded image coder 55
3.2 The one to one correspondence between $MV(k)$ values and coefficient values $X(k)$ 61
3.3 Procedure of finding ROI. 64
3.4 ROI retrievability on Barbara image (a) Before RD optimization, (b) After RD optimization 67
3.5 Formation of compressed bit stream using ROIs (a) Overall bit stream structure after ROI arrangement (b) Formation of quality layers for different codeuints (c) Final SNR scalable compressed bit stream of an ROI using quality layers. 69
3.6 (a) Convex hull of the distortion-rate pairs for the code block C_i, (b) R-D optimization on Barbara image 70
3.7 MSSIM vs. bit rate on (a) Lena (b) Barbara (c) Boat (d) Mandrill (e) Ruler and (f) 256 Level Test Pattern images. 85
3.8 Barbara and Mandrill images compressed at 0.5 bpp using DCT_SPIHT and HLDTT 89
3.9 Boat images decoded at a target bit rate of 0.05 bpp using DCT_SPIHT, HLDTT, DWT_SPIHT, JPEG 2000 and DWT_ILSK. 90
3.10 Ruler images decoded at a target bit rate of 1.0 bpp using DWT_SPIHT, JPEG 2000, DWT_ILSK, DCT_SPIHT, and HLDTT. 91
3.11 (a) Table and left side part of Barbara (b) Hair and face part of Lena (c) Facial portion of Zelda (d) Hair and warps part of Barbara images. 91

4.1 Sequential numbering pattern across correlated wavelet subbands in 4:2:0 color planes 103
4.2 Steps for partitioning a combined wavelet level 103
4.3 Rate-distortion performance comparison of Lena image on L-C planes 111
4.4 Rate-distortion performance comparison of Hall-Monitor image on L-C planes 112

4.5 Rate-distortion performance comparison of News image on L-C planes 113
4.6 Hall monitor image compressed using CLEBP at 1 bpp using replicate and symmetrical paddings 114
4.7 News image compressed using CLEBP at 0.5 bpp using replicate and symmetrical padding 114
4.8 Decoded Lena images compressed at bit rates of 0.01 using SPIHT, NLS, LSK and LEBP. 116
4.9 Decoded cameraman images using SPIHT, NLS, LSK and LEBP . . . 117
4.10 Resolution scalable Lena images decoded at a bit rate of 0.05 bpp by LEBP algorithm 118
4.11 Rearrangement of wavelet transformed coefficients for ROI/random access decoding where correlated coefficients across different scales are mapped into a localized region of the source image. 119
4.12 ROI of facial portion of Lena image (128 × 128 size) which is retrieved at a bit rate of (a)0.125 bpp, (b)0.25bpp, (c)0.5bpp, and (d)1.0bpp. . 119
4.13 Random access decoding of k^{th} block at 1.0 bpp 120
4.14 Proposed block diagram of DCT based embedded coder using various block sizes 121
4.15 Rate-distortion performance of DCT_LEBP algorithm on varying block sizes for (a)Lena and (b)Barbara images. 123
4.16 Comparison of (a) Encoding time(sec) and (b)Decoding time(sec) on varying block sizes of LEBP algorithm on Barbara image. 124

5.1 Two dimensional morton scan sequence of the transformed coefficients 129
5.2 Parent-child relationship of a block-tree in a three level pyramid . . . 130
5.3 Block diagram of proposed DCT_LBT embedded image coder 134
5.4 Rearrangement algorithm of transformed coefficients 135
5.5 Scanning order of hierarchical subbands 136
5.6 Rate distortion performance comparison of WBTC and LBTC 139
5.7 Rate distortion performance comparison of DCT_LBT on varying block sizes 142
5.8 Decoded Bike images compressed at 0.1 bpp 143
5.9 Decoded Woman images compressed at 0.05 bpp 147
5.10 Decoded Lena images compressed at 0.1 bpp 148
5.11 Decoded Ruler images compressed at 0.5 bpp 149

6.1 Hardware structure of the proposed matrix 160

6.2 Block diagram of proposed transform matrix on the 8 × 8 blocks of image data. 161
6.3 Rate-distortion plots of (a) Cameraman, (c) Lena, (d) Goldhill images. MSSIM vs. bit rate plots of (b) Cameraman, (d) Lena, (f) Goldhill images. 164
6.4 Cameraman image reconstructed at a scale factor of 5 using DCT, Approx. DCT, Bouguezel *et al.*, SDCT and Proposed matrix. 166
6.5 Rate-distortion and MSSIM vs. bit rate plots of (a) and (b) Pepper, (c) and (d) f16 images respectively. 167
6.6 f16 image reconstructed at a scale factor of 5 using DCT, Approx. DCT, Bouguezel *et al.*, SDCT and Proposed matrix. 168

List of Tables

1.1 Storage and transmission need for various types of uncompressed images 3

2.1 Perceptual weights applied to high frequency sub-band at coarsest scale 42
2.2 PSNR(dB) Comparison of DTT_SPIHT With Other Algorithms . . . 44
2.3 Comparison of PSNR(dB) Values of DCT_SPIHT With DTT_SPIHT 45
2.4 Comparison of PSNR(dB) and MSSIM Values of DTT_SPIHT With HVS and Without HVS on Lena Image 45
2.5 Computational complexity comparison between different DTT algorithms and our proposed algorithm. 49
2.6 H/W utilization of 1D floating point and integer DTT in Xilinx XC2VP30. 50
2.7 H/W utilization of 2D floating point, integer and Zigzag prune DTT in Xilinx XC2VP30. 50
2.8 Comparison of PSNR between block-pruned and zigzag-pruned reconstructed images of (a) Lena, (b) Barbara and (c) Crowd. 51
2.9 Comparison of PSNR between block-pruned and zigzag-pruned reconstructed images of (d) Finger print, (e) Mountain, and (f) Library. . . 51
2.10 Comparison of PSNR between DCT and DTT of block pruned and zigzag pruned reconstructed images of (a) Lena, (b) Barbara and (c) Crowd. 51
2.11 Comparison of PSNR between block-pruned and zigzag-pruned reconstructed images of (d) Finger print, (e) Mountain, and (f) Library. . . 52

3.1 Distribution of DTT coefficient magnitudes along thresholds and frequency levels(6: Highest, 1: Lowest) for Lena(512×512) image . . . 71
3.2 Comparison of memory requirements between LSK, SPIHT and ILSK algorithms on top 9 passes. 72
3.3 Parameter comparison matrix. A '+' indicates that it is supported. The more '+' indicate that it is more supported. A '-' indicates separate techniques are required to support. 76

3.4 Comparison of cumulative number of bits generated in the first six passes, BRI improvement and % of bit savings 79
3.5 Performance comparison (PSNR in dB) between various algorithms on Lena and Barbara Images 81
3.6 PSNR(dB) comparison of various standard gray scale images without backend arithmetic coding. All the block based transforms use blocks of size 8×8. 87
3.7 PSNR(dB) comparison of various standard gray scale images without backend arithmetic coding. All the block based transforms use blocks of size 8×8. 88
3.8 PSNR(dB) comparison of standard color images without back end arithmetic coding 88
3.9 Cumulative number of bits generated in the first seven p^{th} passes of DCT_SPIHT, DTT_LSK and HLDTT techniques. 92
3.10 Encoding and Decoding times (msec) on Lena image 94
3.11 PSNR(dB) performance on Lena image with varying block sizes without backend arithmetic coding 94
3.12 Encoding and Decoding time of Lena image with varying block sizes . 95

4.1 Distribution of wavelet coefficient magnitudes along thresholds and frequency levels(Highest: 6, Lowest: 1) for Lena(512×512) image . . . 100
4.2 Comparison of memory requirement between LSK, SPIHT and ILSK algorithms on top 7 passes. 101
4.3 Cumulative number of bits generated in the first six passes of SPIHT, LSK and proposed LEBP technique 107
4.4 Performance comparison (PSNR in dB) between various algorithms on Lena and Barbara Images. No back end arithmetic coding employed. 107
4.5 PSNR(dB) comparison of JPEG2000 standard test images (Grayscale) 108
4.6 PSNR comparison for the SPIHT and LEBP coder at different spatial resolutions (Lena 512×512, 0.05 bpp) 115
4.7 PSNR (dB) performance of various standard gray scale images without using back end arithmetic coding 121
4.8 PSNR(dB) performance on Lena image with varying block sizes . . . 122
4.9 PSNR(dB) performance on Barbara image with varying block sizes . 123
4.10 Encoding and Decoding times of Lena and Barbara images with different bit rates 125

5.1 Comparison of encoded string length between NLS and LBTC for Lena and Barbara Images on top six cumulative bit plane passes 138
5.2 Comparison of PSNR(dB) values between SPIHT, NLS, WBTC and LBTC at very low bit rates 139
5.3 Comparison of PSNR(dB) values between SPIHT, NLS, WBTC and LBTC algorithms 140
5.4 Comparison of PSNR(dB) values between JPEG 2000, SPIHT, NLS, WBTC and LBTC algorithms 141
5.5 Comparison of cumulative encoded string length between DCT_NLS and DCT_LBT for Lena and Barbara images on top six bit plane passes 141
5.6 PSNR comparison between Hou *et al.*, JPEG 2000, Song and Cho and DCT_LBP algorithms on 2560 × 2048 size images. 144
5.7 PSNR comparison between Hou *et al.*, JPEG 2000, Song and Cho and DCT_LBT algorithms on 512 × 512 size images 145
5.8 Comparison of encoded string length between DCT_NLS, DTT_NLS and HLBT_DTT for Lena image on top seven cumulative bit plane passes 146
5.9 Comparison of encoded string length between DCT_NLS, DTT_NLS and HLBT_DTT for Ruler image on top five cumulative bit plane passes 146
5.10 Comparison of PSNR(dB) values between DTT_NLS, DCT_LBT(2×2), DCT_LBT(4×4) and HLBT_DTT(4×4) algorithms on 512×512 size images. 150
5.11 PSNR(dB) comparison between Hou *et al.*, JPEG 2000, Song and Cho, WBTC, DCT_LBT and HLBT_DTT Algorithms 152
5.12 Comparison of encoding and decoding time (sec) for Lena image (Wavelet transform and DCT are not included). 153

6.1 Explanation of the Hardware structure of Figure 6.1. (symbols '+' means positive, '-' means negative and 'No' means no operation) . . . 160
6.2 Proposed Luminance 8 × 8 Quantization Matrix 162
6.3 H/W Utilization of 1-D transform of proposed matrix in XC2VP30. . 162
6.4 H/W Utilization of 2-D transform of proposed matrix in XC2VP30. . 162
6.5 Comparison of Adder Cost Savings 163

List of Abbreviations

JPEG	Joint Picture Expert Group
DCT	Discrete Cosine Transform
DTT	Discrete Tchebichef Transform
DWT	Discrete Wavelet Transform
IWT	Integer Wavelet Transform
RB-IWT	Reversible Integer Wavelet Transform
WHT	Walsh Hardmard Transform
RSWT	Reversible Squre Wave Transform
SDFT	Shifted Discrete Fourier Transform
EZW	Embedded Zerotree Wavelet
PSNR	Peak Signal to Noise Ratio
MSSIM	Mean Structural SIMilarity index metric
SPIHT	Set Partitioning in Hierarchical Trees
EBCOT	Embedded Block Coding with Optimized Truncation
SPECK	Set Partitioning Embedded Block
EZBC	Embedded Zero Block Coding
SBHP	Subband Hierarchical Block Partitioning
PCRD	Post Compression Rate Distortion
NLS	No List SPIHT
LSK	Listless SPECK
MPEG	Motion Picture Expert Group
STQ	Significance Tree Quantization
SLCCA	Significance Linked Connected Component Analysis
EQDCT	Embedded Quad Tree DCT
ICT	Integer Cosine Transform
HVS	Human Visual System
SDCT	Signed Discrete Cosine Transform
DA	Distributed Arithmetic
FPGA	Field Programmable Gate Array
WBTC	Wavelet Block Tree Coding
CSPIHT	Color Set Partitioning in Hierarchical Trees
LMSPIHT	Listless Modified Set Partitioning in Hierarchical Trees
SNR	Signal to Noise Ratio
ROI	Region of Interest
CDF 9/7	Cohen Daubechies Feauveau 9/7 tap filter

MRWD	Morphological Representation of Wavelet Data
LIP	List of Insignificant Pixels
LSP	List of Significant Pixels
LIS	List of Insignificant Sets
DCT_SPIHT	Discrete Cosine Transform-Set Partitioning in Hierarchical Trees
DTT_SPIHT	Discrete Tchebichef Transform-Set Partitioning in Hierarchical Trees
DTT_LSK	Discrete Tchebichef Transform-Listless SPECK
HLDTT	Hierarchical Listless Discrete Tchebichef Transform
DWT_SPIHT	Discrete Wavelet Transform-Set Partitioning in Hierarchical Trees
IWT_SPIHT	Integer Wavelet Transform-Set Partitioning in Hierarchical Trees
IS	Insignificant Set
IL	Insignificant Level
IGL	Insignificant Group-of-Level
ILSK	Improved Listless SPECK
OOS	Optimal Subband Shift
BRI	Bit Rate Improvement
LEBP	Listless Embedded Block Partitioning
CLEBP	Color Listless Embedded Block Partitioning
LBTC	Listless Block Tree Coding
DCT_LBTC	Discrete Cosine Transform-Listless Block Tree Coding
HLBT_DTT	Hierarchical Listless Block Tree Discrete Cosine Transform
MCU	Minimum Coded Unit
NEDA	New Efficient Distributed Arithmetic

Chapter 1

Introduction

Pictures have been with us since the dawn of the time. However, the way the pictures have been presented and displayed has changed significantly. In old age, pictures are represented and displayed in a physical way such as painting in cave walls or etching in stones. In recent times, pictures are dealt electronically. Interestingly, the representation used for storage and transmission is quite different from its display. For example, in traditional broadcast television, where this representation which is transmitted is not directly related to the intensities of red, green and blue electron guns in a television set.

The possibilities of image representation increases dramatically by storing images in digital form. There can be numerous ways an image can be stored in any representation, provided that there should be algorithms to convert back to a form usable for display. This process of changing the representation of an image is called *image coding*. If the image representation consumes less storage space than the original, it is called *image compression* [1].

Most of the encoders discussed in this thesis are based on *progressive encoding* to compress an image into a bit stream with increasing accuracy. This means that when more bits are added to the bit stream, the decoded image will contain more details, a property similar to progressive Joint Picture Expert Group (JPEG) encoded images. It will be similar to the representation of a number like π where addition of every digit increases the accuracy of the number, but it can stop at any desired accuracy as needed. Progressive encoding is also known as *Embedded encoding*.

1.1 Representation of Digital Images

Let $f(s,t)$ be represents a continuous image function of two continuous variables, s and t. The continuous image function $f(s,t)$ can be converted to digital image function $f(x,y)$ (i.e., pixels) by spatial sampling and amplitude quantization, where

(x, y) are spatial coordinates. The 2D array $f(x, y)$ contains M rows and N columns. Therefore, $x = 0, 1, 2, ...M - 1$ and $y = 0, 1, 2, ...N - 1$. The relation between pixel values and image is illustrated in Figure 1.1.

The sampling process may be viewed as partitioning xy-plane into grid with origin at $f(0, 0)$ with x-axis vertically downward and horizontal y-axis directs to the right. The number of samples in the grid determines the resolution of the image. For example, 1024 samples points per row and 768 samples per column, yield an image with resolution 1024 × 768.

Each sample is generally thought of as representing the intensity of a picture element or pixel or pel. The set of intensity values that a pixel can be taken is always a power of 2. If a pixel has 2^n values, then it requires n bits of storage. Thus, a two-tone image (e.g., text) will have binary values for each pixel. Continuous-tone gray scale (monochrome) images generally use 8 bits per pixel (bpp) and color images use 24 bits per pixel (each color component requires 8 bpp). Medical and scientific images typically use more bits per pixel, sometimes up to 16 bpp for gray scale.

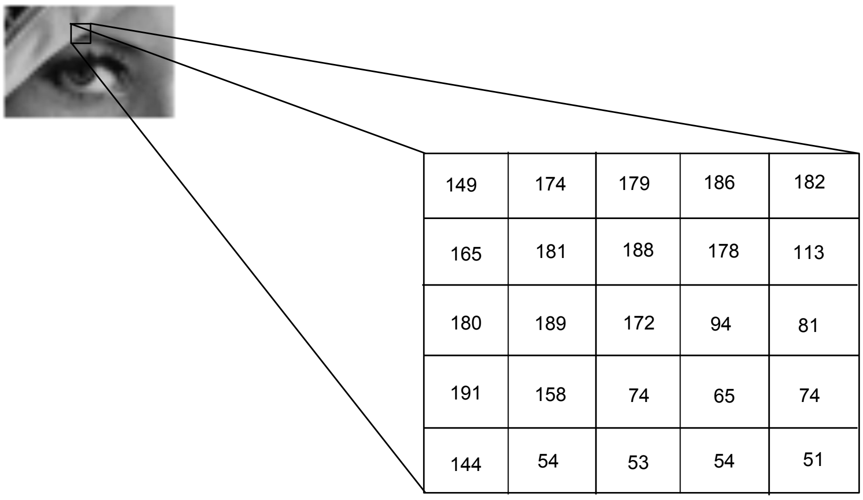
Taking together, the values of all the pixels in an image constitute the raw data representation of the image. The amount of storage required by this raw data will be calculated as the product of the number of pixels and the bits used per pixel. Therefore, the raw data required by 1024 × 768 resolution color image will be:
$1024 \times 768 \times 24$=18874368 bits=2.25 MB.

1.1.1 Need of Image Compression

The raw data required by a single color image may not be a great deal of storage space. Total storage requirements become overwhelming as the number of images need to be stored or transmitted increases. Table 1.1 shows the storage size, transmission bandwidth and transmission time needed for various types of uncompressed images [2]. It is obvious that images requires large transmission bandwidths, more transmission time and large storage space, which are proportional to the size of the image. With the present state of technology, the only solution is to compress the image before its storage and transmission. Then, the compressed image can be decompressed at the receiver end.

1.2 Image Compression Fundamentals

The 2D intensity arrays discussed in section 1.1 are the preferred format for human viewing and interpretation. When it comes to compact image representation, these formats are far from optimal. The reason is that it suffers from three principal kind



149	174	179	186	182
165	181	188	178	113
180	189	172	94	81
191	158	74	65	74
144	54	53	54	51

Figure 1.1: Alternative views of an image data

Table 1.1: Storage and transmission need for various types of uncompressed images

Image type	Size	bits/pixel	Uncompressed	Transmission bandwidth	Transmission time (using 28.8K Modem)
Grayscale	512 × 512	8	262 KB	2.1 Mbit/image	1 min 13 sec
Color	512 × 512	24	786 KB	6.29 Mbit/image	3 min 39 sec
Medical	2048 × 1680	12	5.16 MB	41.3 Mbits/image	23 min 54 sec
Super High Density (SHD)	2048 × 2048	24	12.58 MB	100 Mbit/image	58 min 15 sec

of data redundancy. These are be explained below:

- Coding Redundancy: Coding redundancy is present when the codes assigned to gray levels do not take full advantage of gray levels probability. For example, considering a gray level image having n pixels, the number of gray levels in the image is L and the number of pixels with gray level k is n_k. Then, the probability of occurring gray level k is $p(k) = \frac{n_k}{n}$. If the number of bits used to represent gray level k is $l(k)$, then the average number of bits required to represent each pixel is

  $$L_{avg} = \sum_{k=0}^{L-1} l(r_k) p_k(r_k). \tag{1.1}$$

  Hence, the number of bits required to represent the whole image is $n \times L_{avg}$. Maximum compression ratio is achieved when L_{avg} is minimized. The gray levels are coded in such a way that it results in an image containing coding redundancy if L_{avg} is not minimized.

- Spatial and temporal redundancy: Since the neighboring pixels in an image are highly correlated, information is unnecessary replicated in the representations of correlated pixels. In a video sequence, temporally correlated pixels also duplicate the information.

- Psycho visual redundancy: Image contain information which is not sensitive to the human visual system (HVS) and/or extraneous to the intended use of image.

1.2.1 Measuring Image Information

An important question to answer is 'How many bits are necessary to represent the information of an image ?' or alternatively, Is there any minimum amount of data that are sufficient to describe an image without loss of information ?'. The *information theory* states that, generation of information can be modeled as a probabilistic process. A random event E with probability $P(E)$ contains

$$I(E) = \log \frac{1}{P(E)} = -\log P(E) \tag{1.2}$$

units of information. For true events (i.e., P(E)=1), I(E)=0 and such events do not contain information. Therefore, *information is a measure of uncertainty.* The base of the logarithm determines the number of units used to measure information. If the base 2 is used, the unit of information is the bit. When $P(E) = \frac{1}{2}, I(E) = -log_2\frac{1}{2} = 1$ bit. Therefore, 1 bit is conveyed when one of the two possible equally likely events occurs.

For a source of statistically independent random events $a_1, a_2, ...a_J$ with associated probabilities $P(a_1), P(a_2), ...P(a_J)$, the average information (*entropy* of the source) per source output is:

$$H = -\sum_{j=1}^{J} P(a_j) \log P(a_j). \tag{1.3}$$

where $a_j,\ j = 1, 2, ...J$ are symbols. Since they are statistically independent, the source is called a *zero-memory source.*

Considering an image as an output of an imaginary zero-memory 'intensity source', the histogram of the image can be used to estimate the symbol probabilities of source. The intensity of entropy of source is:

$$\hat{H} = -\sum_{k=1}^{L-1} P_r(r_k) \log_2 P_r(r_k) \tag{1.4}$$

It is not possible to code the intensity values of the imaginary source (and thus the sample image) with fewer $\hat{H}$ bits/pixel.

1.2.2 Fidelity Criteria

Removal of irrelevant visual information may lead to a loss of real or quantitative image information. Two types of criteria can be used to quantify loss of information:

- Objective fidelity criteria: Information loss is expressed as a mathematical function of the input and output of the compression process. For example *root-mean-square* (rms) or *Peak-signal-to-noise-ratio* (PSNR).

  Let $f(\hat{x}, y)$ be an approximation of input image $f(x, y)$ after compression and decompression process. For any value of x and y, the error is

  $$e(x, y) = \hat{f}(x, y) - f(x, y). \tag{1.5}$$

  Therefore, the total error between the two images is:

  $$e = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} [\hat{f}(x, y) - f(x, y)] \tag{1.6}$$

  The rms error is:

  $$e_{rms} = \sqrt{\frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} [\hat{f}(x, y) - f(x, y)]^2}. \tag{1.7}$$

  The PSNR is defined as the ratio of peak signal power to noise power. If the signal lies in the range [0,1], then the expression for PSNR in decibel (dB) scale is given as:

  $$PSNR = 20 \log_{10} \frac{1}{e_{rms}} \, dB. \tag{1.8}$$

  For a monochrome image having pixel range [0, 255], the numerator of Eqn. 1.8 must be changed to 255.

  Though these image metrics in Eqn. 1.7 and Eqn. 1.8 are used extensively for evaluating the quality of the reconstructed image, none of these considered the visual perception system into account. Another metric called Mean Structural SIMilarity Index (MSSIM) which considered image degradation as *perceived change in structural information* rather than *perceive errors* in PSNR or MSE [3]. MSSIM is expressed as:

  $$MSSIM(F, \hat{F}) = \frac{1}{M} \sum_{j=1}^{M} SSIM(f_j, \hat{f}_j) \tag{1.9}$$

  where, F and $\hat{F}$ are the reference and distorted images. f_j and $\hat{f}_j$ are the image content of the j^{th} local window, M is the number of local windows. For each local window

$$SSIM(f,\hat{f}) = \frac{(2\mu_f\mu_{\hat{f}} + c_1)(2\sigma_{f\hat{f}} + c_2)}{(\mu_f^2 + \mu_{\hat{f}}^2 + c_1)(\sigma_f^2 + \sigma_{\hat{f}}^2 + c_2)} \tag{1.10}$$

where μ_f is the average of $f(x,y)$, $\mu_{\hat{f}}$ is the average of $\hat{f}(x,y)$, σ_f^2 is the variance of $f(x,y)$, $\sigma_{\hat{f}}^2$ is the variance of $\hat{f}(x,y)$, $\sigma_{f\hat{f}}$ is the covariance of $f(x,y)$ and $\hat{f}(x,y)$. $c_1 = (k_1L)^2$, $c_2 = (k_2L)^2$ are two variables to stabilize the division with weak denominator. L is the dynamic range of pixel values (typically, $2^{\#bits\ per\ pixel} - 1$). k_1=0.1 and k_2=0.03 by default. If F and $\hat{F}$ are identical then MSSIM=1. For highly uncorrelated case MSSIM=-1.

- Subjective fidelity criteria: Images are viewed by number of viewers and their evaluations are averaged [1].

1.2.3 Image Compression Model

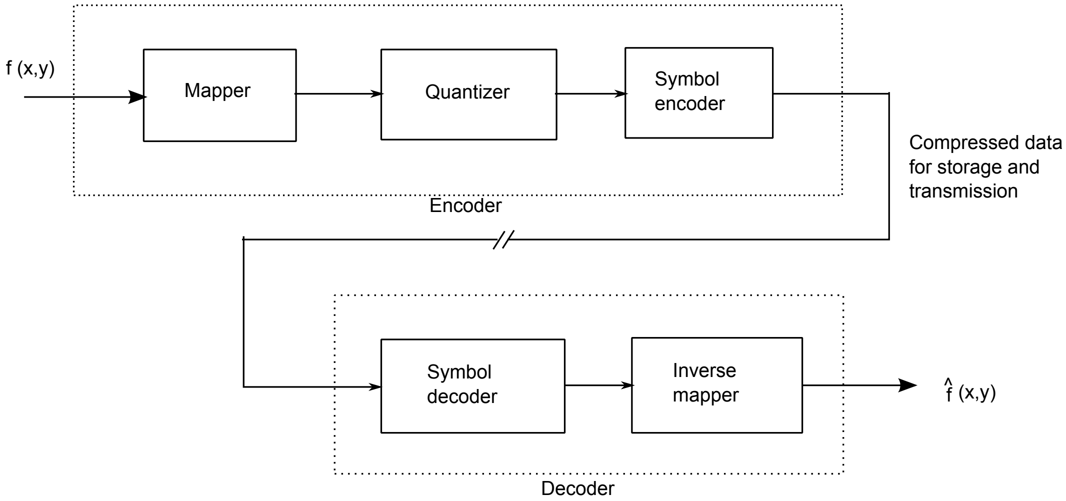


Figure 1.2: General image compression system

A general image compression model is shown in Figure 1.2. The encoder performs compression and decoder performs decompression. The encoder consists of mapper, quantizer and symbol encoder. Usually the mapper transforms an image into an invisible format designed to reduce spatial and temporal (in video sequences) redundancy. Generally, this operation is reversible and may or may not reduce the amount of data needed to represent the image. In video applications, the mapper uses previous and future frames to facilitate removal of temporal redundancy.

The quantizer reduces the accuracy of the output of mapper according to the fidelity criterion. This operation is irreversible and targets to remove irrelevant information from the image. When lossless compression is needed, quantizer must be removed.

The final stage of the encoding process is the symbol encoder, which generates a fixed or variable-length code to represent the quantizer output. Usually the shortest code words are assigned to the most frequently occurring quantizer output values to minimize coding redundancy. This operation is reversible. These three operations lead to removal or decrease of all three redundancies from the input image.

The decoder contains two components: symbol decoder and inverse mapper performing the inverse operations of the symbol encoder and mapper. The inverse quantizer block is not included since quantization is irreversible.

Suppose b and b' represent number of bits in the original data $f(x, y)$ and compressed data $f'(x, y)$ respectively. The relative data redundancy R of the representation with b bits is:

$$R = 1 - \frac{1}{C} \tag{1.11}$$

where C is called the compression ratio, is expressed as:

$$C = \frac{b}{b'}. \tag{1.12}$$

If $C = 10$, the larger representation b has 10 bits of data for every 1 bit of data in the smaller representation b'. This indicates that 90% data($R = 0.9$) are redundant. In the context of digital image compression, b usually is the number of bits needed to represent the original image as 2D array of intensity values (pixels). Considering the case of gray scale image, 8 bits are needed to represent each pixel. Another term called bit rate (BR) which is expressed as:

$$BR = \frac{8}{C} \text{ bits/pixel(bpp).} \tag{1.13}$$

$BR = 0.8$ bpp for $C = 10$. That means, the number of bits that is required to represent the compressed image is 0.8 bits/pixel.

1.2.4 JPEG Baseline Image Compression

JPEG is the first international image compression standard for continuous-tone still gray scale and color images [4]. The goal of this standard is to support a variety of applications for compression of continuous-tone still images of different image sizes in any color space. It has user-adjustable compression ratio with very good to excellent reconstruction quality. It has lower computational complexity for widespread practical applications. Discrete cosine transform (DCT) [5] is used as the transform in the JPEG standard. JPEG defines four modes of operations:

- Sequential lossless mode: It compresses the image in a single scan and the decoded image is an exact replica of the original image.
- Sequential DCT-based mode: It compresses the image in a single scan using DCT-based lossy compression technique. Therefore, the decoded image is an approximation of the original image. This Mode is also called as JPEG baseline mode and it is widely used.
- Progressive DCT-based mode: It compresses/decompresses the image in multiple scans. Each successive scan produces better quality image.
- Hierarchical mode: It compresses the image at multiple resolution for display on different devices.

Each mode consists of a multiple number of options as well, totaling 44 different options or submodes [4], [6],[7]. A brief review JPEG baseline compression is presented in this section because it is widely used (more than 90% of users) among the four modes of JPEG family.

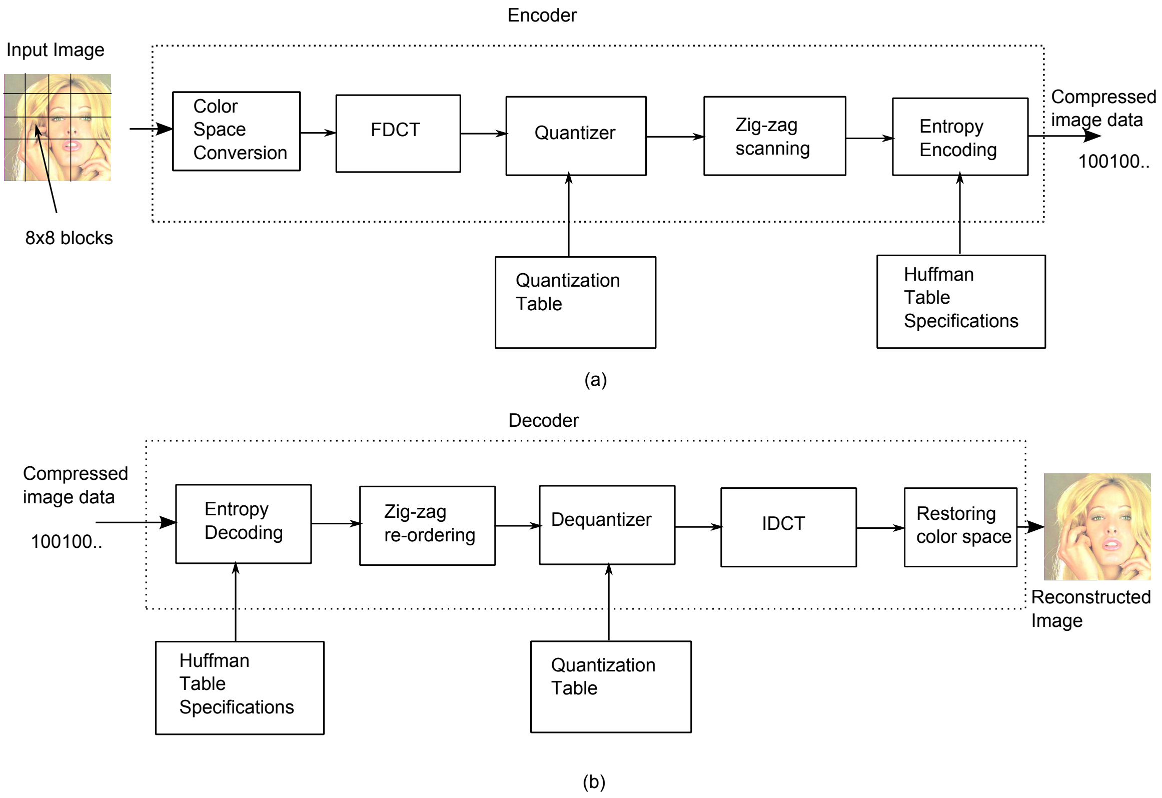


Figure 1.3: Baseline JPEG: (a) Compression, (b) Decompression

In JPEG baseline codec as shown in Figure 1.3, a RGB color image is first mapped into Luminance-chrominance (L-C) color space such as YCbCr, YUV, CIELAB, etc in order to have a better decorrelation between color components. The chrominance channels contain more redundant information and can be subsampled without sacrificing any visual quality of the reconstructed image. Baseline supports 4:2:0, 4:2:2 and 4:4:4 color formats. 4:2:0 format is formed by subsampling the chrominance components by half horizontally and vertically. Each chrominance component in 4:2:2 color format has same vertical resolution as that of luminance component, but the horizontal resolution is halved. In 4:4:4 format both the chrominance components have identical vertical and horizontal resolution as that of luminance components. No color transformation is required for grayscale image.

To apply Forward DCT (FDCT), first the image is divided into non-overlapping 8×8 blocks in raster scan order from left to right and top-to-bottom as shown in Figure 1.3. Then, each pixel is level shifted to convert into signed integer by subtracting 128 from each pixel. The FDCT of an 8×8 block of pixels $f(x, y)$ for $(x, y = 0, 1, ...7)$ is expressed as follows:

$$\begin{aligned}
&F(u,v) = \tfrac{1}{4}C(u)C(v) \sum_{x=0}^{7}\sum_{y=0}^{7} f(x,y) \cos\left[\tfrac{\pi(2x+1)u}{16}\right] \cos\left[\tfrac{\pi(2y+1)v}{16}\right]\\
&\text{for } u = 0, 1, ..., 7 \text{ and } v = 0, 1, ..., 7, \text{ where}\\
&C(k) = \begin{cases} \frac{1}{\sqrt{2}} & \text{for } k = 0 \\ 1 & \text{otherwise.} \end{cases}
\end{aligned} \tag{1.14}$$

After transformation, the transformed coefficients are need to be quantized. This step is primarily responsible for the loss of information and hence introduces distortion in the reconstructed image. Each 64 DCT coefficients are uniformly quantized according to the formula

$$F_q(u,v) = Round\left(\frac{F(u,v)}{Q(u,v)}\right) \tag{1.15}$$

where $F(u, v)$ is a DCT coefficient and $Q(u, v)$ is the quantizer step-size parameter.

The standard does not define any quantization table. It is prerogative of the user's choice to select the quantization matrix. JPEG standard defines two quantization matrices for luminance and chrominance planes. These two quantization matrices have been designed based on the psycho visual experiments by Lohsceller [8] to determine the visibility threshold for 2-D basis functions. These matrices are best suited for natural images with 8-bit precision. If the elements in these tables are divided by 2, perceptually lossless compression is obtained. Quality of the reconstructed image can be controlled by scaling the matrices.

After the transformation and quantization over an 8×8 image sub-blocks, the new 8×8 sub-block shall be reordered in zigzag scan into a linear array. The first coefficient is the DC coefficient and the other 63 coefficients are AC coefficients. The DC coefficient contains lot of energy, hence it is usually of much larger value than AC coefficients. Since there is a very close relation between the DC coefficients of adjacent blocks, the DC coefficients are differentially encoded. This process further reduces entropy. The entropy coding process consists of Huffman coding tables as recommended in JPEG standard. These tables are stored as header information during the compression process so that it is possible to uniquely decode the coefficients during decompression process.

1.2.5 JPEG2000 Image Coding Standard

Although baseline JPEG has been very successful in the market place for more than a decade, it lacks many features desired by interactive multimedia applications, its usage in wired or wireless environments and Internet applications. A fundamental shift in the image compression approach came after the Discrete Wavelet Transform (DWT) became popular [9]-[12]. Exploiting the interesting features in DWT, many scalable image compression algorithms were proposed in the literature [13]-[17]. To overcome the inefficiencies in JPEG standard and serve emerging applications areas in the age of mobile and Internet communications, the new JPEG2000 standard has been developed by ISO/IEC standard committee. It provide a unified optimized tool to accomplish both lossless and lossy compression, as well as decompression using the same algorithm. The systems architecture is not only optimized for compression efficiency for very low bit rates, but also optimized for scalability and interoperability in networks and noisy environments. The JPEG2000 standard will be effective in wide application areas such as Internet, digital photography, digital library, image archival, compound documents, image databases, color reprography (photocopying, printing, scanning, facsimile), graphics, medical imaging, multispectral imaging such as remotely sensed imagery, satellite imagery, mobile multimedia communications, 3G cellular telephony, client-server networking, e-commerce, etc.

The main drawback of the JPEG2000 standard compared to JPEG is that the coding algorithm is much more complex and the computational needs are much higher. Moreover, the bit-plane wise processing may restricts computational performance in a general-purpose computing platform. Analysis [18],[19] shows that the JPEG2000 compression is more than 30 times complex as compared with JPEG. JPEG2000 standard has 12 parts (Part 7 abandoned) with each part adding new features to the core coding standard in Part 1. Out of 11 parts some parts are still under development (Part 8-Part 12). Part 1-Part 6 are explained in [20]-[25].

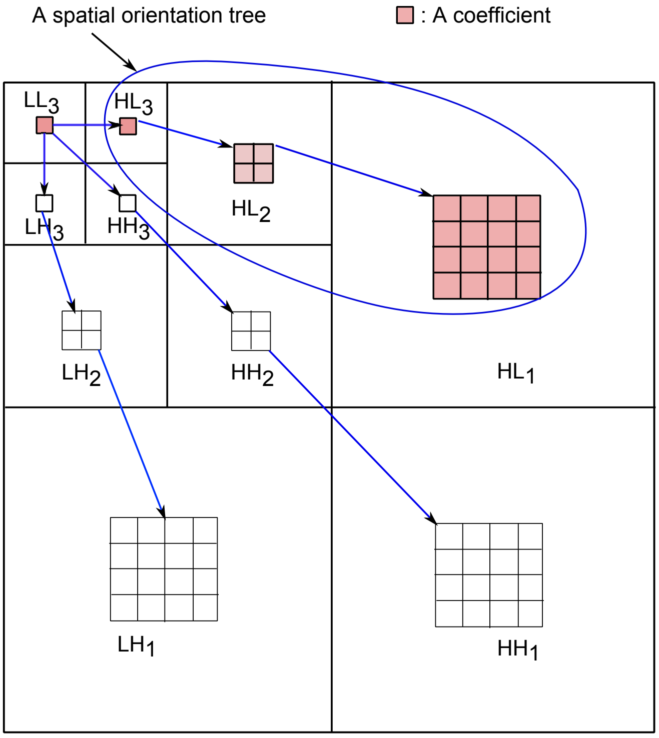


Figure 1.4: Parent-child relationship of wavelet coefficients of a 3-level wavelet decomposition pyramid.

1.2.6 Embedded Image Compression

The Shapiro's EZW (Embedded Zerotree Wavelet) coder exploits the self similarity of the wavelet transformed image across different scales by using a hierarchical tree structure [15]. The coefficients are viewed as a collection of spatial orientation trees. Each tree consists of coefficients from all subbands (both frequency and orientation) that corresponds to the same spatial area in an image. The parent child relationship of a 3-level wavelet decomposition structure is shown in Figure 1.4. The coefficient in LL_3 band (root) does not have any children. The coefficients in HL_3, LH_3 and HH_3 have four children each. A coefficient c_{ij} is called significant with respect to a given threshold n if $|c_{ij}|\geq n$. Otherwise, it is insignificant. Meaningful image statistics have shown that if a wavelet coefficient is insignificant at a particular threshold T, it is very likely that its descendants are insignificant with respect to the same threshold. This constitutes a *zerotree*. A single symbol can be used efficiently to encode all the coefficients in a zerotree.

EZW coding can be thought of as bit-plane coding if the thresholds are powers of 2. It encodes one bit plane at a time starting from most significant bitplane (MSB). With successive bitplane coding and scanning of the trees from lower to higher frequency subbands on bit-plane, EZW achieves embedded coding.

The bit-mapped position information of significant coefficients at each threshold is called *significance map.* The successive passes of sorting (Dominant) and refinement (Subordinate) with decreasing threshold is called *successive approximation quantization.*

The dominant pass and subordinate pass are repeated for every bit-plane pass generating an encoded bit string. This process stops whenever the size of the encoded bit stream reaches the exact target bit rate. By exploiting the parent-child relationship across different scales in a wavelet transformed image, progressive wavelet coders can effectively order the coefficients by bit-planes and transmit most significant information first. Therefore, it results in an embedded bit stream with a set of attributes like progressive transmission, precise rate control, resolution and fidelity scalable. These set of rich features are completely absent in JPEG.

Though a number of coding methods have been proposed in the literature, the fundamental idea inherits from EZW algorithm. One of the listed method is Set partitioning in hierarchical trees (SPIHT) algorithm [16]. SPIHT is very popular since it is able to achieve equal or better performance than EZW without using arithmetic coding. The reduction in complexity from eliminating the arithmetic encoder is significant. Therefore, it is used frequently as a benchmark for performance in evaluation of the state-of-the-art image compression algorithms.

1.3 Background and Scope

Over the past few years, a variety of powerful and sophisticated wavelet based image compression schemes such as Embedded Zerotree Wavelet (EZW) coding scheme, Set Partitioning in Hierarchical Trees (SPIHT), Set Partitioning in Embedded Block (SPECK)[26] and Embedded Block Coding with Optimized Truncation (EBCOT)[27] have been developed. Among the wavelet zerotree based image coding algorithms, SPIHT is the most recognized coding method because of its excellent rate-distortion performance. SPECK is a block based low complexity coding scheme compared to SPIHT because it consists of only two ordered list data structures, whereas SPIHT consists of three ordered auxiliary list data structures. However, performance of SPECK is closer to SPIHT. EBCOT is also block based coding scheme with modest amount of complexity.

SPIHT exploits zero-tree structure, whereas SPECK exploits zero-block structure to achieve inter and intra subband correlations. In zero-tree based algorithms, wavelet coefficients corresponding to same spatial location and orientation are grouped to form a spatial orientation tree. In significance test, a tree with no significant coefficient with respect to a given threshold is coded as zerotree. On the other hand, zero-block based algorithms divide the transformed coefficients into contiguous blocks and perform significant test on the individual blocks. Insignificant blocks are coded as zero blocks while significant blocks are recursively partitioned for search of significant coefficients. The advantage of this method is that it uses adaptive quadtree splitting scheme to zoom into high energy areas in a region to code the blocks with minimum significant maps. Other well-known block based algorithms are embedded zero-block coding (EZBC) by Hsiang and Woods [28] and Subband Hierarchical Block Partitioning (SBHP) by Chrysafis *et al.* [29]. SBHP is a form of SPECK incorporated into JPEG 2000 under development. EZBC exploited the dependence among quadtree representations of subbands and sophisticated context based arithmetic coding to improve the coding efficiency. Danyali and Mertins [30] proposed fully scalable-SPIHT (FS-SPIHT) suitable for heterogeneous networks where users having different network access bandwidth and processing capabilities. Recently, Xie *et al.* [31] enabled SPECK to have full scalability based on the idea of quality layer formation similar to Post Compression Rate Distortion (PCRD) in JPEG 2000. Being a block based coder, EBCOT which is adopted in JPEG 2000 standard, generates feature rich bit streams with low memory requirements but it is highly complex. This is due to use of multiple coding passes within each bit plane, use of context adaptive arithmetic coding and rate-distortion optimization. Cho and Pearlman [32] addressed the reason for different coding performances between different zerotree coding schemes, which are EZW and SPIHT. Subsequently, Moinuddin *et al.* [33],[34] proposed list based block-tree coding algorithms which reduces the dynamic memory requirements with excellent low bit rate performance.

Most of the algorithms discussed above require a large amount of memory space and need for memory management as the list nodes are updated on each bit plane pass. To overcome these shortcomings, listless variants of SPIHT (i.e., No list SPIHT (NLS))[35] and SPECK (Listless SPECK (LSK))[36] have been reported in literature. However, the performance NLS and LSK are very closer to SPIHT and SPECK respectively. Hence, there is a scope to further improve the performance of LSK and NLS.

Though wavelet based coding algorithms provides substantial improvement in image quality at lower rates compared to DCT based coders at a cost of complexity, DCT is still used in many applications such as JPEG[4], MPEG-4 and H.264 [37],[38] because of its compression performance and computational advantages. Recently, DCT based coders with innovative data organization strategies and representation of coefficients have been reported with high compression efficiency [39]-[44]. Embedded image coder based on DCT by Xiong *et al.* [39]. They have introduced a wavelet-like tree structure of DCT coefficients and applied embedded zerotree quantizer to the DCT coefficients as in EZW coder, which yielded a better performance than wavelet based EZW. Davis and Chawla [40] have proposed significance tree quantization (STQ) optimized for a given class of images. Monoro and Dickson [41] have applied sorting algorithm of EZW. Junqiang and Zhuang [42] have applied SLCCA wavelet-based image coder to DCT subbands. Hou *et al.* [43] have presented an image coder that utilizes set partitions based on quadtree splitting (EQDCT). It provides excellent coding performance with lower complexity. Recently, Song and Cho [44] have reported DCT based embedded coders with compression performance higher than JPEG2000 for texture images.

A new class of transform called Discrete Tchebichef Transform (DTT) which is derived from a discrete class of popular Tchebichef polynomials, is a novel orthonormal version of orthogonal transform. It has found applications on image analysis and compression [45]-[49]. Mukundan [45]-[47] proposed orthonormal version of Tchebichef moments and analyzed some of their computational aspects. Mukundan and Hunt [48] have shown that for natural images, DTT and DCT exhibit similar energy compactness performance. Lang *et al.*[49] have made a comparison between 4×4 Tchebichef moment transform and DCT. They claim that there is a significant advantage for 4×4 Tchebichef moments in terms of error reconstruction and average length of Huffman codes. A block wise moment computation scheme which avoids numerical instabilities to yield a perfect reconstruction has been introduced in the literature [50]. For computation of Tchebichef moments, a number of fast algorithms have been proposed [51]-[53]. The Tchebichef moment compression is meant for smaller computing devices owing to its low computational complexity. Ishwar *et al.* [52] have shown that DTT has lower complexity since it requires the evaluation of only algebraic (only add and shift operations, no multiplications) expressions whereas implementation of DCT requires integer approximation or intermediate scaling like Integer cosine transform (ICT) [37]. Abdelwahab [53] has proposed a fast 2×2 pruned DTT algorithm for 4×4 DTT. This reduces computational complexity by 26% compared to the algorithm in [51] without reducing the image reconstruction accuracy. Several algorithms for pruning the 1-D DCT in [54]-[58] and 2-D DCT in [59]-[62] have been addressed. Therefore, there is a need to develop DTT based fast pruning algorithms with better PSNR performance.

Some important characteristics of DTT can be summarized as follows:

- A discrete domain of definition which matches exactly with image coordinates space.

- Absence of numerical approximation terms allows a more accurate representation of image features than others which is not possible using conventional transforms.
- DTT is invariant to linear transforms and can be efficiently used for image reconstruction [45].
- Dynamic range of DTT is comparable to that of DCT [52].
- DTT is robust against channel errors [63].
- DTT polynomials have properties that matches closely with Human visual systems (HVS) [64].
- In video compression, the prediction residuals of motion compensation can contain large variations. DTT can help a consistent video quality when inter-frame coding is used [65].

Therefore, there is a need to further analyze the performance of DTT over DCT on a JPEG baseline codec and embedded codec coupled with some novel coefficient arrangement techniques.

The performance of Listless embedded coding algorithms such as NLS and LSK can be improved using some novel techniques. The algorithms can be coupled with wavelet and DCT/DTT based transforms in order to access the performance over other wavelet or DCT based SPIHT coders. For complexity constrained encoding situations where even a fast fixed-complexity DCT algorithm is too complex, one can resort to approximate the computation of DCT at the cost of some degradation in the image quality. These applications could be multimedia, mobile communications, personal digital assistants (PDAs), digital cameras and Internet where a lot of image transmission and processing are required.

Even though a number of algorithms for fast computation of DCT are available in the literature, there has been a lot of interest towards finding out the approximate integer versions of floating point DCT [66]-[71].A family of integer cosine transforms (ICT) using the theory of dyadic symmetry is proposed [66] where it has been shown that the performance of ICTs are close to that of DCT. A novel architecture has been presented [67] for a 2D 8×8 DCT which needs only 24 adders. The architecture allows scalable computation of 2D 8×8 DCT using integer encoding of 1D radix-8 DCT. 8×8 versions of two transformation matrices, one for the coarsest and another for the finest (represented as $\hat{D}_1$ and $\hat{D}_5$ respectively) approximation levels of exact DCT have been proposed in [68]. Using these two matrices, a trade off of speedup versus accuracy in various bit ranges can be achieved. The performance shows 73 % complexity reduction with only 0.2 *dB* PSNR degradation. A family of 8×8 biorthogonal transforms called binDCT which are all approximates of popular 8×8 DCT have been proposed in [69]. These binDCT show a coding gain of range 8.77-8.82 *dB* despite requiring as low as 14 shifts and 31 additions per eight input samples. 8×8 binDCT shows finer approximations to exact DCT and are suitable for VLSI implementation. A new kind of transform called signed DCT (SDCT) by applying signum function to DCT is proposed in [70]. However, SDCT and its inverse are not orthogonal and it needs 24 additions for transformation. A 8×8 transform matrix is presented in [71] by appropriately inserting 20 zeros into the elements of $\hat{D}_1$ [68]. A reduction of 25 % in computation is achieved over SDCT and this matrix is orthogonal. Unlike the proposed matrices [72]-[74], the transform order need not be a specific integer or a power of 2.

It requires a number of multipliers to implement a transform kernel using conventional approach. Multipliers are the major source of power hungry elements in a hardware device. Here the focus is given on distributed arithmetic (DA) computation which do not require multipliers [75]-[77].Several DA based approaches has been presented in the literature. These approaches uses either look-up table [75] or without look-up table [76],[77] techniques. Therefore, there is a scope to develop integer based novel 8×8 orthogonal sparse transform matrix for the considered set of applications.

1.4 Motivation

Recent applications such as multimedia, mobile communications and Internet require faster algorithms that can compress the image efficiently with reduced complexity. The amount of information that is transported by the computing device/network is also growing exponentially. Therefore, good quality images at earlier stages of transmission are becoming an important element in these types of codecs. This is particularly important if the image is browsed over wireless lines where limited channel capacity, storage requirements and computational complexity are decisive factors. Therefore, NLS and LSK could be the best candidates for the above applications because of their low complexity and fast encoding/decoding speed. However, the performance of these coders is poor at low bit rates. In other words, the decoded image quality is poor at the earlier stages of transmission. Some of the algorithms like wavelet block-tree coding (WBTC) [33], listless modified SPIHT (LMSPIHT) [78] improve the low bit rate performance of SPIHT. WBTC is a wavelet block-tree algorithm which also makes use of three ordered list structure. Though it reduces the memory requirement compared to SPIHT, it is still undesirable for real time hardware implementations because of its list arrays. WBTC requires a lot of memory management as the list nodes are added, removed or moved during bit plane passes. Subsequently, a variant of WBTC algorithm is proposed in [34] that can compress color images. This algorithm also uses similar kind of list arrays. It shows an improved low bit rate performance in case of color images compared to CSPIHT [26]. LMSPIHT is a listless version of modified SPIHT. The low memory feature and excellent low bit rate performance makes LMSPIHT a good candidate for the considered set of applications. However, its performance degrades compared to SPIHT in most of the images at higher rates.

As per the above discussion, some efficient algorithms based on wavelet or block transform are developed. These algorithms provides

- Better low bit rate performance without sacrificing much on the higher rates.
- Lower computational complexity and memory requirement.
- Rich set of features such as scalable in terms of pixel accuracy and resolution, random access, region of interest (ROI) and precise rate control in most of wavelet based coders.

1.5 Objective

Strong academic and commercial interest in image compression results various efficient compression techniques. Some of these techniques have evolved into international standard such as JPEG and JPEG2000. However, the manifold of multimedia applications demand for further improvement in image quality. JPEG 2000 provides superior low bit rate performance with increase of bit stream functionalities than JPEG. This is achieved with a substantial increase of computational complexity.

The objective of the research work is to develop reduced memory, low complexity image compression algorithms which exhibit superior low bit rate performance with a set of desirable attributes like resolution scalability, region of interest (ROI) retrievability, random access decodability and embeddedness. Since block based transforms such as DCT and DTT are have near optimal energy compaction properties, it is possible to make use of these transforms with innovative wavelet like coefficient arrangements to improve the rate distortion performance, while retaining most of the above set of attributes. Therefore, the objectives can be summarized as:

- To analyze the performance of DTT on JPEG baseline as well as embeddded codecs for various kind of test images and to devise a method that reduces the computation as well as hardware requirements.

- To improve the performance of low complexity algorithms such as LSK and NLS, especially at lower rates using DTT based some novel proposed techniques.
- To estimate the complexity and performance improvement of the improved LSK and NLS with the state-of-the-art wavelet based embedded coders such as SPIHT, SPECK, WBTC, JPEG 2000. To incorporate some desirable features such as pixel scalability, resolution scalability, random access decodability and ROI retrievability to the proposed algorithms.
- To extend the domain of improved LSK algorithm for compressing color images and also compare its performance with CSPIHT and other embedded coding techniques.
- To develop efficient low complexity DCT/DTT based embedded coding algorithms especially suitable for low bit rate applications. The decoded image quality performance between proposed DCT and DTT based embedded coders are to be analyzed in comparison with existing state-of-the-art DCT based embedded coders.
- To develop other low complexity transform matrices suitable for portable multimedia applications, such as digital camera, camcorders, etc.

1.6 Organization of the Dissertation

Including this introductory chapter, the thesis has been divided in to seven chapters. The organization of the dissertation is presented below:

Chapter II
Compression performance assessment of discrete Tchebichef transform
This chapter presents the compression performance (PSNR vs. compression ratio) of DTT on a JPEG baseline codec. DTT has been evaluated on 8×8 blocks of image data from left to right and then top to bottom over the whole image in a raster scan order like DCT in JPEG. The compression ratios at different scale factors are also compared between DCT and DTT. A 3×3 zigzag pruning low complexity DTT algorithm is developed and its impact on the decoded image quality is also studied. The distributed arithmetic (DA) based algorithm is also implemented on Xilinx FPGA device to compare it's hardware utilization over DCT and ICT. Finally, this chapter deals with

DTT_SPIHT embedded coding algorithm which combines DTT with SPIHT coding algorithm. The algorithm also incorporates human visual system (HVS) properties to improve the visual performance of the decoded images. The comparison with DCT_SPIHT algorithm indicates a significant PSNR improvement on images having sharp edges and slightly inferior performance with smooth as well as textured images.

Chapter III

A reduced memory, low complexity embedded image compression algorithm using Hierarchical listless DTT

An improved LSK (ILSK) algorithm which combines with DTT algorithm is presented in this Chapter. The ILSK algorithm is based on the fundamental principle of coefficient decaying spectrum of transformed images where it is possible to encode more number of subband blocks using few symbols. The proposed coder named as Hierarchical listless DTT (HLDTT) shows improved low bit rate performance (PSNR or MSSIM) compared to DCT_SPIHT based embedded coders. From extensive simulations, it is verified that the MSSIM performance of HLDTT decoded images are better over most of the existing DCT based embedded coders over a wide range of bit rates. HLDTT reduces the dynamic memory requirements at earlier passes significantly. It also possesses region of interest retrievability (ROI) and random access decodability features. Further, the listless feature of HLDTT reduces the overall hardware complexity and hence, it can be an attractive candidate for browsing images over narrow bandwidth channel or streaming data in the Internet.

Chapter IV

Reduced memory listless scalable embedded image compression algorithms

In this chapter, two listless algorithms are proposed which are Listless embedded block partitioning (LEBP) for gray scale images, and Color listless embedded block partitioning (CLEBP) for color images. The LEBP algorithm comprises ILSK algorithm with lift based discrete wavelet transform. The CDF 9/7 filter taps are used in the lift based wavelet transform because of its good compression performance. The coding performance of these algorithms are compared with some of the best known wavelet based coding algorithms such as SPIHT, SPECK, LSK, NLS and JPEG 2000. With minor rearrangement of the encoded output, LEBP generates feature rich bit stream. This could be an alternative to JPEG2000 for some applications. The reduction of dynamic memory at lower rates and improved scalability of the proposed algorithms by pixel accuracy and resolution are very effective for real-time browsing of images on the Web, downloading or reconstructing the images in a system with limited memory buffer, transmission of images through limited bandwidth channels, decoding the images depending on the available resolution of the rendering systems, etc.

Chapter V

Listless block-tree set partitioning algorithm for very low bit-rate embedded image compression

In this chapter, an improved depth first search algorithm has been developed instead of improved breadth first search algorithm proposed in Chapter 3 and 4. The proposed algorithm named as Listless block-tree embedded coder (LBTC) is a listless approach to wavelet block-tree coding (WBTC) algorithm. LBTC not only reduces the dynamic memory requirements compared to wavelet based WBTC and SPIHT, but also speeds up compression and decompression process without compromising image quality especially at lower bit rates. Further, the proposed algorithm is combined with DCT and DTT block transforms to indicate the superiority compared to almost all the DCT based embedded coders.

Chapter VI

An efficient sparse 8×8 transform matrix for Image compression

A low complexity orthogonal transform matrix by appropriately inserting more number of zeros is proposed in this chapter. The compression performance of the proposed matrix is evaluated on a JPEG baseline codec and compared with some of the DCT/ICT based algorithms for gray scale and color images. A DA based algorithm for fast computation of the matrix is also developed. FPGA synthesis results show that the matrix needs less number of hardware resources. Hence, it can be used in low power and complexity constrained H/W platforms.

Chapter VII

This chapter concludes the dissertation with the contributions. Details of further research directions which can be attempted subsequently are also presented.

1.7 Conclusions

This chapter provides a brief introduction about the fundamentals of image compression and information theory. The fidelity criteria for evaluating the quality of decoded images is discussed. A brief introduction to JPEG image compression and embedded image compression is presented. The background and scope of the work, the motivation and the objective of the thesis are systematically discussed. A brief chapter wise description has been also presented.

Chapter 2

Compression Performance Assessment of Discrete Tchebichef Transform

Preview

The Discrete Tchebichef Transform (DTT) based on orthogonal Tchebichef polynomials can be an alternative to Discrete Cosine Transform (DCT) for JPEG image compression standard. The properties of DTT are not only very similar to DCT, but it has also higher energy compactness and lower computational advantage using a set of recurrence relation. Through extensive simulation, image reconstruction accuracy (i.e., PSNR and MSSIM) and the compression performance at various scaling factors for both DCT and DTT is verified. It has been demonstrated that DTT exhibit better PSNR/MSSIM performance than DCT for images having higher intensity gradation. DTT shows nearly similar PSNR/MSSIM performance with DCT for smooth and textured images. It is also further verified that DTT has better compression ability than DCT in most of the images.

A DTT based hybrid embedded coder has been proposed for image compression applications. In this coder, DTT is coupled with Set partitioning in hierarchical coding techniques (SPIHT). Further, human visual system (HVS) with appropriate perceptual weights are applied to improve the perceptual quality of the reconstructed image. The compression and image reconstruction performance has been compared with some the state-of-the-art coders in the literature. Extensive simulations on various kinds of images indicates strongly that the proposed coder outperforms most of the coders.

A fast zigzag pruning DTT algorithm of different prune lengths has been proposed and compared with the existing DTT fast algorithms. The principal idea of the proposed algorithm is to make use of the distributed arithmetic and symmetry property of 2-D DTT, which combines the similar terms of the pruned output. Normalization of each coefficient has been done by merging the multiplication terms with the quantization matrix so as to reduce the computation. Equal number of zigzag pruned coefficients and block pruned coefficients are used for comparison to test the efficiency of our algorithm. Experimental method shows that the proposed method is quite competitive with the block pruned method. Specifically for 3×3 block pruned case, the proposed method provides lesser computational complexity and has higher peak signal to noise ratio (PSNR). The reconstructed image quality of different pruned length is evaluated both subjectively and objectively. The proposed method has been implemented on a Xilinx XC2VP30 FPGA.

2.1 Introduction

Image Transform methods using orthogonal kernel functions are commonly used in image compression. One of the most widely known image transform method is Discrete Cosine Transform (DCT), used in JPEG compression standard [4].The computing devices such as Personal digital assistants (PDAs), digital cameras and mobile phones require a lot of image transmission and processing. Therefore, it is essential to have efficient image compression techniques which could be scalable and applicable to these smaller computing devices. Discrete Tchebichef Transform (DTT) which is derived from a discrete class of popular Tchebichef polynomials is a novel orthonormal version of orthogonal transform. It has found applications on image analysis and compression [45],[49].

Though various efficient compression techniques have been reported, the wide range of multimedia applications demands for further improvement in compression quality. Therefore, most of the research activities are focused on wavelet based coders rather than DCT based image coders. Wavelet based coders offers superior performance in terms of visual quality and PSNR at very low bit rates (below 0.25 bpp) [15],[16],[79],[80]. This is mainly attributed due to innovative strategies of data organization and representation of wavelet transformed coefficients.

Although wavelets are capable of more flexible space-frequency resolution trade offs than DCT, DCT is still widely used in many practical applications because of its compression performance and computational advantages. Recently, DCT-based coders with innovative data organization strategies and representations of DCT coefficients have been reported with high compression efficiency [39]- [43].

Recently, DTT has found excellent rate-distortion trade-off like DCT and outperforms DCT for image having high intensity gradations [48],[81]. Therefore, DTT is used as a substitute for DCT in an embedded coder. Further, human visual system (HVS) has been applied to increase the subjective quality of the image [82]. The proposed embedded coder consists of HVS with DTT and SPIHT coding techniques. The performance of this kind of coder has been evaluated and compared with DCT based embedded coders, JPEG, improved JPEG [83], Significance tree quantization (STQ) [40], and STQ+Haar.

The Tchebichef moment compression that has been proposed in this dissertation is meant for smaller computing devices owing to its low computational complexity [51]-[53]. Ishwar *et al.* [52] have shown that DTT has lower complexity since it requires the evaluation of only algebraic (only add and shift operations, no multiplications) expressions, whereas implementation of DCT requires integer approximation or intermediate scaling, like Integer cosine transform (ICT) [37].

There are many DCT compression algorithms which can be computed in a fast way by means of direct or indirect methods [84]-[86]. These algorithms assume same number of input and output points. However, in image coding applications, the most useful information about the image data is kept in the low-frequency DCT coefficients. Therefore, only these coefficients could be computed. This gives rise to the application of pruning techniques. Additional processing speed-up is also possible using this idea. Several algorithms for pruning the 1-D DCT in [54]-[58] and 2-D DCT in [59]-[62] have been addressed.

A 2×2 block pruned out of 4×4 DTT algorithm which computes the upper left quarter of 4×4 image blocks has proposed in [53]. Saleh [87] has proposed a fast 4×4 algorithm suitable for different block sizes. Having surveyed on different DCT pruning algorithms, a fast zigzag pruning algorithm and its image reconstruction quality for image coding applications have been proposed.

2.2 Discrete Tchebichef Transform

The Discrete Tchebichef Transform (DTT) is relatively a new transform that uses the Tchebichef moments to provide a basis matrix. As with DCT, the DTT is derived from the orthonormal Tchebichef polynomials. This leads to presume that it will exhibit similar energy compaction properties [45].

For a 2-D image function $f(x, y)$ on the discrete domain of $[0,\ N-1] \times [0,\ M-1]$, the discrete forward Tchebichef Transform of order $(u \times v)$ is defined as:

$$T_{uv} = \frac{1}{\tilde{\rho}(u,N)\tilde{\rho}(v,M)} \sum_{x=0}^{N-1} \sum_{y=0}^{M-1} f(x,y)\tilde{t}_u(x)\tilde{t}_v(y) \tag{2.1}$$

where, $u, x = 0, 1, 2....., N-1.$, and $v, y = 0, 1, 2, ..., M-1.$

Given a set of Tchebichef transform T_{uv} for a digital image $f(x,y)$, the inverse transformation of Tchebichef moment can be defined as:

$$f(x,y) = \sum_{u=0}^{N-1}\sum_{v=0}^{M-1} T_{uv}\tilde{t}_u(x)\tilde{t}_v(y) \quad \text{where, } u,x = 0,1,2.....,N-1., \text{ and } v,y = 0,1,2,.....,M-1. \tag{2.2}$$

The scaled Tchebichef polynomials $\tilde{t}_u(x)$ are defined using the following recurrence relation [88]:

$$\tilde{t}_u(x) = \frac{(2u-1)\tilde{t}_{u-1}(x)-(u-1)\left(1-\frac{(u-1)^2}{N^2}\right)\tilde{t}_{u-2}(x)}{u} \quad \text{where, } u = 0,1,.......N-1., \text{ and } \tilde{t}_0(x) = 1,\ \tilde{t}_1(x) = \frac{2x+1-N}{N} \tag{2.3}$$

The definition as specified above uses the following scale factor [46] for the polynomial of degree u as

$$\beta(u,N) = N^u. \tag{2.4}$$

The set $\{\tilde{t}_u(x)\}$ has a squared-norm given by

$$\tilde{\rho}(u,N) = \sum_{i=0}^{N-1}\{\tilde{t}_u(x)\}^2 = \frac{N\left(1-\frac{1}{N^2}\right)\left(1-\frac{2^2}{N^2}\right)......\left(1-\frac{n^2}{N^2}\right)}{2n+1} \tag{2.5}$$

The values of the squared-norm affect the magnitudes of the corresponding moments T_{uv}. As specified in [46], the computation of T_{uv} can lead to erroneous results when N is large. This problem can be solved by constructing orthonormal versions of Tchebichef polynomials by modifying the scale factor in Eqn. 2.4 as

$$\beta(u,N) = \sqrt{\frac{N(N^2-1)(N^2-2^2)........(N^2-n^2)}{2n+1}} \tag{2.6}$$

By denoting the new set of polynomials with the above scale factor as $\{t_u\}$, the recurrence relation given in Eqn. 2.3 can change to the following

$$t_u(x) = (A_1x + A_2)t_{u-1}(x) + A_3t_{u-2}(x). \quad \text{where, } u = 2,3,..........N-1;\ x = 0,1,........N-1. \tag{2.7}$$

The coefficients A_1, A_2 and A_3 are as follows:

$$A_1 = \frac{2}{u}\sqrt{\frac{4u^2-1}{N^2-u^2}},\ A_2 = \frac{1-N}{u}\sqrt{\frac{4u^2-1}{N^2-u^2}},\ \text{and } A_3 = \frac{u-1}{u}\sqrt{\frac{2u+1}{2u-3}}\sqrt{\frac{N^2-(u-1)^2}{N^2-u^2}}. \tag{2.8}$$

The starting values for the above recursion can be obtained from the following equations:

$$t_0(x) = \frac{1}{\sqrt{N}},\ t_1(x) = (2x + 1 - N)\sqrt{\frac{3}{N(N^2 - 1)}}. \tag{2.9}$$

Denoting the squared norm by $\tilde{\rho}(u, N)$, so that

$$\tilde{\rho}(u, N) = \sum_{i=0}^{N-1} \{t_u(i)\}^2 = 1.0 \tag{2.10}$$

The moment equation in Eqn. 2.1 now reduces to

$$\begin{aligned} &T_{uv} = \sum_{x=0}^{N-1}\sum_{y=0}^{M-1} t_u(x)t_v(y)f(x, y). \\ &u = 0, 1, 2, ...N - 1; v = 0, 1, 2, ...M - 1. \end{aligned} \tag{2.11}$$

The inverse moment transform becomes

$$\begin{aligned} &f(x, y) = \sum_{u=0}^{N-1}\sum_{v=0}^{M-1} T_{uv}t_u(x)t_v(y). \\ &x = 0, 1, 2, ...N - 1;\ y = 0, 1, 2, ...M - 1. \end{aligned} \tag{2.12}$$

Eqn. 2.12 can also be expressed using a series representation involving matrices as follows

$$\begin{aligned} &f(x, y) = \sum_{u=0}^{N-1}\sum_{v=0}^{M-1} T_{uv}\psi_{uv}(x, y) \\ &u = 0, 1, ...N - 1;\ v = 0, 1, ...M - 1. \end{aligned} \tag{2.13}$$

where, ψ_{uv} is called basis image. Assuming equal image dimensions N=M=8, the basis image ψ_{uv} can be represented as:

$$\psi_{uv} = \begin{bmatrix} t_u(0)t_v(0) & t_u(0)t_v(1) & .. & t_u(0)t_v(7) \\ t_u(1)t_v(0) & t_u(1)t_v(1) & .. & t_u(1)t_v(7) \\ .. & .. & .. & .. \\ t_u(7)t_v(0) & t_u(7)t_v(1) & .. & t_u(7)t_v(7) \end{bmatrix} \tag{2.14}$$

Therefore, Tchebichef transform of a square image $I = \{f(x, y)\}_{x,y=0}^{x,y=N-1}$ as in Eqn. 2.1 can be viewed as the projection of the image I on the basis image ψ_{uv}, which is the product of the vectors t_u and t_v. Where $t_u = [t_u(0)\, t_u(1) \,.....\, t_u(N - 1)]$ and $t_v = [t_v(0)\, t_v(1) \,.....\, t_v(N - 1)]$. Eqn. 2.14 can be written as:

$$\psi_{uv} = [t_u]'[t_v] \tag{2.15}$$

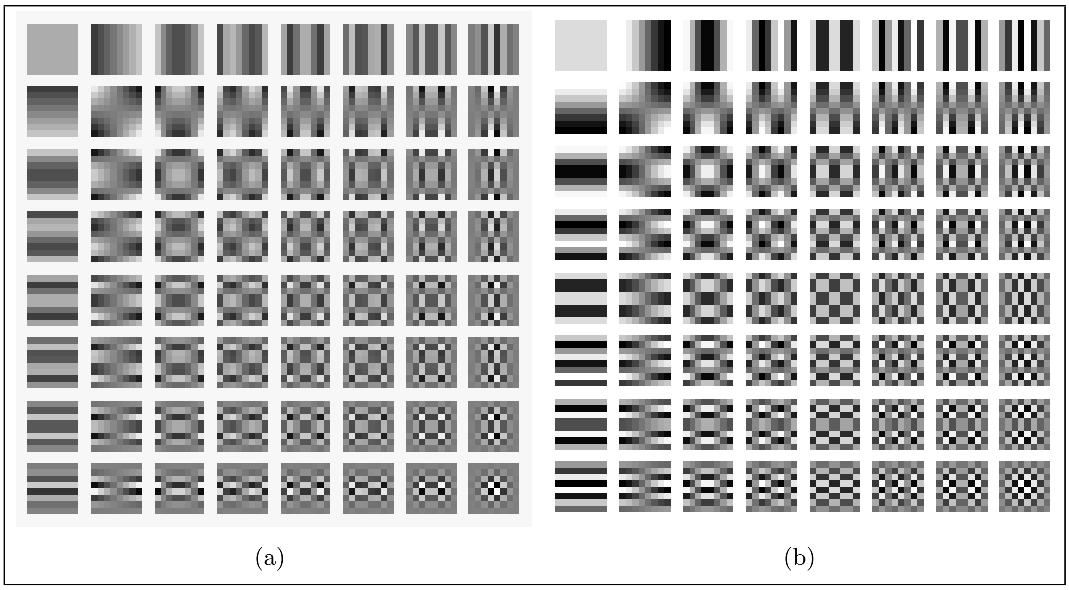


Figure 2.1: The 8×8 basis images of 2D (a) DTT (b) DCT.

In other words, Tchebichef transform T_{uv} measures the correlation between image I and basis image ψ_{uv}. It records a high positive value if there is a strong similarity between those. The basis image ψ_{uv} of an 8×8 image block is shown in Figure 2.1(a). It shows that when the order of the transform is increased, the basis images are changed from low spatial frequency (upper left corner of the figure) to high spatial frequency (lower right corner of the figure). This is quite similar to the basis image of 8×8 DCT which is shown in Figure 2.1(b). Figure 2.2 shows the Tchebichef polynomials $t_u(x)$ for $u = 0, ..., 7$. This conforms that there will be neither large variations in dynamic range of transformed values nor numerical instabilities that occur for large values of N.

2.3 Similar Properties between DTT and DCT

The definition of DTT can be written in separable form as:

$$T_{uv} = \sum_{x=0}^{N-1} t_u(x) \sum_{y=0}^{M-1} t_v(y) f(x,y) \tag{2.16}$$

Therefore it can be evaluated using two dimensional transform as follows:

$$g_v(x) = \sum_{y=0}^{M-1} t_v(y) f(x,y) \tag{2.17}$$

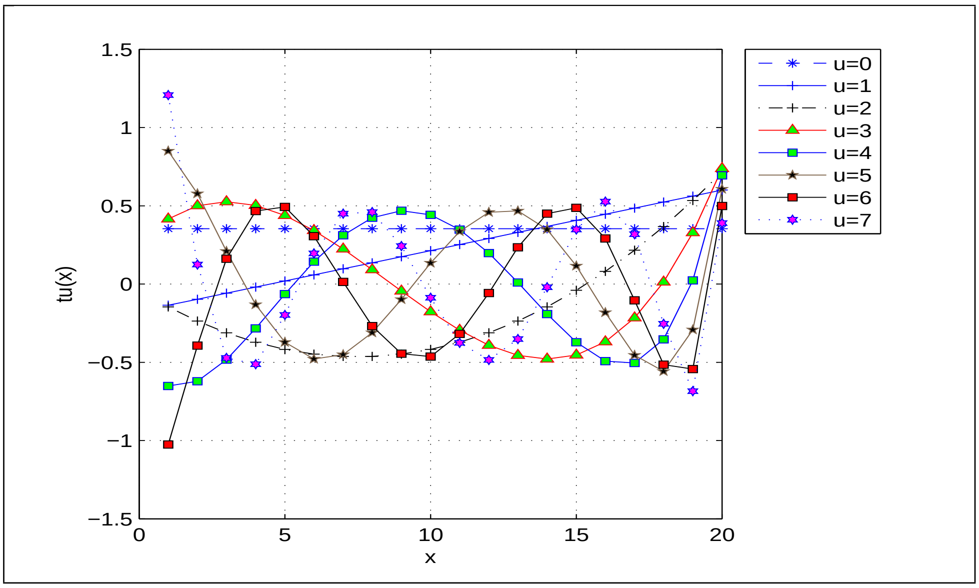


Figure 2.2: Plot of Tchebichef polynomials for $N = 20$

$$T_{uv} = \sum_{x=0}^{N-1} t_u(x) g_v(x) \tag{2.18}$$

The transform equation of DCT can be expressed as:

$$\begin{aligned} C_{uv} &= \alpha(u)\alpha(v) \sum_{x=0}^{N-1} \cos\left[\frac{\pi(2x+1)u}{2N}\right] \sum_{y=0}^{M-1} f(x,y) \cos\left[\frac{\pi(2y+1)v}{2N}\right] \\ \text{where, } \alpha(u)\alpha(v) &= \begin{cases} \sqrt{1/N}, & for\ u,v=0 \\ \sqrt{2/N}, & otherwise \end{cases} \end{aligned} \tag{2.19}$$

From Eqn. 2.18 and Eqn. 2.19, it is clear that 2-D DTT and 2-D DCT are just one dimensional DTT and DCT applied twice by successive 1-D operations, once in x-direction, and once in y-direction.

- Even Symmetry: From [47], it can be shown that Tchebichef polynomials satisfy the property

  $$t_u(N-1-x) = (-1)^u t_u(x), \quad u = 0, 1, ..., N-1. \tag{2.20}$$

  For DCT:

  $$C_m(n) = (-1)^m C_m(N-n-1), \quad m = 0, 1, ..., N-1. \tag{2.21}$$

  The above two properties are commonly used in transform coding methods to get substantial reduction in the number of arithmetic operations.

- Orthogonality: DTT and DCT basis functions are orthogonal. Thus, the inverse

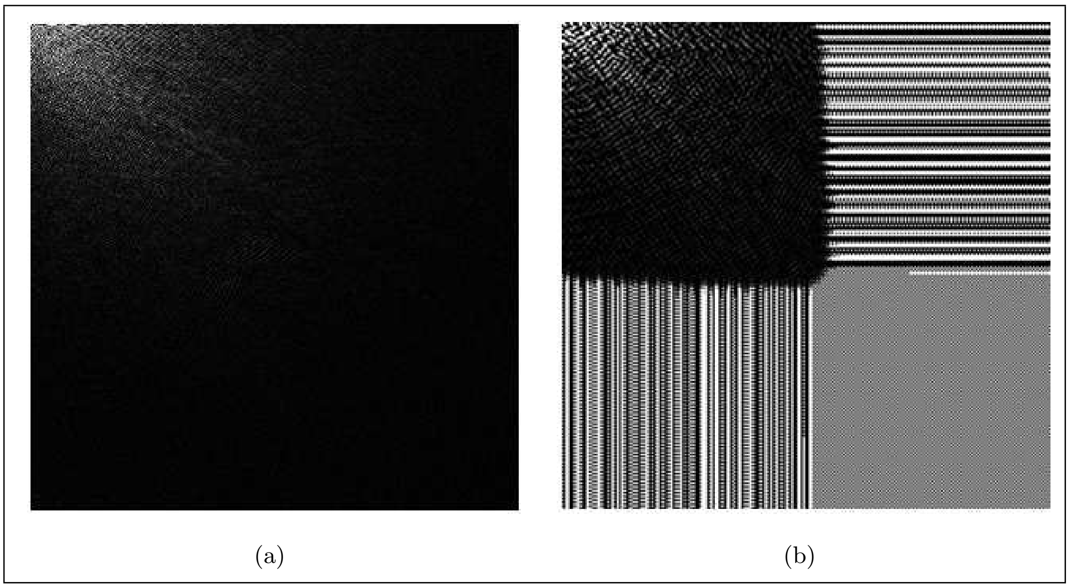

(a) (b)

Figure 2.3: The Energy Compaction Property on a 256 × 256 image using (a) DCT, and (b) DTT.

transformation matrix of A is equal to its transpose (i.e., A^T). Therefore, this property render some reduction in the pre-computation complexity. 2-D basis images of DTT and DCT are shown in Figure 2.1.

From Figure 2.1, it is clear that the basis images of DTT and DCT are quite similar in nature. Rows in the spectrum are increased in horizontal frequencies while columns are increased in vertical frequencies. For both images low frequencies resides in the upper part of spectrum.

- Energy Compaction: Efficiency of a transformation scheme can be judged by its ability to pack input energy into as few coefficients as possible. Further, the quantizer discard coefficients with relatively small amplitudes without introducing visual distortion in the reconstructed image. DTT and DCT exhibit excellent energy compaction properties for highly correlated images. The energy of the image is packed into low frequency region i.e. top left region as shown in Figure 2.3. It is observed from Figure 2.3(b) that the basis function of DTT collapse for large image sizes. This is due to the numerical error that propagates through the calculation using Eqn. 2.7 [46]. By performing image blocks of size less than 64 × 64, the errors can be safely avoided.

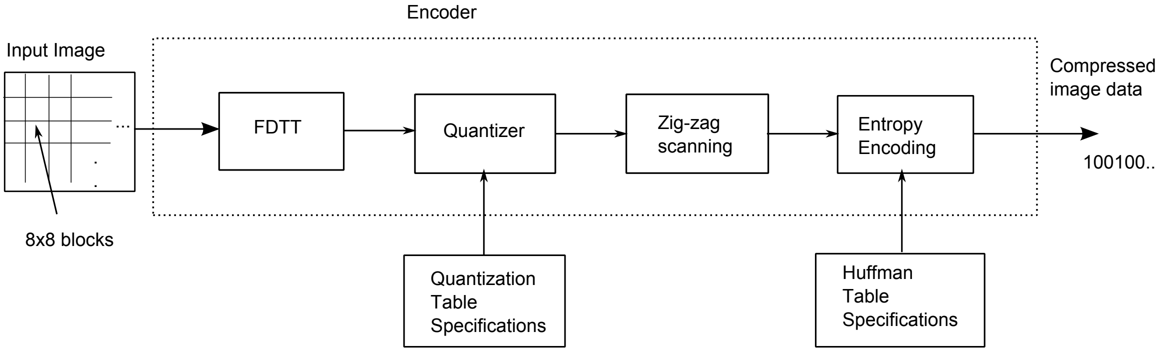


Figure 2.4: Implementation of FDTT on the 8 × 8 blocks of image data.

2.4 Application of DTT on JPEG baseline Standard

Using the JPEG Compression platform, 8 × 8 forward DTT has been used in place of DCT. Figure 2.4 shows how the FDTT is performed on the 8 × 8 blocks of image data in order to achieve good compression performance. The image is divided into 8 × 8 blocks of pixels in order to apply FDTT. The 8 × 8 blocks are processed from left-to-right and from top-to-bottom. After transformation, quantization process and entropy coding are performed.

Quantization is a process which removes the high frequencies present in the original image. This is due to the fact that the eye is much more sensitivity to lower frequencies than to higher frequencies. This is done by dividing values of high indexes in the vector (the amplitude of higher frequencies) with larger values. Values of low indexes are divided with amplitudes of lower frequencies. The standard JPEG luminance quantization table have been used in simulation.

After the transformation and quantization over an 8 × 8 image sub-blocks, the new 8 × 8 sub-block shall be reordered in zigzag scan into a linear array. The first coefficient is the DC coefficient and the other 63 coefficients are AC coefficients. Because the DC coefficient contains a lot of energy, it has usually much larger value than AC coefficients. Since there is a very close relation between the DC coefficients of adjacent blocks, the DC coefficients are differentially encoded. This process further reduces entropy.

The entropy coding process consists of Huffman coding tables as recommended in JPEG standard. Table K.3 is supplied for coding the luminance DC difference and Table K.4 is supplied for coding chrominance DC difference. Similarly, there are two Huffman tables (Table K.5 for luminance and K.6 for chrominance) for encoding the AC coefficients in Annex K of the baseline JPEG standard for reference. These tables are stored as header information during the compression process so that it is possible to uniquely decode the coefficients during decompression process.

2.4.1 Simulation Results

To evaluate the performance of DTT, simulations are conducted on various kinds of monochrome images such as Fingerprint image of (798×958) size; Lena, slope and Resolution chart of (256×256) sizes; Ruler, Mandrill, Barbara and Numbers images of (512×512) sizes. The performance of DTT is also performed on color images such as Lena and f16, each of (512×512) sizes. All the simulations are performed on a MATLAB software running on Windows XP platform.

2.4.1.1 Coding Performance for Grayscale Images

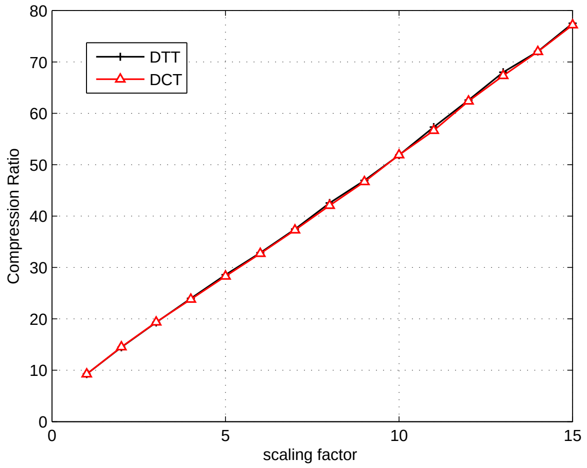
Figure 2.5 shows the compression performance of Lena, Mandrill and ruler images by varying the scaling factor of the quantization matrix from 1 to 15. It is observed that DTT has similar or higher compression performance than DCT in most of the images.

The PSNR performance of Figure 2.6 - Figure 2.9 are plotted by varying the scale factor from 1 to 15. It is observed that the PSNR/ rate-distortion performance of DTT is better than ($\sim$ 0.2 dB) DCT in Fingerprint and Slope images shown in Figure 2.6 (a) and (b) respectively. In smooth image such as Lena, DCT outperforms DTT, which is shown in Figure 2.7 (a). The performance of DTT follows closely with DCT on lower compression ratios, while DCT outperforms DTT at higher compression ratios in resolution chart image shown in Figure 2.7 (b).

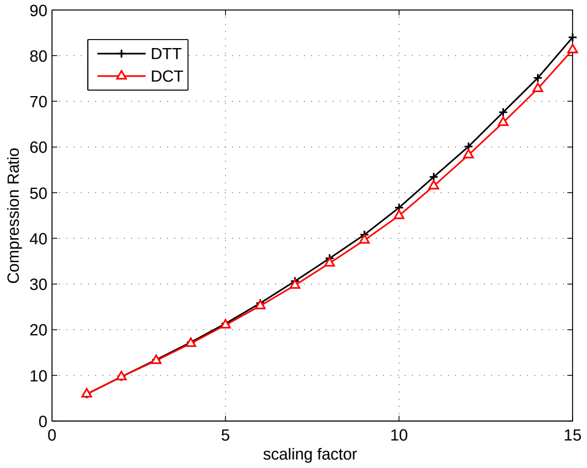
Figure 2.10 shows the visual quality of Lena image decoded at a scale factor of 5 using DCT and DTT on a JPEG baseline standard. It is observed that DCT outperform DTT in Lena image.

DTT exhibit a significant PSNR performance (typically, 1.5-2.5 dB) over all the compression ratios in Ruler image shown in Figure 2.8(a). This can be analyzed by observing Figure 2.11 that the decoded image using DTT shows less artifacts in between the black lines of Ruler image. Numbers image consists of a lot of edge details and smooth areas. Therefore, DTT follows closely or even better than DCT. DCT performs closely and even better than DTT in texture images such as Barbara and Mandrill images of Figure 2.9(a) and (b) respectively. The decoded image of Mandrill using DCT (shown in Figure 2.12) has 0.2386 dB of PSNR improvement at a scale factor of 5 compared to DTT decoded image.

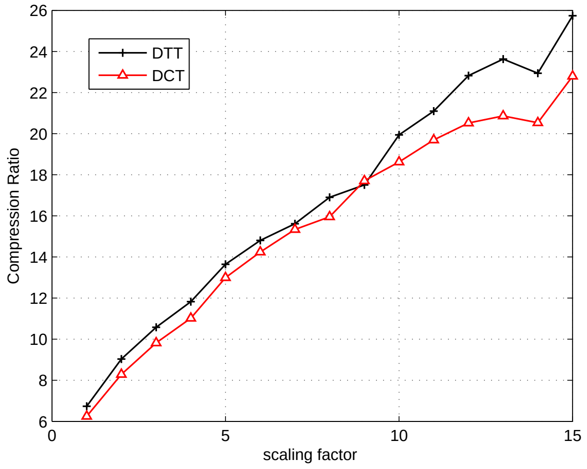
Therefore, it concluded that DTT exhibit better coding performance (PSNR and MSSIM) in Ruler, slope, Fingerprint and Numbers images, while the coding performance lowers in Lena and Mandrill and Barbara images. However, DTT has similar



(a)



(b)



(c)

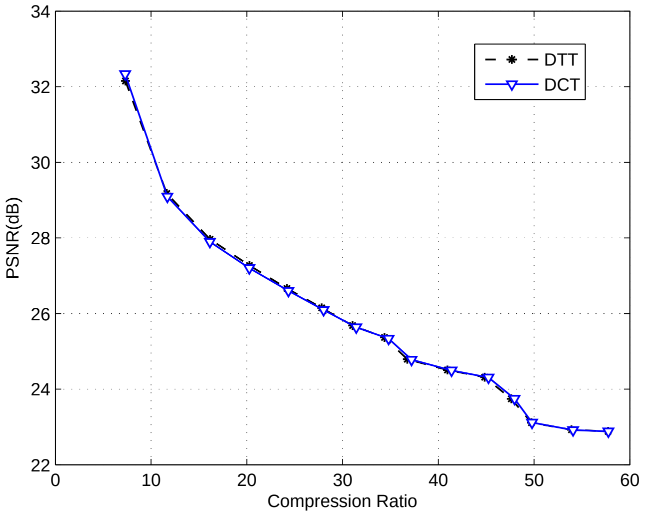
Figure 2.5: The Compression ratio vs scaling factor plots between DCT and DTT on a JPEG baseline codec (a) Lena, (b) Mandrill, and (c) Ruler images.

or better compression capability than DCT. In other words, the average number of Huffman encoded bits using DTT is lesser than DCT.

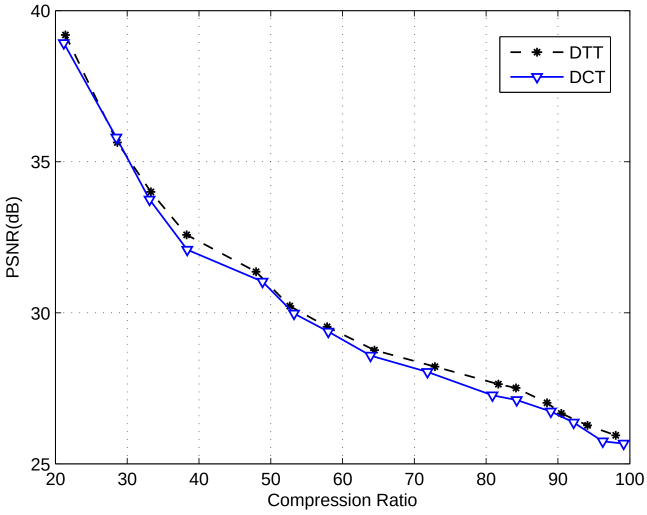
2.4.1.2 Coding Performance for Color Images

Additional processing steps such as color conversion and chroma sub sampling are used at the front end and back end in the proposed system (like in Figure 1.3) while compressing color images. 4:2:0 chroma sub sampling is selected for evaluation in this experiment.

It is observed from Figure 2.13(a) that the rate-distortion performance of DTT in f16 image follows closely with DCT. There is a slight decrease in performance of DTT is observed at lower compression ratios in Lena image of Figure 2.13(b). The MSSIM performance between DCT and DTT is very closely related in both images. Figure 2.14 and Figure 2.15 show the decoded visual quality of f16 and Lena images respectively as the scale factor is varied from 1 to 5. This indicate that images decoded using DTT exhibit similar visual performance as with images decoded using DCT.

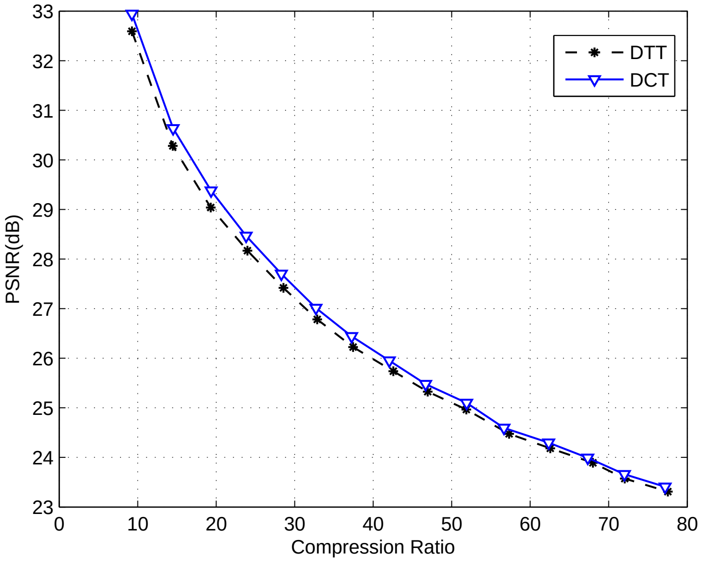


(a)

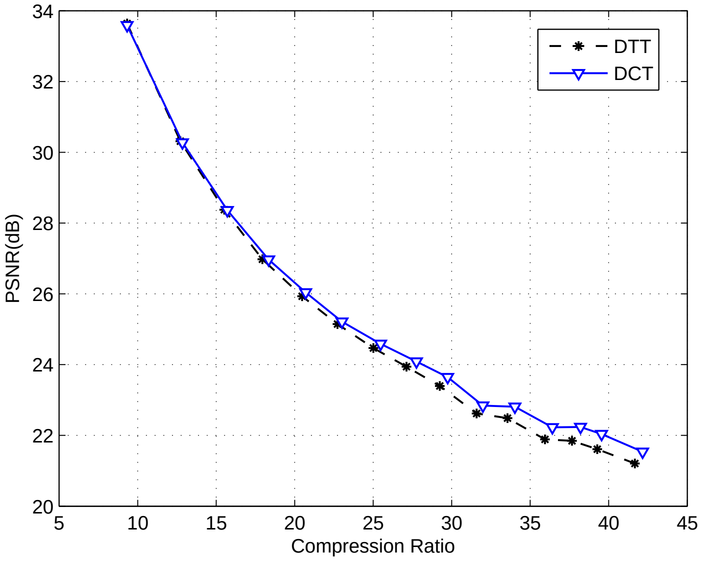


(b)

Figure 2.6: The rate-distortion comparison plots between DCT and DTT on a JPEG baseline codec (a) Fingerprint, and (b) Slope images.

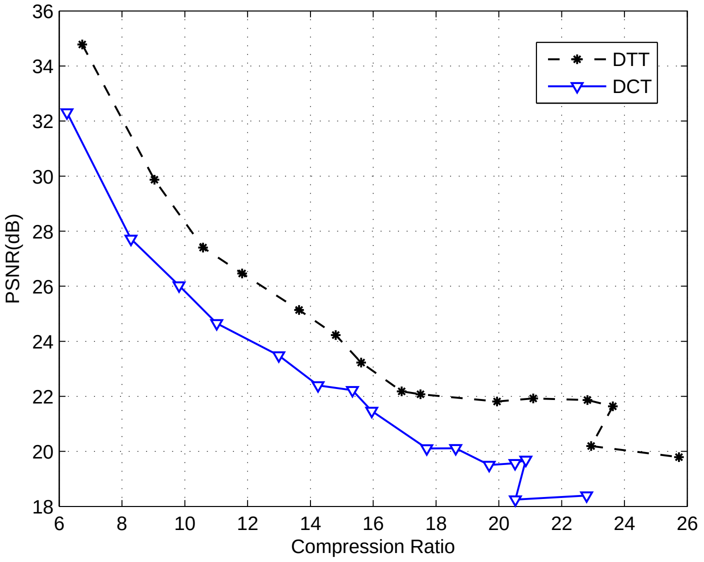


(a)

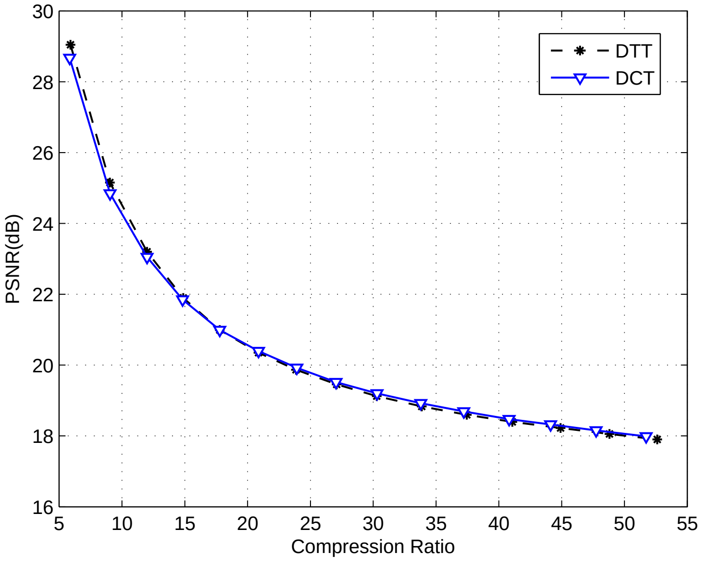


(b)

Figure 2.7: The rate-distortion comparison plots between DCT and DTT on a JPEG baseline codec (a) Lena, and (d) Resolution chart images.

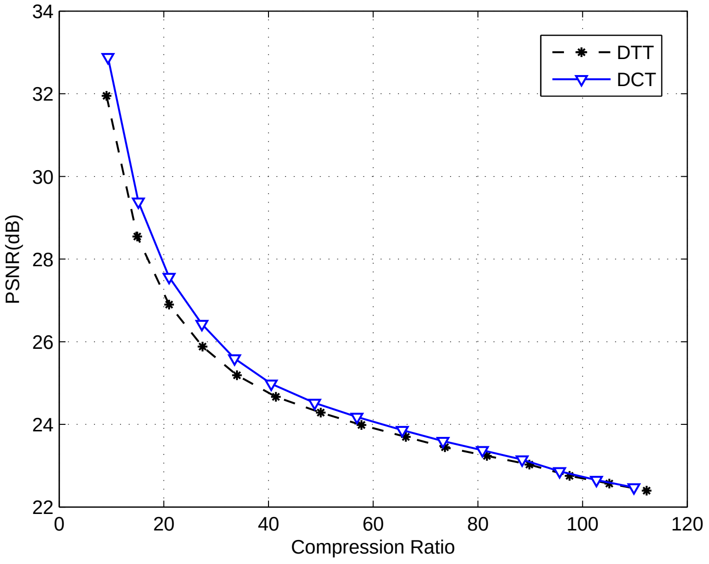


(a)

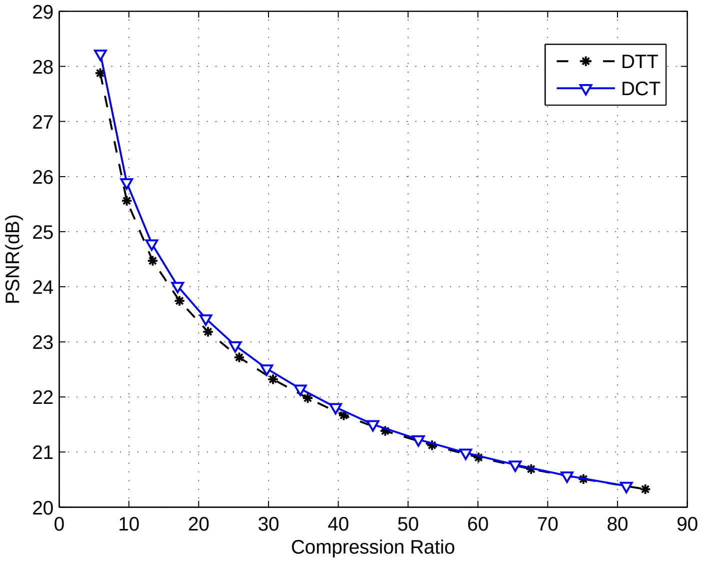


(b)

Figure 2.8: The rate-distortion comparison plots between DCT and DTT on a JPEG baseline codec (a) Ruler, and (b) Numbers images.

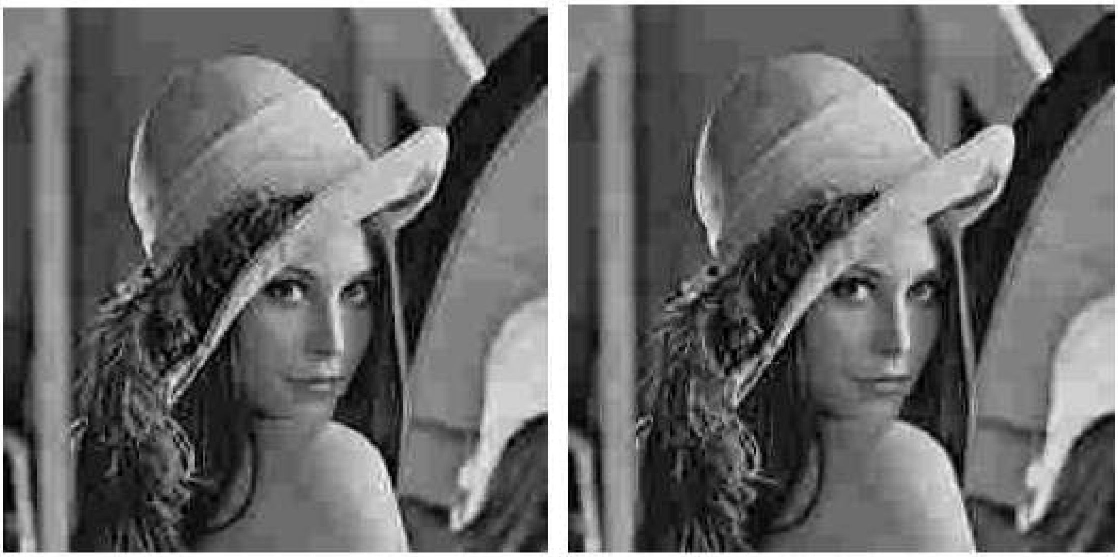


(a)



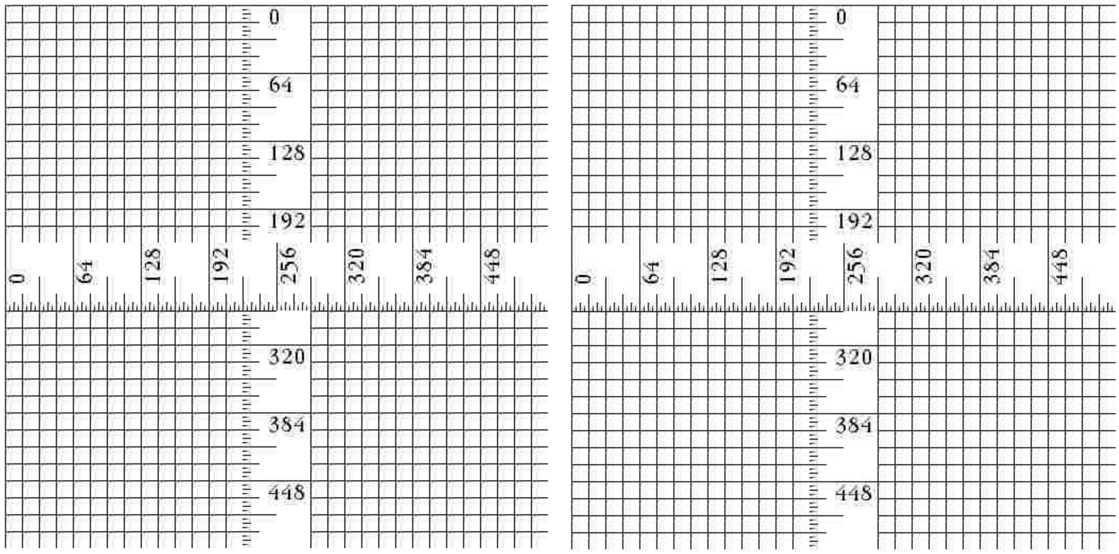
(b)

Figure 2.9: The rate-distortion comparison plots between DCT and DTT on a JPEG baseline codec (a) Barbara, and (b) Mandrill images.



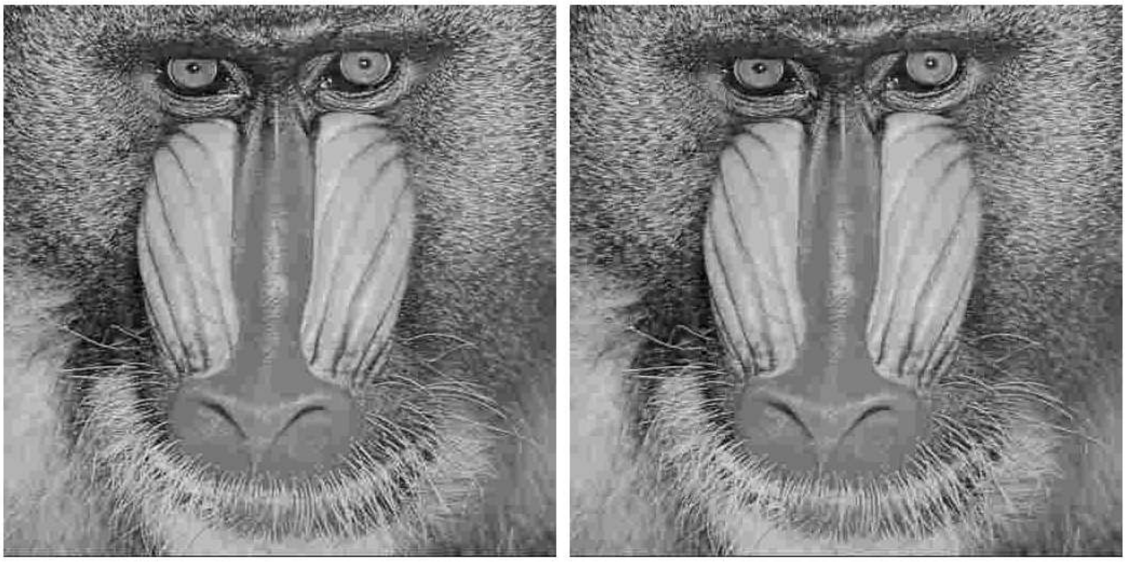
(a) (b)

Figure 2.10: The reconstructed Lena images at a scale factor of 5 using (a) DCT (PSNR=27.6998, MSSIM=0.8011) (b) DTT (PSNR=27.4198, MSSIM=0.7930).



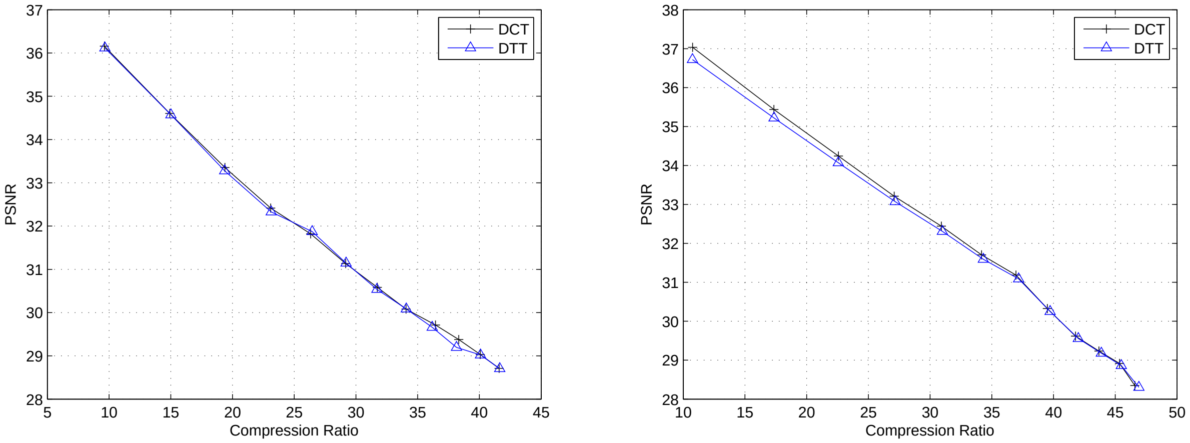
(a) (b)

Figure 2.11: The reconstructed Ruler images at a scale factor of 5 using (a) DCT (PSNR=23.4805, MSSIM=0.9890) (b) DTT (PSNR=25.1387, MSSIM=0.9904).



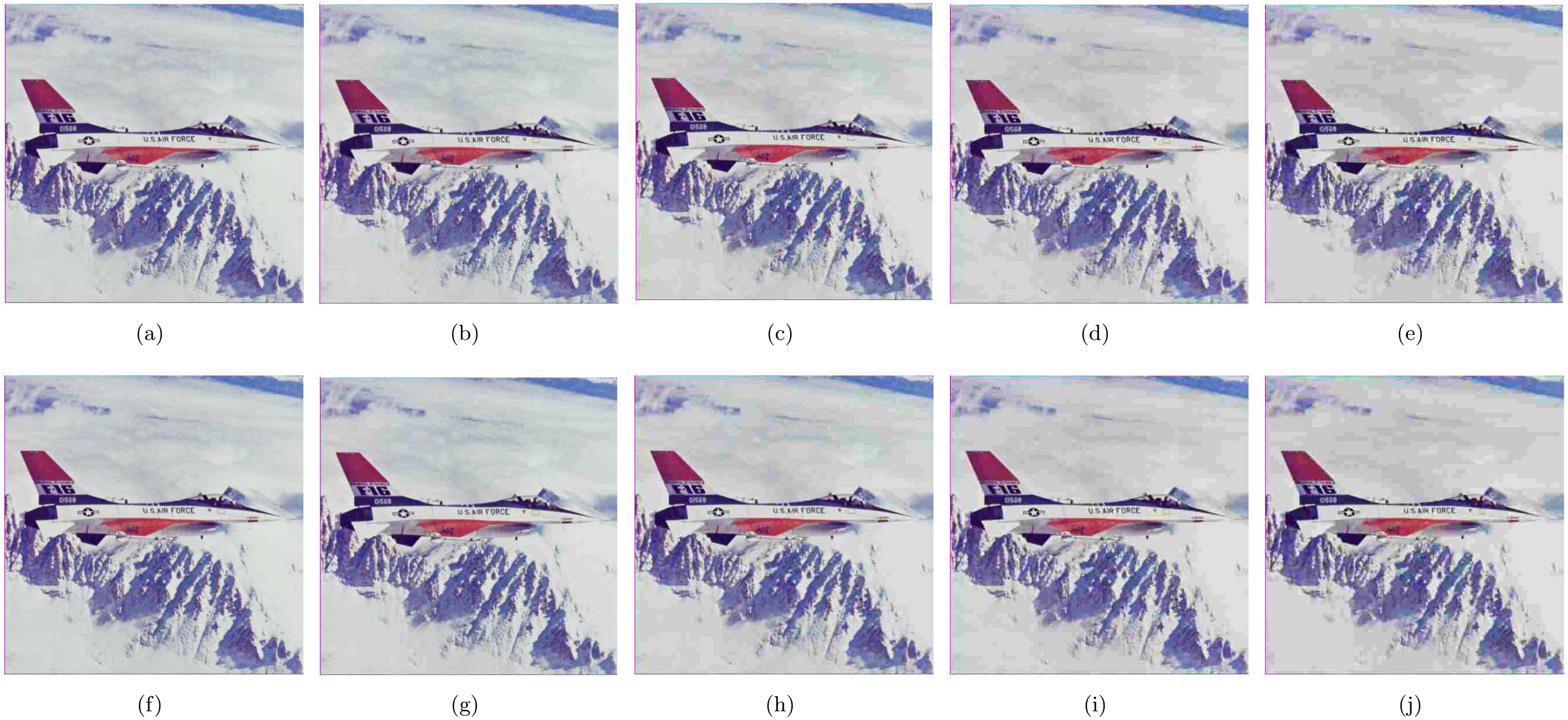
(a) (b)

Figure 2.12: The reconstructed Lena images at a scale factor of 5 using (a) DCT (PSNR=23.4209, MSSIM=0.8708) (b) DTT (PSNR=23.1823, MSSIM=0.8582).



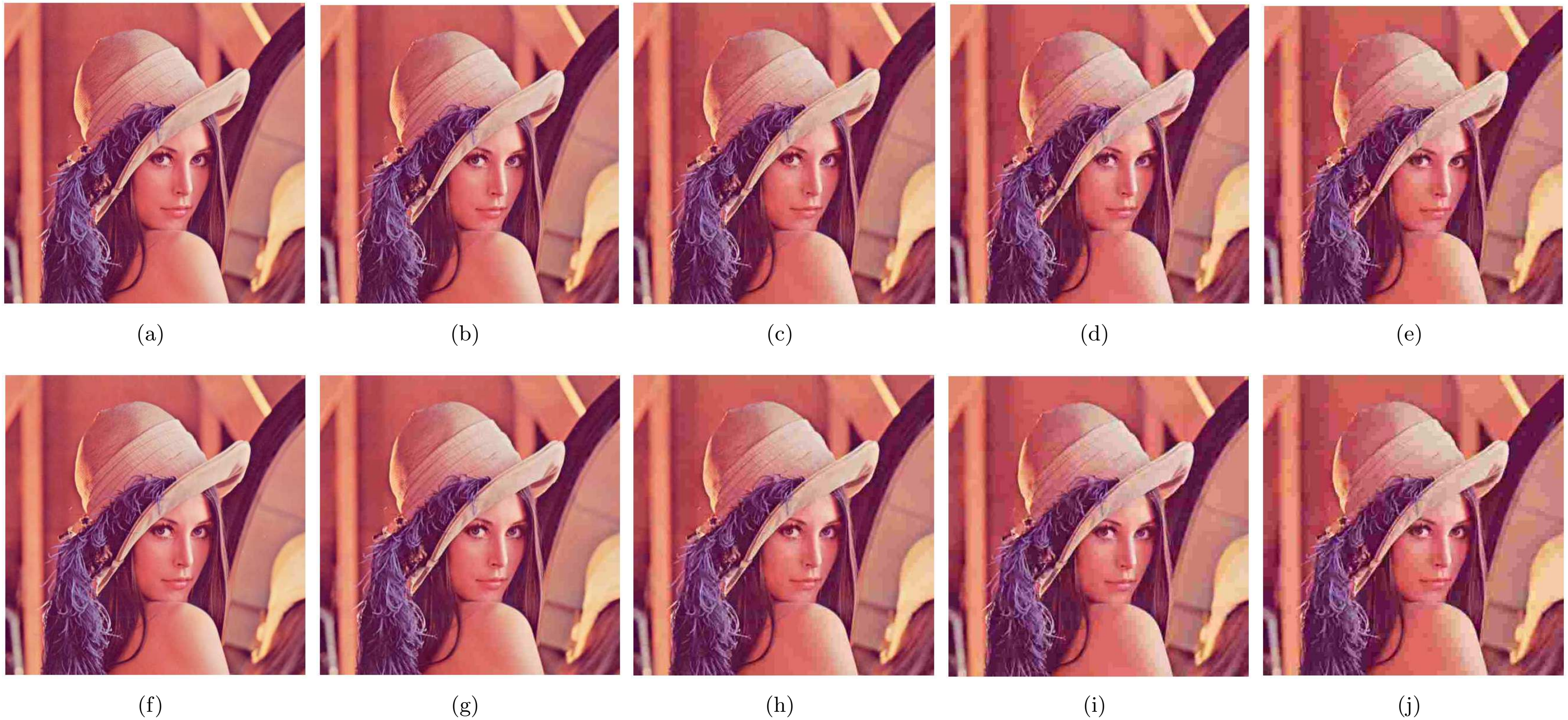
(a) (b)

Figure 2.13: The rate-distortion comparison plots between DCT and DTT on a JPEG baseline codec (a) f16 and (b) Lena color images.


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

(f) (g) (h) (i) (j)

Figure 2.14: Visual quality comparison of the decoded images between DCT(e.g., (a)-(e)) and DTT(e.g., (f)-(j)) on a JPEG baseline codec by varying the scale factor from 1 to 5: (a)CR=9.6289, MSSIM=0.9331, (b)CR=14.9181, MSSIM=0.9063 (c)CR=19.3893, MSSIM=0.8772, (d)CR=23.0964, MSSIM=0.8578, (e)CR=26.3267, MSSIM=0.8398.; (f)CR=9.6680, MSSIM=0.9307, (g)CR=15.0113, MSSIM=0.9019, (h)CR=19.3626, MSSIM=0.8728, (i)CR=23.1402, MSSIM=0.8538, (j)CR=26.4556, MSSIM=0.8379.


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

(f) (g) (h) (i) (j)

Figure 2.15: Visual quality comparison of the decoded images between DCT(e.g., (a)-(e)) and DTT(e.g., (f)-(j)) on a JPEG baseline codec by varying the scale factor from 1 to 5: (a)CR=10.7668, MSSIM=0.9247, (b)CR=17.3498, MSSIM=0.8888 (c)CR=22.5746, MSSIM=0.8550, (d)CR=27.0908, MSSIM=0.8250, (e)CR=30.9071, MSSIM=0.8013.; (f)CR=10.7577, MSSIM=0.9209, (g)CR=17.3623, MSSIM=0.8865, (h)CR=22.5636, MSSIM=0.8535, (i)CR=27.1774, MSSIM=0.8239, (j)CR=30.9973, MSSIM=0.7997.

2.5 Application of DTT on SPIHT Embedded Coder

In this section, the application of DTT on SPIHT embedded coder is discussed. The quality of reconstructed images is analysed both subjectively and objectively.

2.5.1 The Proposed DTT_SPIHT Embedded Coder

The proposed DTT_SPIHT embedded coder is shown in Figure 2.16. The input image is divided into non-overlapping 8 × 8 blocks. Each block is transformed using DTT. The coefficients are arranged into 3 level wavelet pyramid structure. The coefficients are quantized by SPIHT coding algorithm. The next stage is to use entropy coding which will give additional compression to the bit stream. For fair comparison, the back end entropy coding is not employed into the proposed algorithm.

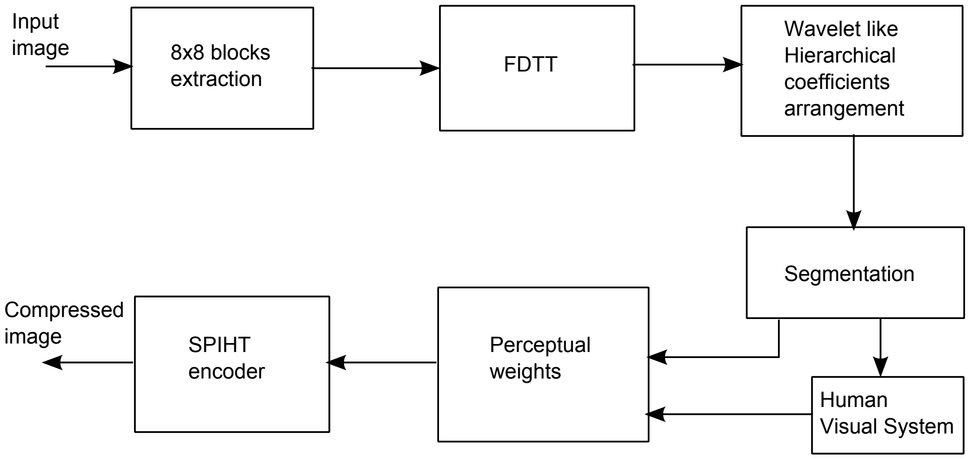


Figure 2.16: Block diagram of HVS based DTT_SPIHT embedded image coder

2.5.1.1 Rearrangement Algorithm of Transformed Coefficients

Figure 2.17 shows the arrangement of 8 × 8 DTT coefficients in a 3-level wavelet pyramid structure. After labeling 64 coefficients in each block, the parent child relationship is defined as follows: The parent of coefficient i is $\lfloor \frac{i}{4} \rfloor$ for $1 \leq i \leq 63$, while the set of four children associated with coefficient j is $\{4j, 4j+1, 4j+2, 4j+3\}$ for $1 \leq j \leq 15$. The DC coefficient 0 is the root of DTT coefficients tree, which has only three children: coefficients 1,2 and 3. In the proposed structure, offsprings corresponds to direct descendants in the same spatial location in the next finer band of the pyramid. A tree corresponds to a node having 4 children which always form a group of 2 × 2 adjacent pixels. In Figure 2.16, arrows indicate that the same index coefficients of other 8 × 8 blocks are grouped together so that the entire image can form an overall 3-level pyramid structure.

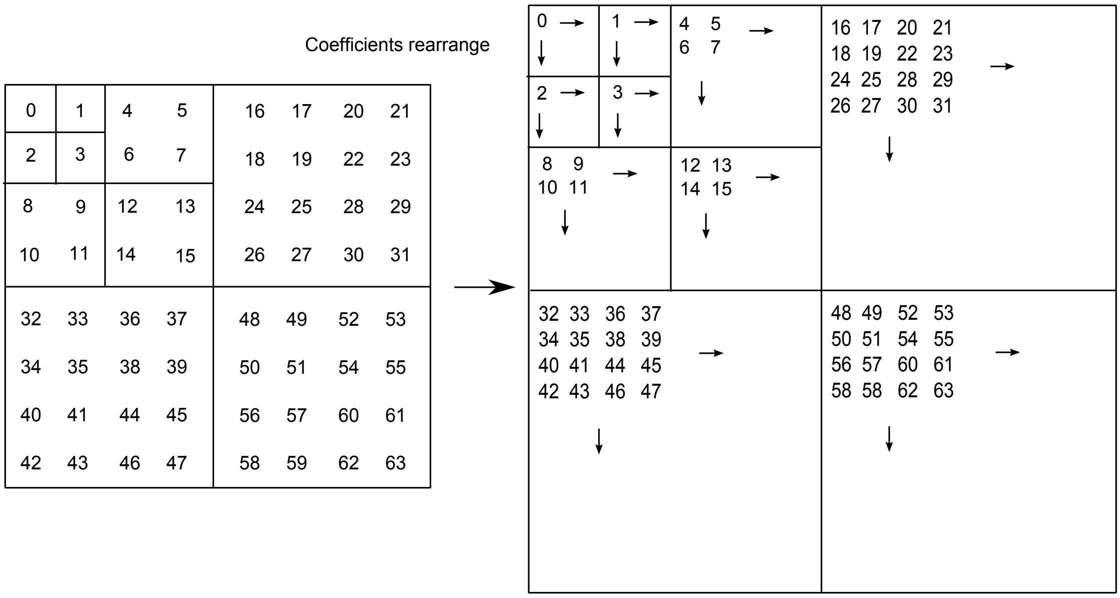


Figure 2.17: Rearrangement algorithm of 8 × 8 transformed coefficients.

Table 2.1: Perceptual weights applied to high frequency sub-band at coarsest scale

Image block	Texture image	Edge image	Smooth image
Texture	0.5	1	1
Edge	2	2	2
Smooth	1.5	1.5	1.8

In the proposed decomposition method, further decomposition of LL_3 band into a 3-level pyramid is performed so that the coarsest level will be a 8 × 8 band. The overall level of decomposition is six. Then, SPIHT encoding algorithm is applied to the overall structure.

2.5.1.2 Human Visual System

In the proposed coder, different perceptual weights have been added across the sub-bands. It is found from the research in vision psychophysics that sensitivity of human eyes to the distortion reduces in order from edge block, smooth block and texture block. Different sensitivities suggest that different perceptual weights should be assigned to different blocks [82]. These perceptual weights decreases from coarse to fine scale in accordance with the energy decreasing characteristics of the wavelet coefficients. On the other hand, this ensures most significant coefficients transferred with highest priority. Therefore, it can improve the reconstructed image quality. Table 2.1 shows the perceptual weights applied to high frequency sub-bands at the coarsest scale.

The segmentation process determines the type of image block. Entropy and variance values of different blocks play an important role during segmentation. The entropy value of smooth block is smaller than the edge and textured block. The variance of textured block is smaller than edge and smooth block.

2.5.1.3 SPIHT Algorithm

SPIHT algorithm [16] keeps track of the state of sets by means of three lists, i.e, list of insignificant sets (LIS), list of significant pixels (LSP) and lists of insignificant pixels (LIP). The algorithm uses the following sets to code a bitmap effectively: $O(i,j)$ is the set of coordinates of all offspring of node (i,j), $D(i,j)$ is set of coordinates of all descendants of node (i,j), $L(i,j)$ is the set of coordinates defined as $D(i,j)-O(i,j)$, and $H(i,j)$ is set of all tree roots. The significance test of a wavelet coefficient is defined as follows:

$$S_n(\Gamma) = \begin{cases} 1, \underset{(i,j\epsilon\Gamma)}{max} \mid (T(i,j) \mid \geq 2^n \\ 0, \text{otherwise.} \end{cases} \tag{2.22}$$

where, $T(i,j)$ is the coefficient at node (i,j). Γ is the testing coordinate set in Eqn. 2.22. Briefly, SPIHT coding method is described in the following two passes:

1. Initialization: LSP=ϕ. LIP contains all tree roots at coarsest scale. LIS contains all tree nodes. Choose the threshold according to (2.22).

2. Sorting pass: Output $S_n(i,j)$ for all the nodes (i,j) of LIP. If $S_n = 1$, move the node (i,j) to the LSP and output the sign of $T(i.j)$. For all the nodes of (i,j) of LIS, if the nodes belongs to type A, output $S_n(D(i,j))$. If $S_n(D(i.j)) = 1$, output $S_n(k,l)$ for any node $(k,l)\ \epsilon\ O(i,j)$. If $S_n(k,l) = 1$, append node (k,l) to the LSP and output its sign. Otherwise, move (k,l) to the LIP. If $L(i,j) \neq \phi$, then move (i,j) from LIS, while marking as type B. Otherwise, remove the node (i,j) from LIS. If the node belongs to type B, then output $S_n(L(i,j))$. If $S_n(L(i,j)) = 1$, append the four direct subsequent nodes to the LIS as type A.

3. Refinement pass: For each elements (i,j) in the LSP, output the n-th most significant bit except those added above.

4. Update: Decrement n by 1 and go to sorting pass.

2.5.2 Simulation Results and Analysis

To evaluate the performance of the proposed hybrid image coding algorithm, experiments are conducted on Lena, Barbara, 256 Level Test Pattern, Ruler and Numbers images. The size of each image is 512 × 512. The first image is a smooth image, second one is a texture image, third, fourth and fifth are images having sharp edges.

Table 2.2 shows a comparison of proposed DTT_SPIHT algorithm with some of the best known algorithms in the literature. In comparison to Improved JPEG,

Table 2.2: PSNR(dB) Comparison of DTT_SPIHT With Other Algorithms

Rate(b/p)	JPEG		Improved JPEG		EZDCT		STQ		STQ+Haar		DCT_SPIHT		DTT_SPIHT	
Image	Barbara	Lena	Barbara	Lena	Barbara	Lena	Barbara	Lena	Barbara	Lena	Barbara	Lena	Barbara	Lena
0.25	25.2	31.6	26.0	31.9	25.4	30.7	26.6	31.2	26.8	32.3	26.9	31.8	26.5	31.6
0.50	28.3	34.9	30.1	35.5	29.4	34.8	30.1	35.4	30.7	35.6	30.6	35.6	29.8	35.3
0.75	31.0	36.6	33.0	37.5	32.5	37.1	32.7	37.0	33.4	37.2	33.5	37.7	32.7	37.5
1.00	33.1	37.9	35.2	38.8	34.9	38.7	35.4	38.8	35.6	39.0	36.1	39.3	35.3	39.2

DTT_SPIHT shows a PSNR reduction of 0.2 dB for bit-rates 0.5 and 0.3 dB for bit-rate 0.25 bpp on Lena image. The PSNR values of DTT_SPIHT is high at bit rate above 0.75 bpp. For Barbara image, DTT_SPIHT shows a PSNR reduction of 0.3 dB between 0.5-0.75 bpp. The advantage of the proposed method is its simplicity in comparison to improved JPEG. Entropy encoding and decoding processes are not used by our algorithm. By incorporating back end arithmetic coding stage in the our coder, additional 5-10 % compression can be achieved.

Comparing with EZDCT algorithm, the proposed DTT_SPIHT shows a PSNR gain up to 1.0 dB at low bit-rates in Lena and Barbara images. For Barbara image, STQ algorithm shows a PSNR gain of almost 0.1 dB over 0.25 to 1 bit-rates. DTT_SPIHT outperforms STQ on all bit rates in case of Barbara image. STQ+Haar shows a maximum of 0.9 dB gain over DTT_SPIHT on Barbara image. For bit-rates between 0.5 and 1 bpp DTT_SPIHT shows a good PSNR gain over STQ+Haar algorithm on Lena image. DCT_SPIHT always outperform the proposed DTT_SPIHT algorithm for Lena and Barbara images.

Table 2.3: Comparison of PSNR(dB) Values of DCT_SPIHT With DTT_SPIHT

Rate(b/p)	DCT_SPIHT			DTT_SPIHT		
	256 Level-Test Pattern	Ruler	Numbers	256 Level-Test Pattern	Ruler	Numbers
0.125	17.3	13.0	17.4	17.3	13.8	17.4
0.25	19.1	16.3	19.3	19.0	19.3	19.1
0.50	21.8	22.9	22.1	21.8	23.1	22.1
0.75	24.4	26.5	25.0	24.5	27.4	25.1
1.00	26.6	28.6	27.7	26.8	31.0	27.8

Table 2.4: Comparison of PSNR(dB) and MSSIM Values of DTT_SPIHT With HVS and Without HVS on Lena Image

Bit-rate in bpp	DTT_SPIHT without HVS		DTT_SPIHT with HVS	
	PSNR	MSSIM	PSNR	MSSIM
0.125	28.5	0.8432	28.4	0.8552
0.25	31.6	0.9172	31.5	0.9238
0.50	35.3	0.9637	35.0	0.9655
0.75	37.5	0.9785	37.2	0.9737
1.00	39.3	0.9856	38.7	0.9861

The performance of propose DTT_SPIHT algorithm can be well judged form Table 2.3, which shows a PSNR comparison between DCT_SPIHT and DTT_SPIHT on 256 Level Test Pattern, Ruler and Numbers images. It is observed that DTT_SPIHT shows a gain of 0.1 to 0.2 dB on 256 Level Test Pattern image in all the considered bit rates. On Ruler image, the proposed algorithm outperforms DCT_SPIHT at any

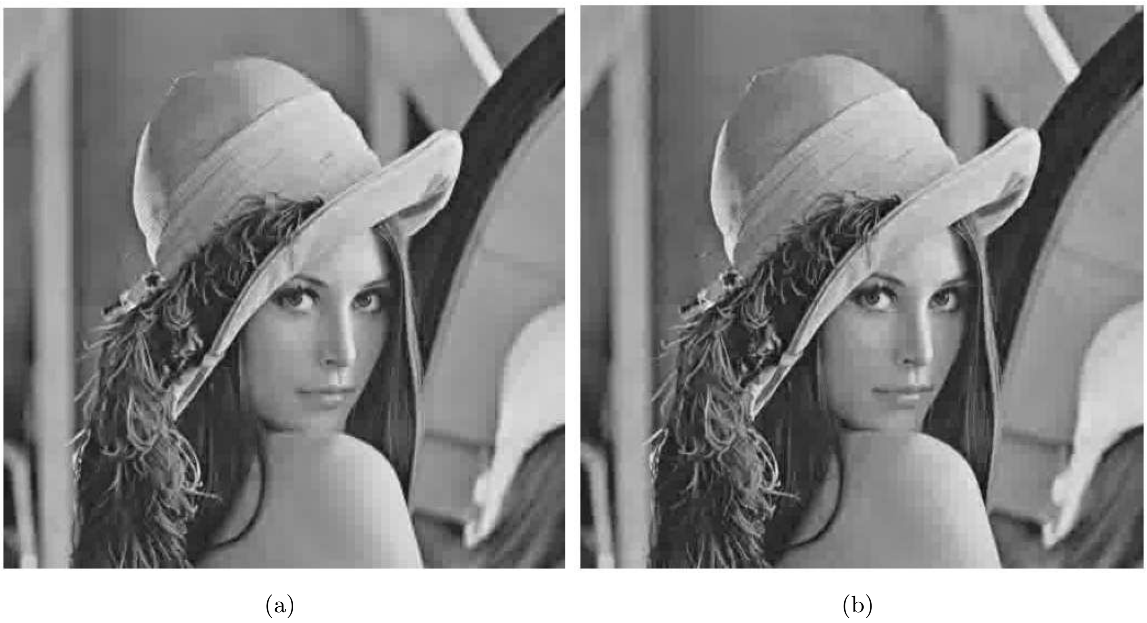

(a) (b)

Figure 2.18: The reconstructed images of Lena using HVS based DTT_SPIHT at a bit-rate of 0.25 bpp (a) HVS on (PSNR=31.5, MSSIM=0.9236) (b) HVS off (PSNR=31.6, MSSIM=0.9172).

bit-rate significantly (e.g., 3.0 dB at 0.25 bpp). Maximum PSNR improvement in Numbers image is 0.1 dB on the considered bit rates except at 0.25 bpp.

Table 2.4 shows the comparison of hybrid DTT_SPIHT with HVS is on and off. It is demonstrated that when HVS is on, it reduces the objective quality of the image between 0.1 to 0.3 dB at lower bit-rates (0.125 to 0.5 bpp) for Lena image. Figure 2.16 (a) and (b) shows the reconstructed Lena image at a bit-rate of 0.25 bpp while HVS is on and HVS is off respectively. It has been observed that the shoulder and facial portion of Lena image in Figure 2.18(a) is smoother than in Figure 2.18(b). This is obvious by comparing the MSSIM performances listed out in the same table. This indicates that a significant improvement in the subjective quality of decoded Lena image at lower bit rates compared to higher bit rates is observed. Similar perceptual improvement is also observed for other images.

2.6 Development of 4×4 Zigzag Pruning DTT Algorithm

Although the result analysis discussed in the previous Section 2.4 uses 8 × 8 DTT, the following analysis deals with 4 × 4 DTT. Because, small block size DTT needs lower computation; which is highly desirable for low power computing devices.

2.6.1 Proposed Zigzag Prune 4×4 DTT Algorithm

The 2-D DTT from Eqn. 2.11 can be expressed in matrix form as:

$$T == \tau F \tau'. \tag{2.23}$$

where F is the 2-D input data, τ is the Tchebichef basis and T is the 2-D matrix of transformed coefficients. The transform kernel for 4 point DTT can be defined from Eqn. 2.11 as:

$$\tau = \begin{bmatrix} 1/2 & 1/2 & 1/2 & 1/2 \\ -3/2\sqrt{5} & -1/2\sqrt{5} & 1/2\sqrt{5} & 3/2\sqrt{5} \\ 1/2 & -1/2 & -1/2 & 1/2 \\ -1/2\sqrt{5} & 3/2\sqrt{5} & -3/2\sqrt{5} & 1/2\sqrt{5} \end{bmatrix} \tag{2.24}$$

By defining $x = 1/2$ and y=$1/\sqrt{5}$, Eqn. 2.23 can be written as:

$$\tau = \begin{bmatrix} x & x & x & x \\ -3xy & -xy & xy & 3xy \\ x & -x & -x & x \\ -xy & 3xy & -3xy & xy \end{bmatrix} \tag{2.25}$$

Factorizing τ in Eqn. 2.25, we can have

$$S = \begin{bmatrix} x & x & x & x \\ xy & xy & xy & xy \\ x & x & x & x \\ xy & xy & xy & xy \end{bmatrix}, \ \hat{\tau} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ -3 & -1 & 1 & 3 \\ 1 & -1 & -1 & 1 \\ -1 & 3 & 3 & 1 \end{bmatrix} \tag{2.26}$$

S is a scaling matrix and can be separated from the core transform computation. The expression in Eqn. 2.23 can be factorized as

$$T = (\hat{\tau} F \hat{\tau}') \odot \hat{S}, \quad where, \quad \hat{S} = \begin{bmatrix} x^2 & x^2y & x^2 & x^2y \\ x^2y & x^2y^2 & x^2y & x^2y^2 \\ x^2 & x^2y & x^2 & x^2y \\ x^2y & x^2y^2 & x^2y & x^2y^2 \end{bmatrix} \text{and } \hat{T} = \hat{\tau} F \hat{\tau}'. \tag{2.27}$$

Symbol $\odot$ indicates element-by-element multiplication. Since $\hat{\tau}$ is orthogonal, but not orthonormal, normalization can be done by merging $\hat{S}$ into the quantization matrix.

By substituting Eqn. 2.24 in Eqn. 2.23, each transformed coefficients can be calculated for the input matrix F. Furthermore, the even symmetry property allows to group terms of the form $f(x, 1) \pm f(x, 3)$ for $x = 0, 1, 2, 3$ to further reduce the number of arithmetic operations. The coefficients are selected in a zigzag pruned way and the computational complexity is compared with that of equal number of block pruned coefficients as specified in [87]. For the specific case, the nine zigzag pruned coefficients are compared with nine block pruned coefficients. Starting from upper left coefficients, the normalized nine zigzag pruned coefficients, $\hat{T}_{ij}$'s from Eqn. 2.27 are given as:

$$\begin{aligned}
\hat{T}_{00} &= [(A+C)+(E+G)+(I+K)+(M+O)],\\
\hat{T}_{01} &= [\{(3B+D)+(3F+H)\}+\{(3J+L)+(3N+P)\}],\\
\hat{T}_{10} &= [3\{(E+G)-(A+C)\}+(M+O)-(I+K)],\\
\hat{T}_{20} &= [(A+C)+(E+G)-\{(I+K)+(M+O)\}],\\
\hat{T}_{11} &= [3(3F+H)-3(3B+D)+(3N+P)-(3J+L)],\\
\hat{T}_{02} &= [(A-C)+(E-G)+(I-K)+(M-O)],\\
\hat{T}_{03} &= [(B-3D)+(F-3H)+(J-3L)+(N-3P)],\\
\hat{T}_{12} &= [3\{-(A-C)+(E-G)\}-(I-K)+(M-O)],\\
\hat{T}_{21} &= [\{(3B+D)+(3F+H)\}-\{(3J+L)+(3N+P)\}],
\end{aligned} \tag{2.28}$$

where

$$\begin{aligned}
&A = f(0,3)+f(0,0), B = f(0,3)-f(0,0),\\
&C = f(0,2)+f(0,1), D = f(0,2)-f(0,1),\\
&E = f(3,3)+f(3,0), F = f(3,3)-f(3,0),\\
&G = f(3,2)+f(3,1), H = f(3,2)-f(3,1),\\
&I = f(1,3)+f(1.0), J = f(1,3)-f(1,0),\\
&K = f(1,2)+f(1,1), L = f(1,2)-f(1,1),\\
&M = f(2,3)+f(2,0), N = f(2,3)-f(2,0),\\
&O = f(2,2)+f(2,1), P = f(2,2)-f(2,1).
\end{aligned}$$

The nine normalized block pruned coefficients are given as:

$$\hat{T}_{00},\ \hat{T}_{01},\ \hat{T}_{02},\ \hat{T}_{10},\ \tilde{T}_{11},\ \hat{T}_{12},\ \hat{T}_{20};\ \hat{T}_{21},\ \hat{T}_{22.} \tag{2.29}$$

The expressions for all the coefficients are same as that of zigzag pruned coefficients defined in Eqn. 2.28 except, coefficient $\hat{T}_{22}$, which is can be expressed as:

$$\hat{T}_{22} = [\{(A-C)+(E-G)\}-\{(I-K)+(M-O)\}]. \tag{2.30}$$

2.6.2 Complexity Analysis

The proposed zigzag pruned DTT algorithm is compared with the recently proposed algorithms in [51],[52],[53],[87] and the traditional separability-symmetry algorithm. For a $n \times n$ pruned block, n^2 coefficients are needed for image reconstruction. There-

Table 2.5: Computational complexity comparison between different DTT algorithms and our proposed algorithm.

No. of coefficients used for image reconstruction (pruned DTT)	Number of operations Multiplications/ Additions/ Shifts					
	Separability & Symmetry	Nakagaki & Mukundnan[51]	Ishwar2008 et. al.[52]	Abdelwaheb[53]	Block Pruned Method[87]	Proposed
1	-	-	-	-	0/15/1	0/15/0
4	-	-	-	24/48/0	2/39/7	0/39/5
9	-	-	-	-	6/66/14	0/67/11
16	64/96/0	32/60/0	0/80/16	-	12/80/20	0/80/16

fore, it is obvious that comparison should be made between $n \times n$ block pruned with n^2 zigzag pruned.

Table 2.5 shows that the proposed zigzag pruned algorithm gives lower computation complexity than other algorithms. Specifically comparing with recently proposed block pruned method in [87], zigzag prune algorithm has lower computational complexities for any pruned sizes. By using only one coefficient, (DC component) 15 additions are required to compute $\hat{T}_{00}$. For 4 zigzag pruned coefficients, 39 additions and 5 shift operations are required as compared with 2 multiplications, 39 additions and 7 shift operations in 2×2 pruned size. Similarly, for 9 zigzag pruned coefficients, the proposed algorithm needs 67 additions and 11 shift operations compared to 6 multiplications, 66 additions and 14 shift operations in 3×3 block pruned coefficients. A substantial reduction in computational complexities is achieved. This is due to the fact that the coefficients are normalized by merging the multiplication terms with the quantization matrix.

It is also clear that the complexity of the proposed algorithm is lower than that of algorithm in [53] for 2×2 pruned block. The DTT algorithm presented in [52] is a full 4×4 DTT algorithm which is having same complexity as the full 16-coefficient zigzag algorithm.

2.6.3 Hardware Utilization

The proposed algorithm is implemented on Xilinx XC2VP30 FPGA device. We developed a distributed arithmetic based approach to compute 1-D and 2-D DTT transform on Xilinx XC2VP30 platform. This is due to the fact that, DA is free from multiplications [77]. All the coefficients are determined by integer shift and addition operations. Table 2.6 shows the hardware resource utilization of 1-D floating point DTT algorithm. The number of slices and 4 input LUTs are 2% and 1.2% of the available resources. This is much higher than 1-D integer DTT.

Considering the case of 2-D DTT, it can be seen from Table 2.7 that pruned DTT does not require any flip flops (memory) in comparison to 2-D DTT which requires

Table 2.6: H/W utilization of 1D floating point and integer DTT in Xilinx XC2VP30.

		Float DTT		Integer DTT	
Resources	Available	Utilise	% Utilisation	Utilize	% Utilisation
No. of slices	13696	277	2	61	0.4
Flip Flops	27392	0	0	0	0
4 input LUTs	27392	493	1.2	112	0.4
Bonded IOBs	556	76	13	84	11

Table 2.7: H/W utilization of 2D floating point, integer and Zigzag prune DTT in Xilinx XC2VP30.

		Float DTT		Integer DTT		Zigzag prune DTT	
Resources	Available	Utilise	% Utilisation	Utilize	% Utilisation	Utilize	% Utilisation
No. of slices	13696	707	5	98	0.7	380	2.7
Flip Flops	27392	204	0.9	94	0.3	0	0
4 input LUTs	27392	1286	4	157	0.6	715	2
Bonded IOBs	556	59	10	63	11	263	47

almost 1% of the resources. This makes the transform a combinational circuit instead of a sequential one. This is a major advantage of using pruning method. Further, the number of slices and 4 input LUTs are 2.3% and 2% lesser in 9-pruned DTT than 2-D float point DTT. However, the number of bonded IOBs are 37% more in pruned DTT. By observing into Table 2.7, it is evident that using integer based transform a number of hardware resources can be saved. The pruned DTT is calculated using direct approach rather than row-column approach. The merit of direct approach in calculating transform is that it does not need any memory elements.

2.6.4 Results and Comparisons

A comparison of reconstructed image quality is made between block-pruned sizes of 1×1, 2×2 and 3×3 with that of 1,4 and 9 zigzag pruned coefficients respectively. Table 2.8 Shows the PSNR comparison between block pruned DTT and zigzag pruned DTT. It can be observed that the PSNR of reconstructed images using 9 zigzag pruned coefficients are higher than that of 3×3 block-pruned image sizes. Comparing with 4 zigzag pruned coefficients and 2×2 block pruned coefficients, the PSNR of block pruned coefficients are higher. Nevertheless, there is advantage of computational complexities in both cases. Similarly, comparison is made between block-pruned DTT with block-pruned DCT and zigzag-pruned DTT with zigzag-pruned DCT in Table 2.9. It has seen observed that 9 zigzag pruned coefficients of DTT/DCT show always a higher PSNR than that of 3×3 block-pruned coefficients. For example, in case of Lena image, 9 zigzag pruned DTT shows a PSNR gain of 0.7 dB, Barbara shows a significant gain of 1.54 dB and Crowd image shows a PSNR gain of 1.02 dB. Similar performance improvement is also noticed in 9 zigzag pruned DCT images shown in

Table 2.10 and 2.11. Furthermore, for images such as Lena, Barbara and Crowd of Table 2.10, DCT shows slight better performance than that of DTT. For images such as Finger print, Mountain and Library of Table 2.11, DTT outperforms DCT of any pruning sizes. For instance, Finger print image shows a PSNR gain of 1.27 dB in 9-prune sizes and 0.27 dB in 4-prune sizes. For Mountain and Library images the PSNR gain is slightly higher. Hence, 9 zigzag pruned coefficients are enough for practical image or video coding applications.

Table 2.8: Comparison of PSNR between block-pruned and zigzag-pruned reconstructed images of (a) Lena, (b) Barbara and (c) Crowd.

No. of DTT coefficients retained	PSNR(dB)					
	Lena		Barbara		Crowd	
	Block prune	Zigzag prune	Block prune	Zigzag prune	Block prune	Zigzag prune
1	26.92	26.92	23.37	23.37	21.62	21.62
4	33.35	32.35	25.61	25.23	30.13	29.43
9	39.29	40.00	30.03	31.57	38.54	39.59

Table 2.9: Comparison of PSNR between block-pruned and zigzag-pruned reconstructed images of (d) Finger print, (e) Mountain, and (f) Library.

No. of DTT coefficients retained	PSNR(dB)					
	Finger print		Mountain		Library	
	Block prune	Zigzag prune	Block prune	Zigzag prune	Block prune	Zigzag prune
1	11.08	11.08	17.08	17.08	16.25	16.25
4	14.94	16.84	19.60	19.82	18.90	19.44
9	22.28	24.38	22.97	23.17	22.69	23.45

Table 2.10: Comparison of PSNR between DCT and DTT of block pruned and zigzag pruned reconstructed images of (a) Lena, (b) Barbara and (c) Crowd.

No. of coefficients retained for image reconstruction	PSNR(dB)											
	Lena				Barbara				Crowd			
	Block prune		Zigzag prune		Block prune		Zigzag prune		Block prune		Zigzag prune	
	DCT	DTT	DCT	DTT	DCT	DTT	DCT	DTT	DCT	DTT	DCT	DTT
1	26.92	26.92	26.92	26.92	23.37	23.37	23.37	23.37	21.62	21.62	21.62	21.62
4	33.43	33.36	32.38	32.35	25.68	25.61	25.29	25.23	30.20	30.12	29.47	29.43
9	39.65	39.29	40.24	39.99	30.29	30.03	31.78	31.57	39.15	38.54	40.03	39.59

2.7 Conclusions

In this chapter, an 8 × 8 DTT algorithm is implemented in the JPEG in place of 8 × 8 DCT algorithm. By using various test images, it has been observed that DTT has energy compaction property competitive with that of DCT and thereby provides compression performance relatively close with DCT. Therefore, it can be a suitable

Table 2.11: Comparison of PSNR between block-pruned and zigzag-pruned reconstructed images of (d) Finger print, (e) Mountain, and (f) Library.

No. of coefficients retained for image reconstruction	PSNR(dB)											
	Finger print				Mountain				Library			
	Block prune		Zigzag prune		Block prune		Zigzag prune		Block prune		Zigzag prune	
	DCT	DTT	DCT	DTT	DCT	DTT	DCT	DTT	DCT	DTT	DCT	DTT
1	21.88	21.88	21.88	21.88	17.07	17.07	17.07	17.07	16.25	16.25	16.25	16.25
4	28.05	28.25	29.11	29.38	19.59	19.59	19.80	19.82	18.90	18.90	19.42	19.44
9	30.42	30.75	31.94	33.21	22.96	22.97	23.16	23.17	22.69	22.69	23.42	23.45

candidate for applications such as PDA, mobile phones and digital cameras where efficient image compression techniques are required. The proposed technique uses same quantization matrix and zigzag scan pattern as used in JPEG standard. The performance of DTT can be improved further by suitably selecting a proper quantization matrix and adaptive scanning technique.

A novel hybrid HVS based DTT_SPIHT embedded image coding algorithm has been proposed. It has been demonstrated that the proposed image coding algorithm shows an impressive PSNR gain over standard baseline JPEG, EZDCT and STQ, at all bit-rates but comparable with Improved JPEG, STQ+Haar and DCT_SPIHT on smooth and textured images. DTT_SPIHT consistently outperforms DCT_SPIHT for images having sharp edges. By incorporating HVS, the perceptual quality of the proposed algorithm has been improved at a little cost of PSNR values. This is especially noticeable at bit-rates $\leq$ 0.5 bpp. Future research direction is to use adaptive HVS and modified SPIHT algorithm, which is expected to improve the image quality subjectively and objectively at low bit-rates.

Finally, a fast algorithm of 2-D 4×4 DTT has been proposed which pruned the coefficients in a zigzag fashion. This zigzag order pruning can be more suitable for still images and video coding applications because of considerable improvement in objective image quality and fast processing. The pruning algorithm is implemented in a Xilinx XC2VP30 FPGA, which shows considerable amount of hardware savings than a 4×4 floating point DTT. Furthermore, it has been shown that compression using DTT is very similar to compression using DCT for natural and artificial images. Further research work in this field can be extended to develop a fast DTT algorithm for input block of size 8×8.

Chapter 3

Low Complexity Embedded Image Compression Algorithm Using Hierarchical Listless DTT

Preview

Listless set partitioning embedded block (LSK) and Set partitioning embedded block (SPECK) are known for their low complexity and simple implementations. However, the drawback is that these block based algorithms encode each insignificant subband by a zero. Therefore, these algorithms generate many zeros at earlier passes. It is known from the statistics of transformed images that the numbers of significant coefficients at higher bitplanes are likely to be very few. An improved LSK (ILSK) algorithm that codes a single zero to several insignificant subbands has proposed. This reduces the length of the output bit string, encoding/decoding time and memory requirement at early passes. Further, ILSK algorithm has been coupled with discrete Tchebichef transform (DTT). The proposed new coder named as Hierarchical listless DTT (HLDTT) has some desirable attributes like full embeddedness for progressive transmission, precise rate control for constant bit-rate traffic, region of interest retrievability and low complexity for low power applications. The performance of HLDTT is assessed using PSNR and MSSIM. From the simulation result, HLDTT shows significant improvement in PSNR values from lower to medium bit rates. At the same time, HLDTT shows an improvement in MSSIM values over most of the DCT based embedded coders on all bit rates.

3.1 Introduction

Hand held mobile or portable devices have limited memory, processing power and battery life. Real time processing and transmission of images using these devices require an image coding algorithm that can compress efficiently with reduced complexity. Wavelet based image coders such as Embedded zerotree wavelet coder (EZW) [15], Set partitioning in hierarchical trees (SPIHT) [16], Set partitioning embedded block (SPECK) [26], Morphological representations of wavelet data (MRWD)[79] and Significance-linked connected component analysis (SLCCA) [80] provide excellent rate distortion performances by exploiting magnitude correlation within or across bands of decomposition. Each of these coders generates a fidelity progressive bit stream by encoding bit-planes of quantized dyadic subband coefficients.

SPIHT exhibits much better performance over EZW due to additional partitioning steps. The SPIHT algorithm has low complexity. However, being a tree based algorithm, it is memory intensive. SPECK is a block based distortion scalable embedded coder. It uses recursive block splitting methods to isolate significant coefficients. It has excellent coding performance with very low computational complexity. However, it requires memory intensive operation as it uses list structure. A variant of SPIHT called NLS which uses a state table with four bits per coefficient to keep track of set partitions is presented in [35]. Latte *et al.* [36] presented a listless SPECK (LSK) algorithm which uses special markers as in NLS. These markers are updated as block splitting forms new significant blocks.

Though SPECK and LSK are low complexity image coding algorithms with performance nearly close to SPIHT, these coders do not fully exploit the coding performance at lower bit rates. By looking at the statistics of transformed images, the number of significant coefficients whose magnitudes are higher than certain thresholds are very few on earlier bitplane passes. Since LSK does an explicit breadth first search, it codes zeros to each insignificant subbands as it moves from coarsest to finest subbands. There could be six to seven bit plane passes where LSK codes many zeros as many subbands are likely to be insignificant with respect to early thresholds. A block-tree based wavelet algorithm presented in [33] exploits both inter and intra subband correlations to improve the coding performance at very low bit rates. However, it uses depth first scanning and processes ordered list arrays for set partitioning.

In this chapter, an improved LSK algorithm (ILSK) which combines with DTT has been proposed. The proposed new coder which combines DTT with ILSK is named as Hierarchical Listless DTT (HLDTT) is proposed. HLDTT not only reduces the encoder and decoder complexity, but also improves the coding efficiency at lower bit rates. This is achieved by comparing magnitude of coefficients within and across several subbands/levels. A combination of DTT with LSK coder (named as DTT_LSK) is also proposed to test the effectiveness of HLDTT. The performance of these coders are evaluated and compared with some of DCT based embedded coders.

It has been found that the proposed coders DTT_LSK and HLDTT outperforms almost all DCT based embedded coders at lower bit rates. The performance of HLDTT with Integer wavelet transformed based SPIHT (IWT_SPIHT) [89], discrete wavelet transformed/Lift based SPIHT [10] (DWT_SPIHT)[90],[91] and JPEG 2000 [92] is also compared. It is observed that HLDTT shows an average of 1.0 dB PSNR reduction over the JPEG 2000 on considered bit rates. The performance of all these coders are also assessed using state-of-the-art image quality metric MSSIM [3]. It is found that the proposed coder outperforms DCT_SPIHT on all the bit rates for most of the images.

3.2 The Proposed HLDTT Embedded Coder

The proposed HLDTT embedded coder is shown in Figure 3.1. The input image is divided into non-overlapping 8×8 blocks. Each block is transformed using DTT. At the first iteration, the coefficients are arranged into 3 levels of wavelet like pyramid structure. In the next iteration, coefficients in LL_3 subband are further divided into 8×8 blocks and DTT is reapplied to these 8×8 blocks. This facilitates compression at lower bit rates. This is due to better compaction of energy into few DC coefficients. Further, three levels of pyramidal arrangement made using these DTT coefficients. Reapplication of DTT to 8×8 blocks of LL_3 subband makes the coarsest level to be of 8×8. The overall decomposition levels become six. The coefficients are converted to integers and quantized by ILSK coding algorithm. Reverse process is carried out at the decoder side for image reconstruction.

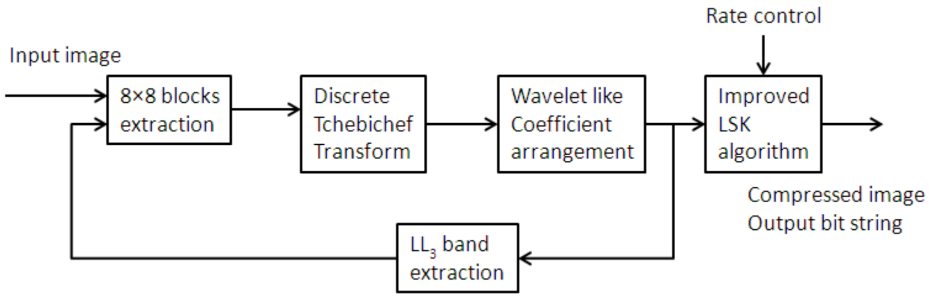

Rate control
Input image
8×8 blocks extraction
Discrete Tchebichef Transform
Wavelet like Coefficient arrangement
Improved LSK algorithm
Compressed image Output bit string
LL_3 band extraction

Figure 3.1: Block diagram of HLDTT embedded image coder

3.2.1 Algorithm for Rearrangement of Transformed Coefficients

The algorithm for rearrangements of the DTT transformed coefficients is exactly similar to that shown in Figure 2.17 in Chapter 2.

3.2.2 Improved LSK Algorithm

The proposed ILSK algorithm uses linear indexing scheme to address a coefficient [35]. The two dimensional arrangement of $M \times N$ wavelet coefficients are arranged into an one dimensional array of length I. The partitioning rules of ILSK are presented as follows:

ILSK makes use of two types of state table markers. These are (i) Fixed markers ($MF[k]$ and $M[k]$) (ii) Variable markers ($MV[k]$) where $k \in \mathbb{N}$. In case of fixed markers, ks' are the leading indices of wavelet levels, whereas in case of variable markers, ks' are the indices of all wavelet coefficients in a transformed image. The initial and final value of state table markers $MF[k]$ and $MV[k]$ depend upon the desired level of decomposition. For instance, for a $N \times N$ (assuming equal dimensions) image and L level of decompositions, the initial and final values are given by

$$[log_2(N) - L] \text{ and } [log_2(N)] \tag{3.1}$$

respectively. $M[k]$ state table hold markers having fixed values (i.e., $log_2(N)$). These markers point to the leading nodes of each wavelet level. Each marker holds 4 bits per coefficients to keep track of set partitions. $MF[k]$ state table markers track the pyramid levels rather than a particular band in a pyramid level. So, a particular pyramid level can be skipped at once instead of a subband by just assigning a single 0 instead of 3 zeros (a wavelet pyramid level consists of 3 subbands, i.e, HL, LH and HH). Next, $M[k]$ state table markers are used to skip several wavelet pyramid levels rather than a single pyramid level at higher bit planes during significant passes. Lastly, $MV[k]$ markers keep track of set partitions within a pyramidal band. ILSK algorithm uses strictly breadth first scan like LSK, though, both coders produce different output bit strings.

There are three passes per bit plane. First, the insignificant pixel (IP) pass which is similar to LIP pass in SPECK where a lone insignificant pixel is being tested for significance. Second, the insignificant set (IS) pass tests each multiple pixel sets for significance like LIS pass in SPECK. IS pass comprises two passes. These are: (a) The insignificant level (IL) pass which tests a pyramid level for insignificance and (b) The insignificant group of levels (IGL) pass which tests several pyramid levels for insignificance. Finally, the refinement (REF) pass that refines pixels found significant

in previous bit-plane passes. IL and IGL passes in IS pass are effective for some of the higher (initial passes) bit plane. As the scanning of bit planes from MSB to LSB goes down, IL and IGL passes shall be ignored. This is because most of the wavelet coefficients become significant at lower bit planes. It can be noted that IL pass and IGL pass make use of $MF[k]$ and $M[k]$ state tables respectively.
Each marker and its meaning is listed below
$\mathbf{S} * \mathbf{P}$ Each of these three symbols is used for a single pixel.

- SIP: The pixel is insignificant or untested for this bit plane.
- SNP: The pixel is newly significant so it will not be refined for this bit plane.
- SSP: The pixel is significant and will be refined in this bit plane.

The following markers are used on the leading node of each lower level of pyramid. As the image is scanned, these markers indicate that the next level or subband or a block is insignificant.

Fixed markers ($MF[k]$):

- $MF[1]$: The pixel is the first index of pyramid level L. This pixel along with all coefficients in the same pyramid level can be skipped.
- $MF[257]$: The pixel is the first index of pyramid level $L-1$. This pixel along with all coefficients in the same pyramid level can be skipped.
- $MF[1025]$: The pixel is the first index at pyramid level $L-2$. This pixel along with all coefficients in the same pyramid level can be skipped.
  $\vdots$
- $MF[65537]$: This is the first index at the finest pyramid level. This pixel along with all coefficients in this level can be skipped.

Variable markers ($MV[k]$): The function of variable markers for $(L-1)^{th}$ level (whose leading index is 257) is explained below. The same can be generalized for other levels.

- $MV[257]$=$MF[257]$ indicates that the entire wavelet level $L-1$ can be skipped.
- $MV[257]$=$MF[257]$-1 indicates that a subband in the wavelet level $L-1$ can be skipped.
- $MV[257]$=$MF[257]$-2 indicates that $1/4^{th}$ of a subband block in the wavelet level $L-1$ can be skipped.
  $\vdots$

- $MV[257]$=0 indicates that block size equals to pixel size. The pixel is to be tested for significance.

Fixed markers ($M[k]$):

- $M[65]$ indicates that all pyramid levels except L can be skipped.
- $M[257]$ indicates that all pyramid levels except L and $L-1$ can be skipped.
- $M[1025]$ indicates that all pyramid levels except L, $L-1$ and $L-2$ can be skipped.

  ⋮
- $M[65537]$ indicates that only the first(finest) pyramid level can be skipped.

$k = 65, 257, 1025...65537$ are the leading indices starting from resolution level L to the finest resolution level. A total of six levels of arrangement is made in images of 512×512 dimensions.

After mapping the two dimensional arrangement of wavelet coefficients into an one dimensional array of length I, the coding process starts with most significant bit plane and proceeds towards the finest resolution until a bit budget is met. The significance level for each bit plane is s=2^n which is done with bitwise AND operation. The decoding operation is exactly reverse of encoding operation with some low level changes. The decoder sets the bits and sign of coefficients with bitwise OR instead of bitwise AND. Symbol Γ_n is significance test function and symbol L_L is the L^{th} level of a pyramid. In each pass, the coefficient array *val* is examined for significance with respect to current threshold. The output of $\Gamma_n(\ )$ will be '0' or '1' depending on the relative magnitude of the coefficient array (*val*) with respect to current threshold n. Note that *val*[k] refers to a single coefficient and *val*[$k : I$] or *val*[k : *end of* L_L] refer to a coefficient array. The algorithm is initialized by computing maximum threshold which is defined as:

$$n = \left\lfloor \log_2(\max_k |val(k)|) \right\rfloor \tag{3.2}$$

The initialization of state table markers are defined below:

1. Initially, the markers $MF[k]$ and $MV[k]$ correspond the leading indices of sub-bands. The markers are initialized according to Eqn. 3.1 as follows:

- $MF[1, 17, 33, 49] = MV[1, 17, 33, 49] = 3$ for LL_6(Coarsest subband). (Assuming $M^*[k_1, k_2, ...k_n] = M^*[k_1] = M^*[k_2] = ... = M^*[k_n]$, where symbol M^* can be replaced by MV or MF or M and $k_1, k_2, .., k_n \in k$).

- $MF[65, 129, 193] = MV[65, 129, 193] = 4$, correspond the leading nodes of HL_6, LH_6 and HH_6 subbands.
- $MF[257, 513, 769] = MV[257, 513, 769] = 5$, correspond the leading nodes of HL_5, LH_5 and HH_5 subbands.
  ⋮
- $MF[65537, 131073, 196609] = MV[65537, 131073, 196609] = 9$, correspond the leading nodes of HL_1, LH_1 and HH_1 subbands.

2. $M[65, 257, 1024, ..., 65537] = 9$.

3. Other $MV[k]$ markers which do not correspond the leading indices of subbands are initialized to any arbitrary value greater than the highest marker value. These $MV[k]$ markers are marked as SIP.

3.2.3 The Pseudo code of ILSK Algorithm

```
Pass1 : Insignificant Pixel Pass
  $k = 0$, while $k \leq I$                                  //Start of IP pass
  ①. IF $MV[k] = $ SIP                                       //insignificant pixel
        output(d $\leftarrow val[k]$ AND s)                      //send bit
     if d
        output(sign[$k$])                                    //send the sign
        $MV[k] \leftarrow$ SNP                                  //newly significant
        else, move to next pixel.
  ②.ELSE, move to Insignificant set pass

Pass2 : Insignificant Set Pass
  ①. IF $MV[k] = MF[k]$ & $MV[k] \neq$ SIP          //Significance test a Level
       if $k \leq I/2^{2L}$                           //if at coarsest subband, LL$_6$
          ProcessS( )
       else, output{d $\leftarrow \Gamma_n(val[k : I])$}         //IGL pass(uses $M[k]$ state table)
          if d, output{d $\leftarrow \Gamma_n(val[k : end\ of\ L_L)]$}    //IL pass(uses $MF[k]$ state table)
             if d, ProcessL( )
          else, skip to Refinement Pass.
  ②. ELSEIF, $MV[k] \neq MF[k]$ & $MV[k] \neq$ SIP        //Significance test of
                                                       //a subband/block
          ProcessS( )
  ③. ELSE, Move past the pixel.
       if $k > I$, Move to Refinement pass
```

```
        Pass3 : Refinement Pass :
          $k = 0$, while $k \le I$
          ①. IF $MV[k]$ = SSP                              //if significant
            output($val[k]$ $AND$ $s$)                     //refine the pixel
            move past the pixel
          ②. ELSEIF, $MV[k]$ = SNP                         //if newly significant
            $MV[k] \leftarrow$ SSP                         //significant for next plane
            move past the pixel
          ③. ELSEIF, $MV[k] = MF[k]$ & $MV[k] \ne$ SIP
              if $k \le I/2^{2L}$, skip the set tested insignificant
              else, check $\Gamma_n(val[k:I])$             //IGL pass
                  if insignificant
                    skip to Pass4
                  else, check $\Gamma_n(val[k : end\ of\ L_L])$     //IL pass
                      if insignificant
                        skip the Level
                      else, move to the next subband/set
          ④. ELSE, move to the next subband/set/pixel

      Pass4 : Update Pass :
        $n = n - 1$                                        //Decrease threshold by 1
        Move to Pass2

ProcessS( )
                      output(d $\leftarrow \Gamma_n$(subband/set))
                        if d, QuadSplit()
                          if (block size = pixel size)
                            move to Pass1
                          else, move past subband/set
ProcessL( )
                    output(d $\leftarrow \Gamma_n$(Level $L_L$))
                      if d, Split $L_L$ into 3 parts.
                      else, move to the leading index of $L_{L-1}$.
```

3.2.3.1 Description of Functions and Parameters used in Pseudocode

1. Significance test function ($\Gamma_n(X)$): The significance test function computes whether any coefficient inside a wavelet level/subband/block or even a coefficient is significant with respect to current threshold or not. This is simply achieved by logical AND operation. For an example, assuming coefficients inside a block X=[-126 110 20 -34], and current threshold value $n = 6$. Γ_n can be calculated as

$$\Gamma_n(X) = \sum_{\text{all k}} [(2^n \le |X(k)|) \text{ AND } (\,|X(k)| \le 2^{n+1})] \tag{3.3}$$

if $\Gamma_n(X) = 0$, then output $= 0$
else, QuadSplit().

2. Function QuadSplit( ):

Figure 3.2 shows one to one correspondence between $MV[k]$ values and the coeffi-

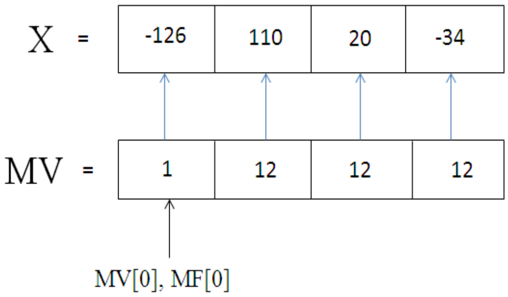


Figure 3.2: The one to one correspondence between $MV[k]$ values and coefficient values $X(k)$ is shown. Here the values of k=0,1,2 and 3. If $\Gamma_n(X) \neq 0$, $MF[0]$ do not change its value while $MV[0]$ change its value to indicate quad partitioning.

cient values $X(k)$. By updating the markers, it is possible to quad split X. This can be explained as follows:

Let initially $MV[0] = MF[0] = 1$ and $n = 6$. $MV[1]$, $MV[2]$ and $MV[3]$ will be initialized to any value other than the values that are initialized to fixed and variable markers (i.e., symbol SIP). $MV[k] = 12$ for k=1,2, and 3 is shown in Figure 3.2 for illustration. Since $\Gamma_n(X) \neq 0$ for threshold $n = 6$, $MF[0]$ marker does not change its value; while $MV[0]$ changes its value by decrementing to 1. The following code will split X into four individual coefficients by updating $MV[k]$ values to 0.

```
$MV[k] = MV[k] - 1;$
for $j = 1, 2, 3$
    $MV[k + (j \times 2^{2\times MV[k]})] = MV[k]$
end
```

It is to be noted that if $MV[k] = 0$, there would not be any quad splitting. The corresponding coefficient in X is an insignificant pixel (SIP) and it will be tested for significance in Insignificant pixel pass (Pass 1) of the algorithm.

If X is a wavelet level (In Figure 3.2, X is shown as a quad block. At least 3 such quad blocks are needed for X to become a level), then the significance function for L^{th} level, $\Gamma_n(L_L)$ is examined according to Eqn. 3.3. If $\Gamma_n(L_L) \neq 0$, then L_L is to be split into 3 parts instead of quad split, because a wavelet level consists of three subbands. The lines of the code are exactly equal to the function QuadSplit( ), except, j takes the values 1 and 2. Assuming $MV[k]$ marker be initialized to 2, the code will update $MV[k]$ markers to 1 for values of k=0, 4 and 8. This is because, it is assumed that X

consists of 12 coefficients, so k varies from 0 to 11. Therefore, by updating markers, X is partitioned into 3 parts. When HLDTT examines $\Gamma_n(L_L)$, this condition is called IL pass. The IL pass is effective (i.e., when $\Gamma_n(L_L)$=0) for higher bit planes, because more coefficients present in lower subband levels are likely to become insignificant.

If X consists of several wavelet pyramid levels (At least 3 quad blocks for 1st level and (3×4) quad blocks like X in Figure 3.2 for 2nd level), the $\Gamma_n(X)$ function is also calculated according to Eqn. 3.3. If $\Gamma_n(X)$ is significant, then the algorithm check the significance of a wavelet level. i.e., $\Gamma_n(L_L)$. If $\Gamma_n(L_L) \neq 0$, it will be split into 3 parts, else the L^{th} wavelet level can be skipped according to equation:

$$k = 2^{2\times MF[k]} + 1 \tag{3.4}$$

The value k will be the leading index of next lower wavelet level (L_{L-1}). If $\Gamma_n(X)$=0, the algorithm immediately proceeds to Refinement pass by skipping all wavelet levels according equation

$$k = 2^{2\times M[k]} + 1 \tag{3.5}$$

This is the situation where IGL pass is effective. Typically, IGL pass will be effective for top 1 to 5 passes in most of the standard gray scale images. As the algorithm scans down towards least significant bit planes, IGL pass shall be ignored. This is because, most of the coefficients are likely to become significant.

3. $MV[k] = MF[k]$ indicates, a level to be skipped. $MV[k] \neq MF[k]$ indicates a subband/block (not level) is significant w.r.t. threshold n. If it is a subband, it will be quad split and encoded. Each block can be split in to individual coefficients.

4. If $MV[k] \neq SIP$, then marker is pointing to the leading index of a set/block. If $MV[k] = SIP$, marker points to a coefficient which is insignificant (i.e., $MV[k] = 0$) or not yet tested for significance (i.e, $MV[k] = 12$ in Figure 3.2).

3.2.4 Region of Interest Retrievability

HLDTT groups the transformed coefficients into different subbands as shown in Figure 3.3(a). This is similar to a hierarchical octave band decomposition as shown in Figure 3.3(b), where L levels of decomposition results in $3L + 1$ subbands. For an image of size $M \times N$ which is decomposed at level L, the number of trees N_T that can be organized will be equals to the number of coefficients inside the lowest frequency subband. Therefore,

$$N_T = M/2^L \times N/2^L \tag{3.6}$$

If we denote the coordinate of a coefficient in the coarsest subband by (i, j) and the coordinate of the top left corner of the ROI block by $(i*, j*)$, then we have the following relationship:

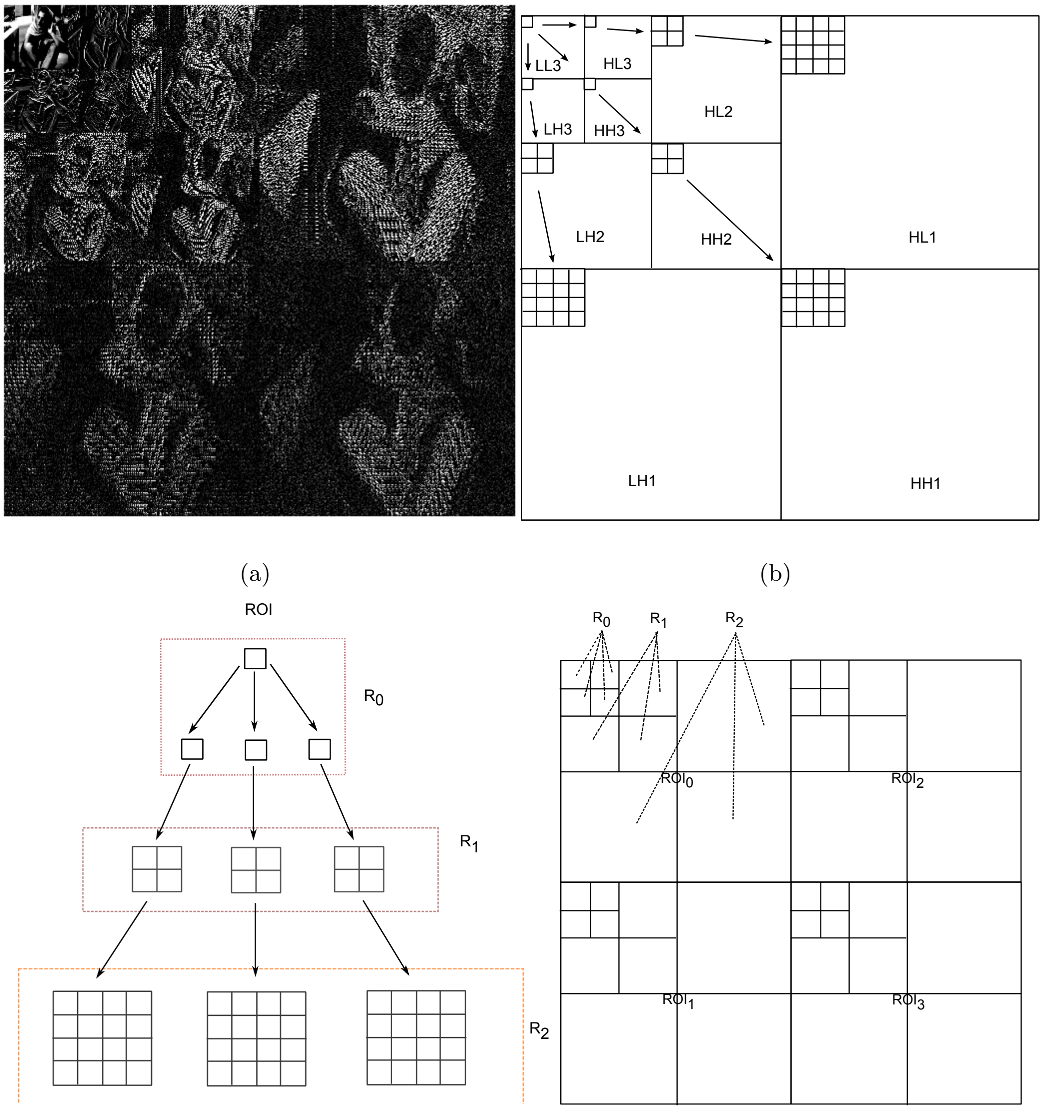
$$\begin{aligned} i* &= i \times 2^L \\ j* &= j \times 2^L \end{aligned} \tag{3.7}$$

In Figure 3.3(b), 64 square regions are retrieved independently by organizing 6 levels of decompositions of Barbara image (three resolution levels R_0, R_1 and R_2 are shown here for illustration). In order to detect ROI, it is necessary to group the coefficients as trees. This can be explained as follows:

Figure 3.3(c) shows 3 hierarchical trees extracted from Figure 3.3(b). Therefore, it is obvious that 3 such trees having a root located in the coarsest subband constitute a single ROI. Extending Figure 3.3(c) up to 6 levels, the finest level of each tree consists of 32 × 32 coefficients. Arranging 3 such trees like in Figure 3.3(d), each ROI will have 64×64 coefficients. In the proposed algorithm each tree constitutes 6 resolution levels. Therefore, the whole image comprises 64 × 3 such trees having roots located on the coarsest subband. Figure 3.3(d) shows the arrangement of $3 \times 4 = 12$ such trees to form 4 ROIs.

Since each ROI is 64 × 64 size, selecting 64 × 64 blocks starting from top left coordinate (0,0) of Barbara image of Figure 3.4(a) in a horizontal raster scan manner (left to right and then top to bottom), we can find the selected facial portion is at a co-ordinate (64,256). It is observed that 64 × 64 block size at (64,256) only covers a quarter of Barbara face. Therefore, it need to select some more 64 × 64 blocks to accommodate the complete face portion. This requires selecting next top left co-ordinate (64,320) as well as the blocks having top left co-ordinates (128,256) and (128,320) of the original 512 × 512 image.

In order to detect the facial portion of Barbara image whose top left co-ordinates are at location $i* = 64$ and $j* = 256$ in the original image, the corresponding coordinates (i, j) in the lowest frequency band is (1,4) according to Eqn. 3.7. Similarly, other 3 descendant coefficients of (1,4) are located at (9,4) in HL_6, (1,12) in LH_6 and (9,12) in HH_6 subbands in 6 levels of hierarchical arrangements. Therefore, we need to group together (1,4), (9,4), (1,12) and (9,12) coefficients corresponds to resolution level R_0 and their respective descendant tree blocks of next finer resolution levels. After grouping the coefficients the arrangement look like ROI_0 portion of Figure 3.3(d). Similar coefficient arrangements are made for ROI_1 having tree root (1,5) in LL_6, ROI_2 having tree root (2,4) in LL_6 and ROI_3 having tree root(2,5) also in LL_6 to



(a) (b)

(c) (d)

Figure 3.3: (a) Arrangement of DTT coefficients after 6 levels of decompositions on Barbara image (b) Hierarchical decomposition of transformed coefficients across different scales of a pyramid (c) Grouping of DTT coefficients as trees and resolution levels (d) Arranging 4 ROIs from the whole image.

accommodate the face of Barbara image. Then, each ROI block is mapped into one dimensional arrangement in a recursive zigzag manner. The length of each ROI form R_0 to R_{L-1} is initially compressed at 2.0 bpp. Therefore, the length of each ROI becomes (64×64)pixels $\times$ 2(bpp) $=$ 8192 bits. This length information is used to locate the ROI positions. Figure 3.4(b) shows the Barbara image with ROI after applying rate-distortion (R-D) optimization. A detailed discussion about R-D optimization is deferred to subsection 3.2.4.1.

HLDTT is not resolution scalable. However, it is signal-to-noise-ratio (SNR) scalable. In order to optimally allocate bits across different ROIs for a given overall bit rate, we have used the quality layer concept of EBCOT [27] algorithm. The set of coefficients in the same resolution level of a spatial orientation tree (SOT) is called codeunit [31]. This means, codeunit C_0 corresponds to resolution level R_0, C_1 corresponds to R_1 and C_{L-1} corresponds to R_{L-1} as shown in Figure 3.5(a).

Figure 3.5(a) also shows the encoded bit stream structure of ROI blocks. The initial ROI block includes two overhead fields, i.e. l_0 and R. Other ROIs only include l field. The l field records the length of ROI and it also differentiates the boundary between ROIs. R field records the number of resolution levels. These two fields are generally set by the user.

R_0 is the lowest resolution level that corresponds to codeunit C_0, R_{L-1} is the highest resolution level corresponding to codeunit C_{L-1}. $c_0, c_1, ...$and c_{L-1} are the bit strings generated during encoding codeunits.

Each codeunit C_i in a ROI consists of quality layers $(Q^j, Q^{j+1}, ...)$ as shown in Figure 3.5(b). The header of quality layer Q^j consists of $l_i{}^j$ fields as shown in Figure 3.5(c). $l_i{}^j$ is the length of optimal truncation point (marked as $T_i{}^j$) for C_i codeunit. The boundary between the quality layers is obviously the sum of $l_i{}^j$, where $i = 0, 1, 2, ..L - 1$.

Therefore, total number of overhead bits are $((N_l \times N_{ROI}) + N_R) \times O_H$, where N_l is the number of bits in l field, N_{ROI} is the number of ROIs, N_R is the number of bits in R field and O_H is the overhead bits in codeuints. Then each ROI is decoded by HLDTT algorithm with bit rate lower than the encoded rate. In this example, the 4 ROI blocks corresponding to facial portion are decoded at 1.0 bpp while the ROI corresponding to other blocks are decoded with 0.125 bpp. Therefore, to retain ROI retrievability, it is necessary to group the coefficients into trees and code them independently. The main difference between the proposed algorithm and S-SPECK is that S-SPECK is a scalable wavelet coder which can achieve ROI retrievability and resolution scalability at one time. The proposed coder do not possesses resolution scalability unlike wavelet coders where resolution scalability is an inherent property. This is because the size of DTT is fixed for a particular coder.

3.2.4.1 Rate-Distortion optimization

Each codeunit is bit plane coded. The R-D characteristics curve of Figure 3.6(a) shows the available truncation points corresponds to each pass for a codeunit C_i. For each truncation points, $z \in \{1, 2, ...Z_j\}$, the length $T_i^{(z)}$, identifies the smallest prefix of the embedded bit stream for each pass. The first available truncation point $z = 0$ always correspond to discarding the entire bit stream so that $T_i^{(0)} = 0$. In the proposed method, each code units are of varying size which are proportional to their resolution levels.

It is evident that the R-D properties of the overall compressed image representation depend upon the selection of appropriate truncation points for each code block. Therefore, in general, given any $s > 0$, any set of truncation points, $\{z_j\}$ which minimizes the Lagrangian cost function [93]

$$\sum_j D_i^{(z_j)} + s \sum_j T_i^{(z_j)} \tag{3.8}$$

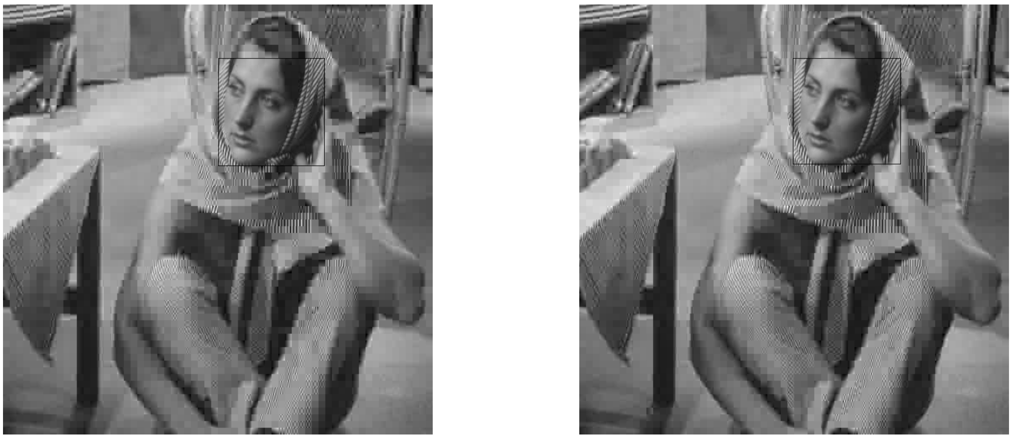
is optimal. Where $D_i^{(z_j)}$ is the distortion associated with codeblock C_i truncated at pointed z_j and $T_i^{(z_j)}$ is the distance associated with same truncated point z_j. The value of s is selected so that the solution which minimizes Eqn. 3.8 achieves the desired overall bit rate or distortion.
The algorithm to find the valid truncation points z_j, that minimizes Eqn. 3.8 is as follows:

Algorithm 1 R-D optimization

```
Step-1: initialize $z_j = 0$.
Step-2: Find all the set of truncation points for codeunit $C_i$;
Step-3:
for $k = 1, 2, ....$ do
    Set $\Delta T_i^k = T_i^k - T_i^{z_j}$ and $\Delta D_i^k = D_i^k - D_i^{z_j}$;
    if $\frac{\Delta D_i^k}{\Delta T_i^k} > s$ then
        update $z_j = k$
    else
        remove $z_j$
    end if
end for
```

Let H_i be the set of all truncation points for codeblock C_i, which are solution to the optimization problem for some value of the parameter s. H_i describes the vertices of the lower convex hull of the set of distortion-rate pairs, $(D_i^{(z)}, T_i^{(z)})$. These are illustrated in Figure 3.6(a) as black dots. Vertices in the interior of convex hull (whose $\frac{\Delta D_i^k}{\Delta T_i^k} < s$, and shown as white dots) shall never be selected by an optimal algorithm because these truncation points contribute least to the overall image quality.

Since the distribution of the magnitude of DTT coefficients vary among code units, each codeunit contributes different number of bits to a quality layer. The optimized codeunit truncation length T_i^j mark the different contributions from each codeunits to quality layer Q^j, which minimize the distortion $D_i^j = \sum_j D_i^{z_j}$, subject to the length constraint, $\sum_j T_i^{z_j} \leq T_i^j$ as shown in Figure 3.5(b). Subsequent layer, Q^{j+1}, contain additional contributions from each codeunit, having length $T_i^{j+1} - T_i^j$, which minimizes the distortion, $D_i^{j+1} = \sum_j D_i^{z_j}$ subject to the length constraint $\sum_j T_i^{z_j} \leq T_i^{j+1}$.



(a) (b)

Figure 3.4: ROI retrievability on Barbara image (a) Before RD optimization, (b) After RD optimization

The pseudocode for ROI detection is as follows:

```
                         Algorithm for ROI:

Initialization :
    Threshold $n = \lfloor \log_2(\max|c_{ij}|) \rfloor$ .
    Set the index of the current ROI as 0.
    Set the maximum encoding bit length as $l_k$
    and the maximum resolution levels as R.
Step1 : Organize the wavelet coefficients as spatial orientation trees,
        and then group them into ROI blocks.
Step2 : Map each resolution level to it's codeunit
Step3 : If current index of ROI is 0 then
        Fill the header of ROI with $\{l_0, R\}$
        Else fill $\{l_k\}$ where, $k = 1, 2, ....$
Step4 : For each codeunit perform HLDTT encoding.
        Use rate − distortion optimization and mark the truncation points for each codeunit.
Step5 : Generate quality layer $Q^j$ by concatenating
        the overall incremental contribution $(T^j - T^{j-1})$ from all the codeunits.
Step6 : If the maximum length of quality layer is met
        or if the desired rate is achieved then
        $k = k + 1;\ j = j + 1$
        Go to step 3
        Else, go to step 4.
```

If the length of every codeunit is determined, then finding out the exact target rate for every code block is a trivial task. In most of the cases, valid truncation points do not lie at desired target bit rate. If the truncation points for each codeunits are large, it is sufficient to find smallest value of T such that $R(T) \leq R^{max}$. $R(T)$ is the bit rate at truncation point and R^{max} is the available bit rate corresponding to the codeunit. The lowest quality layer is formed by concatenating the valid truncation points for all the codeunit of an ROI. The following example illustrated below, discusses how to find out the exact bit rate for a quality layer Q_i.

Example: Let the target rate for quality layer-0 is $R(Q^0)$. For each codeunit, the R-D optimization algorithm sets a maximum slope s and eliminated all the slopes that are smaller than s (The coefficient with maximum R-D slope determines the initial value of s). Then, s is reduced by a certain factor, and is adjusted until the desired length (rate) is reached. Let the bit string length corresponding to the truncation points for codeunit C_0 is T^0. Then the bit rate for the codeunit-0 (C_0) is, $R^0 = \frac{T^0}{T_0^{max}}$, where, T_0^{max} is the maximum available length of the bit string corresponding to C_0. Similarly for other codeunits $C_1, C_2, ...C_{L-1}$, the bit rates can be expressed as $R^1 = \frac{T^1}{T_1^{max}}, R^2 = \frac{T^2}{T_2^{max}}, ..., R^{L-1} = \frac{T^{L-1}}{T_{L-1}^{max}}$ respectively.

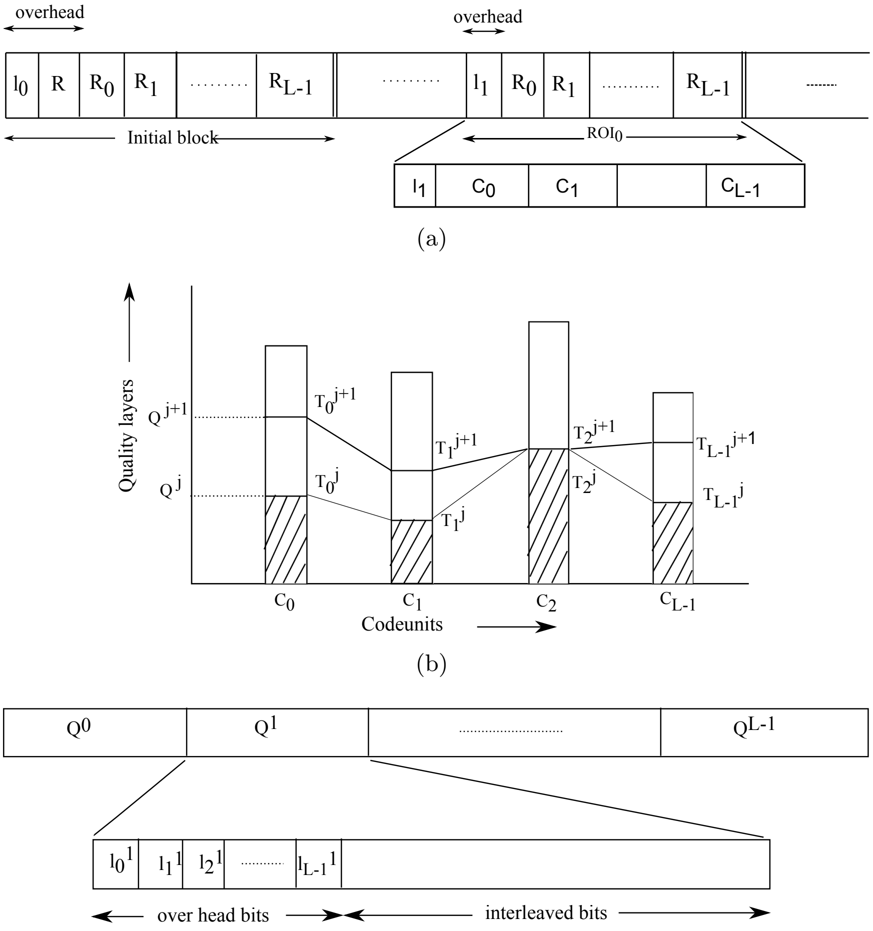
The overall bit rate of the quality layer Q^0 formed by concatenating the truncation points. This means overall rate for the quality layer-0 is

$$R(Q^0) = \frac{T^0 + T^1 + ... + T^{L-1}}{T_0^{max} + T_1^{max} + ... + T_{L-1}^{max}} \tag{3.9}$$

In our experiment, every codeunits for a quality layer is truncated at a bit rate of 1.0 bpp in the facial part of barbara image. That means $R^0 \approx R^1 \approx .... \approx R^{L-1} \approx 1.0$ bpp. It is interesting to note that the overall bit rate for the quality layer, $R(Q^0) \approx \frac{R^0+R^1+....+R^{L-1}}{L}$. The reason is that the ratio of the proportional increase between truncation points length and maximum bit length of codeunits have kept the bit rate fixed.

Figure 3.4(b) shows the overall PSNR improvements on the reconstructed Barbara image before R-D optimization and after R-D optimization. In both the cases, the bit rate of ROI part is varied (from 0.125-2.0) while keeping bit rate fixed on the rest of the part at 0.125 bpp. It is observed that there is an overall PSNR improvement of 0.13-0.25 dB. Subjective enhancement in Figure 3.4(b) over Figure 3.4(a) shows the validity of the proposed R-D optimization algorithm.

In case of EBCOT, post compression rate distortion (PCRD) is applied to compute the truncation points. However, the PCRD in EBCOT is a time consuming operation because of computation of square values of each coded pixel. S-SPECK [31] speeds



(a)

(b)

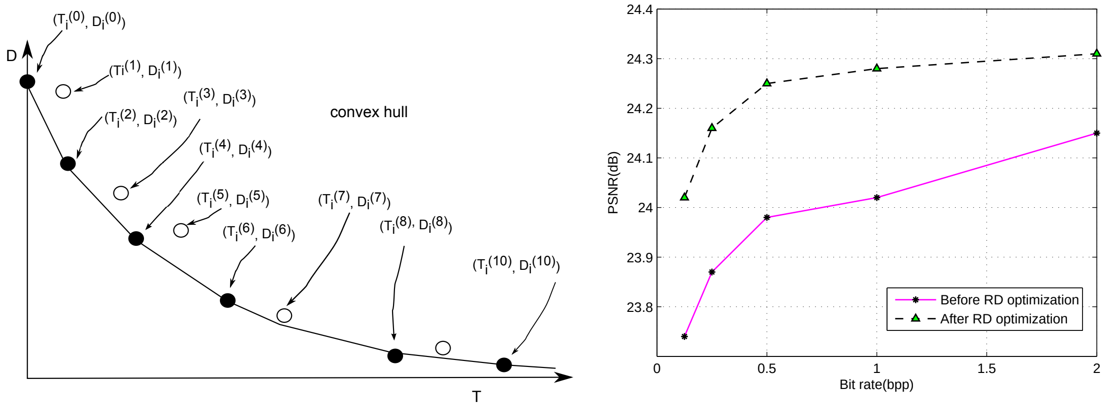
(c)

Figure 3.5: Formation of compressed bit stream using ROIs (a) Overall bit stream structure after ROI arrangement (b) Formation of quality layers for different codeuints (c) Final SNR scalable compressed bit stream of an ROI using quality layers.

up this computation by executing rate distortion method during its encoding process. HLDTT executes the rate distortion function during the encoding process like S-SPECK. In HLDTT, the quality layer Q^j is formed by collecting the bits from different codeunits of the encoder like in S-SPECK. If the maximal length of quality layer is met, a new empty quality layer Q^{j+1} replaces the current quality layer Q^j and waits for bits from the encoder. This ensures that the quality layers in HLDTT are optimal because these consist of as many significant coefficients as possible.

3.2.4.2 Strengths of the proposed method compared to Maxshift method in JPEG2000

- In the proposed method, it is possible to have grading the compression among different ROIs depending on user priorities. Since the background and several



(a) (b)

Figure 3.6: (a) Convex hull of the distortion-rate pairs for the code block C_i, (b) R-D optimization on Barbara image

ROIs are compressed at the same bit rate in Maxshift method, there is no possibility of grading the compression among different ROIs is allowed.

- In Maxshift method, scaling parameter is greater than or equal to maximum bit plane coefficient. Therefore, overflow problem arises due to limited precision implementation. The problem is avoided by scaling the background coefficients rather than scaling up the ROI coefficients. Still, some of the least significant planes for the background may lost in most of the implementation. However, overflow problem never arises in the proposed method.
- Several artifacts can disturb the general appearance of the recovered image in Maxshift method if not enough information about the background is encoded. However, artifacts never appear on the ROI part of the image in the proposed method (The artifacts which appear in the proposed scheme at lower rates is due to block based DTT).
- The proposed scheme naturally supports random access attributes without having information about the background pixels. Maxshift supports random access attributes but background pixel values are to be known beforehand in order to scale up the ROI part.

3.2.5 Memory Requirement

The number of coefficients in the DC band is $I_{dc} = M_{dc} \times N_{dc}$ where $M_{dc} = M \times 2^{-L}$, $N_{dc} = N \times 2^{-L}$, M is the number of rows, N is the number of columns and L is the number of subband decomposition levels. The coefficients are stored in a single array of length I. Zero tree coders can optionally trade memory for computation by precomputing and storing maximum magnitude of all possible descendant and grand descendant sets [35]. For ILSK, the precomputed maximum length array L_{max} has length $[I - (2^L + 1)]$. If W bytes are needed for each sub band coefficients, then the bulk storage memory required is IW for the subband data and $MN/2$ (each marker is of half byte) for the state table $MV[k]$. For L levels of wavelet decomposition, $MF[k]$ needs $\lceil (3L + 4)/2 \rceil$ bytes (the number of fixed markers are $(3L + 4)$ and each marker is of half byte) and $M[k]$ state table needs $\lceil (L/2) \rceil$ bytes. Therefore, the total memory required for ILSK is

$$M_{ILSK} = IW + MN/2 + \lceil (3L + 4)/2 \rceil + \lceil (L/2) \rceil. \tag{3.10}$$

The memory required for LSK is:

$$M_{LSK} = IW + MN/2 \tag{3.11}$$

It is to be noted that $MF[k]$ markers do not update. This is used as a reference for $MV[k]$ markers to check the significance of a level/subband. When comparing Eqn. 3.10 with Eqn. 3.11, M_{ILSK} is $(2L + 2)$ bytes higher than M_{LSK}. Assuming $L = 6$, it is only 14 bytes.

Table 3.1: Distribution of DTT coefficient magnitudes along thresholds and frequency levels(6: Highest, 1: Lowest) for Lena(512×512) image

Threshold, 2^n	Magnitude range	Frequency level					
		6	5	4	3	2	1
2^{12}	$2^{12}, ..., (2^{13} - 1)$	1	0	0	0	0	0
2^{11}	$2^{11}, ..., (2^{12} - 1)$	28	0	0	0	0	0
2^{10}	$2^{10}, ..., (2^{11} - 1)$	51	15	2	0	0	0
2^{9}	$2^{9}, ..., (2^{10} - 1)$	52	69	19	1	0	0
2^{8}	$2^{8}, ..., (2^{9} - 1)$	45	149	126	134	4	0
2^{7}	$2^{7}, ..., (2^{8} - 1)$	30	148	333	597	156	4

3.2.5.1 Dynamic Memory Comparison

Comparing Eqn. 3.10 and Eqn. 3.11, it is concluded that ILSK requires slightly more memory than LSK. However, the memory required by ILSK is significantly lower than LSK, typically on top 5 bit plane passes. The reason is that markers $M[k]$ skip $MF[k]$ and $MV[k]$ markers corresponding to several lower subbands/levels. Table 3.1 shows the distribution of DTT subband coefficients on top 6 thresholds of 6 levels of coefficient arrangement. An estimation of approximate memory required for ILSK, SPIHT and LSK on top 9 passes on Lena image is shown in Table 3.2. Note that

Table 3.2: Comparison of memory requirements between LSK, SPIHT and ILSK algorithms on top 9 passes.

No. of sorting passes	Lena Image		
	$M_{LSK}(kB)$	$M_{SPIHT}(kB)$	$M_{ILSK}(kB)$
1	128	0.8985	0.0313
2	128	0.9004	0.1250
3	128	0.9375	0.5
4	128	5.0088	2.0
5	128	10.039	8.0
6	128	48.872	128
7	128	82.535	128
8	128	136.892	128
9	128	208.512	128

the minimum number of bits required to represent markers depends upon the level of decomposition and number of markers. For six levels of decomposition LSK needs at least 4 bits to represent each marker, which is same as ILSK.

It is observed that in most of the images there are few coefficients above most significant bit (MSB) plane. For an instance, Barbara/Lena/Mandrill image is having only one significant coefficient in LL_6 subband above MSB plane as shown in Table 3.4. Similarly, all 7 significant coefficients above MSB plane in Boat image are confined inside the coarsest subband. According to ILSK algorithm, $M[k]$ markers related to leading node of next subband simply skip $MF[k]$ and $MV[k]$ markers related to the lower levels/subbands. Therefore, ILSK algorithm skips to next lower significant bit plane from the MSB plane. Since, LL_6 subband is having a dimension of 8×8, the maximum dynamic memory required is $\frac{8\times8}{2} = 32$ bytes ($\sim 0.0313\ kB$) on first sorting pass. The memory required by other passes is calculated in a similar procedure. LSK needs a fixed amount of memory i.e., $\frac{MN}{2}$ irrespective of the number of passes. However, the memory required by the ILSK are almost same as that of LSK, typically 6, and above passes. The memory required by SPIHT coder can be found from [33].

For a 512 × 512 image using 2 bytes per coefficient and 6 levels of wavelet like arrangement, with using the optional pre-computed maximum length array, the worst case memory required is: $\{\frac{(512\times512)}{2} \times (8\text{bits}) + (2L+2)\} \simeq 128$ kB for ILSK, 128 kB for LSK and 1450 kB for SPIHT. The memory required for DTT is not calculated here. This part can be efficiently handled by distributed arithmetic based approaches [77]. Therefore, our algorithm is suitable for a fast and simple implementation.

3.2.6 Complexity Analysis

The complexity of a coding algorithm depends upon target rates, number of bit planes to be processed and number of list arrays to be processed. Encoding algorithm performs basic operations such as memory access, magnitude comparison against a threshold and some input/output operations. On the other hand, decoding algorithm performs bit manipulations, input/output and memory access operations. It is to be noted that decoding algorithms are faster than encoding algorithms because:

- No comparison operations are required.
- Efficient skipping of insignificant sets/blocks/subbands.

The complexity of a algorithm also depends upon target rate, because more number of operations are required at higher target rates.

In the list based coders, the processing of the lists account for much overhead computations. DCT_SPIHT and proposed algorithms DTT_LSK and HLDTT are based on bit plane and requires multiple passes across bit planes to reconstruct a coefficient. DCT_SPIHT algorithm accesses coefficients through lists many times to reconstruct itself. Coefficients in LSP of SPIHT are further processed by the refinement pass. A bit of DCT coefficient is added to the compressed bit stream for each entry of LSP.

In DCT_SPIHT, the information about the coefficient significance is managed by LIP. A coefficient, once entered in LIP, remains there until it becomes significant and moves to LSP. The proposed coders, DTT_LSK and HLDTT do an explicit breadth first search without using lists. State informations are kept in fixed size arrays to enable fast scanning of bit planes. In HLDTT, special markers are placed on the lower nodes of subbands/levels/group of levels instead of searching for the trees to find predictable insignificance. These markers are updated when new significant sets/blocks are formed by magnitude comparison. With these sparse marking, a large section (blocks/subband/level/group of levels) are skipped at once as the breadth first scan moves through the lower nodes of pyramidal decomposition array. If a subband/level/group of levels are insignificant with respect to a threshold, it is just skipped. Otherwise, it is partitioned recursively until a significant coefficient is found. The probability that a block/subband is insignificant at higher bitplanes is relatively high. Therefore, the proposed algorithm ILSK has the advantages of quickly finding significance of a coefficient by recursively partitioning scheme and efficient block skipping. Hence, it has the advantage over memory access time. This expect that the encoder of HLDTT and DTT_LSK are faster than DCT_SPIHT.

At the decoder, significance of a coefficient is reconstructed from the received bit map. When a coefficient/block/subband/level is insignificant, the decoder skips all operations and moves to test the significance of the next coefficient/block/subband/level. However, the reconstruction of a significant coefficient in the larger sets is more time consuming operation in DCT_SPIHT. This is because of quad or set partitioning. Therefore, the decoder of DCT_SPIHT is expected to be more complex than DTT_LSK and HLDTT.

3.3 Comparison of DTT with ICT, DCT, IWT, Filter based DWT and Lift based DWT

Generally, DCT of 8 × 8 block size is used in JPEG standard due to its superior compression performance. H.264 standard uses 4 × 4 ICT owing to its low computational complexity [37]. The transition from DCT to ICT has inflicted approximation error but the computational ease of ICT out weights the error. In addition to all the advantages of ICT, DTT has some additional advantages. These are:

- Due to polynomial nature of DTT, the transform can be converted to an integer representation without the need of intermediate scaling which is contrary to 4 × 4 ICT. Further, DTT can be implemented using multiplier free and less computationally intensive algorithms. This, not only makes DTT accurate, but also reduces computational cost.
- The dynamic range of DTT is comparable to ICT. 4 × 4 DTT has a dynamic range gain of 8. Two dimensional transform is computed by transforming rows and columns. Therefore, the total gain is $8^2 = 64$. Since $log_2 64 = 6$, 6 more bits are required to store transformed coefficients than the signal input. Therefore, DTT needs 16-bit arithmetic to compute the transform like DCT.
- DTT orthogonal polynomials basis are less sensitive to noise [63].
- DTT is invariant to linear transforms and can be efficiently used for image reconstruction.
- The compression performance of DTT is very close to DCT for natural images. For unnatural images having sharp edges and large intensity variations, DTT outperforms DCT.
- In case of video compression, the prediction residuals of motion compensation can also contain large variations. DTT can help a consistent video quality when inter-frame coding is used [65].

- A 4×4 DCT needs 74 additions and 16 multiplications [94], whereas 4×4 DTT needs 66 additions and 32 multiplications [51].
- A 4×4 ICT requires 64 additions and 16 shift operations [95] while integer transformation of 4×4 DTT requires 80 additions and 16 shift operations [52].
- The fast algorithm proposed by Cho and Lee [96] requires 96 multiplications and 466 additions for 8×8 DCT implementation. Therefore, each point requires 1.5 multiplications and 7.28 addition operations. On the other hand, computational cost of DWT depends on the length of the filters. Complexity of conventional Mallat algorithm [9] has $(\mid h \mid + \mid g \mid +2)$ multiplications and $(\mid h \mid + \mid g \mid)$ additions, where h and g are the length of the filter. Hence CDF 9/7 [90] roughly has 18 multiplications and 16 additions per point. This complexity is further reduced by 50% using lifting scheme. However, this is still large compared to DCT. There is no fast 8×8 DTT algorithm reported in the literature. Using distributed arithmetic based approach [77], a fast and low complexity 8×8 DTT can be implemented which can compete with 8×8 DCT. The reason is that DTT has similar properties like DCT and both transforms have been derived from Chebyshef polynomials of the first kind. Recently, Shu et al. [97] reported a fast Tchebichef transform algorithm which is independent of block size.
- By investing retinal receptive field, Balvias [64] has showed that orthogonal polynomials have certain properties that are similar to those of human visual systems (HVS). It is also suggested that visual analysis of retina can be regarded as a process of expansion in orthogonal polynomial basis. It has been experimentally verified that the decoded images using Tchebichef polynomials have better visual characteristics than that of DCT polynomials.
- Lossless IWT scheme presented by Sweldens [10],[91] allows low complexity and efficient implementation of DWT. The IWT mainly has three advantages.
  - It has prefect lossless reconstruction capabilities. This is particularly important for medical image processing.
  - It has lower computational complexities than DWT.
  - It reduces memory demands of compression algorithms.
- However, using IWT instead of DWT, degrades the rate distortion performance of lossy codecs (about 3-5 dB than CDF 9/7 bi-orthogonal wavelets). This is due to the fact that the transform is no more unitary and the information

content is no longer directly related to magnitude. This is particularly harmful for decoders with rate allocation based on bit planes such as SPIHT and SPECK. Mantissa-rounding operations are needed in order to confirm that the transformed coefficients should be integers for integer inputs.

- In Table 3.3, various parameters between DCT, ICT, IWT, Filter based DWT, Lift based DWT and DTT are summarized using a parameter comparison matrix. Although most of the parameters are self-explanatory, others deserve some comment. Concerning to coding efficiency, DWT is having higher coding efficiency than any block based transforms. The coding efficiency of DTT is very near to DCT. Parameter such as genericity refers to the ability to compress different types of images. DTT is quite competitive with DCT and DWT, because DCT and DWT exhibit lesser compression compared to DTT for images having higher intensity gradation. However, DCT and DWT perform best for smooth images. In wavelet based coding scheme, a separate HVS model is adopted to improve the perceptual quality of images in contrast to DTT/DCT based embedded coding schemes, where polynomials have certain HVS properties.

Table 3.3: Parameter comparison matrix. A '+' indicates that it is supported. The more '+' indicate that it is more supported. A '-' indicates separate techniques are required to support.

Parameter	Transforms					
	DCT	ICT	IWT	Filter based DWT	Lift based DWT	DTT
Lossless compression performance	++++	+++	+++++	++++	+++++	++++
Lossy compression performance	++++	++++	++	+++++	+++++	++++
Coding efficiency	++++	+++	+++	+++++	+++++	++++
Image recon. quality for un-natural images	++++	++	++	+++	+++	+++++
Image recon. quality for natural images	+++++	++++	++++	+++++	+++++	+++++
Low complexity	++++	+++++	+++	++	+++	++++
Similarity with HVS properties	+++	++	-	-	-	+++++
Separability & symmetry	+++++	+++++	+++++	+++++	+++++	+++++
Genericity	++++	+++	+++	++++	++++	++++

- In order to ensure optimum rate-distortion performance using IWT, optimal subband shift scheme (OSS) is employed [89]. In this case, each subband is multiplied by integer powers of two. This increases the PSNR performance (about 3-5 dB) of most of the medical images. OSS scheme takes the full advantages of scaling factor K of the Reversible IWT (RB-IWT) which is defined to be 1 [10]. Table 3.5 shows the PSNR performances using OOS scheme for Lena and Barbara images. It is observed that the lossy compression performance is degraded by 0.67-2.48 dB in Lena and 0.3-3.12 dB in Barbara images compared to DWT_SPIHT. In order to take the full advantage of K, the value of K is defined to be $\sqrt{2}$. This violates the very purpose of integer transforms. The PSNR performances using $K = \sqrt{2}$ are also presented in Table 3.5 for same set of images. It is observed that the performance is still 0.33-0.80 dB lower in Lena image and 0.22-1.79 dB lower in Barbara image compared to DWT_SPIHT. While comparing with HLDTT(16 × 16)[No AC], IWT_SPIHT(OOS scheme) shows a PSNR loss of 0.33-1.7 dB at 0.5-1.0 bpp in Lena image and 0.05-1.93 dB on all the considered bit rates for Barbara image. However, IWT_SPIHT(OOS scheme) shows a gain of 0.21-0.63 dB on 0.0313-0.25 bpp for Lena image.

- In order to make ICT orthonormal, the scaling matrix is to be multiplied with integer matrix. In JPEG like codecs, the scaling matrix will be merged with the quantization process so as to ensure integer transform with fast compression and decompression processes. However, the scaling matrix of ICT cannot be merged with SPIHT quantization process like in JPEG. Furthermore, if ICT is combined with SPIHT, the algorithm will similar to DCT with SPIHT. Therefore, the purpose of generating integer transformed coefficients will be destroyed because of scaling matrix.

3.4 Simulation Results and Analysis

To evaluate the performance of proposed hybrid image coding algorithms, experiments are conducted on Lena, Barbara, Mandrill, Boat, Goldhill, ruler, 256 level test pattern and Texmos1 images. The implementation is done in MATLAB 7.10.0 under Window XP, Intel Core 2 Duo CPU with 3 GHz speed and 4 GB of RAM space. The size of each image is 512 × 512.

3.4.1 Coding Performance

In order to evaluate the coding performance, we first compare the cumulative number of bits generated in different pass of DCT_SPIHT and the proposed techniques

DTT_LSK as well as HLDTT. Table 3.4 shows the cumulative number of bits generated in the first six passes of three algorithms on four different types of standard images (a) Barbara, (b) Lena, (c) Mandrill and (d) Boat.
The symbols illustrated in Table 3.4 are defined as follows:

- $A_0(S_0)$: A_0 is the cumulative number of bits generated on top n passes of DCT_SPIHT and S_0 is the number of significant coefficients corresponding to these n passes, where, $n = 1, 2, 3, ...6$.
- $A_1(S_1)$: A_1 is the cumulative number of bits generated on top n passes of DTT_LSK and S_1 is the number of significant coefficients corresponding to these n passes.
- $A_2(S_2)$: A_2 is the cumulative number of bits generated on top n passes of HLDTT and S_2 is the number of significant coefficients corresponding to these n passes.
- αA_{20} is the percentage of bit saving in HLDTT with respect to DCT_SPIHT.
- βA_{21} is percentage of bit saving in HLDTT with respect to DTT_LSK.
- BRI_{20} is the bit rate improvement of HLDTT with respect to DCT_SPIHT.
- BRI_{21} is the bit rate improvement of HLDTT with respect to DTT_LSK.

The bit rate (bpp) at n^{th} pass is give by:

$$\text{Bit rate} = \frac{\text{Cumulative number of bits generated}}{\text{Original image size}}. \tag{3.12}$$

The bit rate improvement (BRI) in bits per pixel (bpp), can be found by taking the bit rate difference between any two algorithms. For example, at n^{th} pass

$$\begin{aligned} \text{BRI}_{20} &= \text{Bit rate(DCT\_SPIHT)} - \text{Bit rate(HLDTT)}, \quad \text{and} \\ \text{BRI}_{21} &= \text{Bit rate(DTT\_LSK)} - \text{Bit rate(HLDTT)}. \end{aligned} \tag{3.13}$$

It is evident form the Table 3.4 that the proposed technique HLDTT consistently outperforms the other two. The percentage of bits saved (i.e., αA_{20}) by the proposed technique, HLDTT over DCT_SPIHT in the first six passes are about 91-96 %, 81-89 %, 60-76 %, 36-54 %, 15-23 % and 8-21 % respectively. Further HLDTT saves (i.e., βA_{21}) 58-82 %, 30-50 %, 10-23 %, 5-10 %, 1-4 % and 0.4 -0.6 % of bits in the first six bit plane passes respectively compared to DTT_LSK. Therefore, HLDTT is more efficient in sorting significant coefficients than DCT_SPIHT and DTT_LSK. One of the main reason is that clusters of zeros are more likely to occur at the early passes

Table 3.4: Cumulative number of bits generated in the first six passes of DCT_SPIHT, DTT_LSK and HLDTT techniques, % of bit savings and BRI improvements on various monochrome images.

Pass no.	$A_0(S_0)$	$A_1(S_1)$	$A_2(S_2)$	αA_{20}	βA_{21}	BRI_{20}	BRI_{21}
	Barbara						
1	449(1)	83(1)	15(1)	96.7	82.0	0.0017	0.0003
2	933(28)	251(27)	174(27)	81.4	30.7	0.0029	0.0003
3	1537(92)	677(90)	599(90)	61.0	11.5	0.0036	0.0003
4	2555(235)	1720(245)	1634(245)	36.0	5.0	0.0035	0.0004
5	6238(680)	5140(731)	5060(731)	18.9	1.5	0.0045	0.0003
6	21599(2362)	16937(2254)	16857(2254)	21.9	0.5	0.0181	0.0003
	Lena						
1	449(1)	83(1)	15(1)	96.7	82.0	0.0017	0.0003
2	924(28)	237(29)	163(29)	82.4	31.2	0.0029	0.0003
3	1557(101)	693(98)	618(98)	60.3	10.8	0.0036	0.0003
4	2550(241)	1699(241)	1621(241)	36.5	4.6	0.0035	0.0003
5	5845(663)	5005(696)	4927(696)	15.7	1.6	0.0035	0.0003
6	15197(1945)	14032(1963)	13956(1963)	8.2	0.6	0.0047	0.0003
	Mandrill						
1	449(1)	83(1)	15(1)	96.7	82.0	0.0017	0.0003
2	906(10)	186(10)	92(10)	89.8	50.5	0.0031	0.0004
3	1416(62)	436(61)	333(61)	76.5	23.6	0.0041	0.0004
4	2156(157)	1084(157)	977(157)	54.7	9.9	0.0045	0.0004
5	4257(406)	3380(444)	3275(444)	23.1	3.1	0.0037	0.0004
6	20719(2325)	18153(2232)	18050(2232)	12.9	0.6	0.0102	0.0004
	Boat						
1	445(7)	89(7)	37(7)	91.7	58.43	0.0016	0.0002
2	938(27)	234(26)	161(26)	82.8	31.2	0.0030	0.0003
3	1532(92)	589(90)	517(90)	66.3	12.2	0.0039	0.0003
4	2453(199)	1427(201)	1347(201)	45.1	5.6	0.0042	0.0003
5	6126(651)	4799(642)	4721(642)	23.0	1.7	0.0054	0.0003
6	17253(2112)	15695(2120)	15619(2120)	9.5	0.4	0.0062	0.0003

when the threshold is high. The proposed algorithm encodes these clusters more efficiently than DCT_SPIHT and DTT_LSK. It is to be noted that the number of significant coefficients for a particular transform do not vary at a given threshold. It can vary if the core transform differs. For instance, at bit plane pass 6 on Barbara image, DCT_SPIHT has 108 more significant coefficients than DTT_LSK and HLDTT. But the coding gain of DTT_LSK or HLDTT at lower bit rates are high. This is due to the fact that the encoding bit length of DCT_SPIHT is quite larger (i.e., 21.9 %) and contains many zeros than that of DTT_LSK or HLDTT. This, lowers down the coding gain of DCT_SPIHT. The BRI (bpp) of HLDTT over DCT_SPIHT shows an impressive gain compared to DTT_LSK. This indicates the inefficiency of DCT_SPIHT at higher bit plane passes (lower bit rates). Therefore, the performance of DCT_SPIHT at lower bit rates is poorer than HLDTT and DTT_LSK.

In Table 3.5, the PSNR results of four DCT based algorithms on Lena and Barbara images are quoted from the corresponding papers [4],[41],[79],[95]. These coders employed adaptive arithmetic coding which produces additional compression to the final bit stream. MRDCT [79] performs best than other DCT-based methods on Lena image, whereas, EQDCT [43] performs best on Barbara image. In order to compare the effective low bit rate gain of HLDTT(8×8), DCT_SPIHT is evaluated under the same approach as that shown in Figure 3.1. In order to compare the performance improvement for the reason of DCT, DCT is combined with SPIHT and compare with DCT_ILSK in first case. In second case, DTT is kept fixed and compare DTT_SPIHT with HLDTT coding techniques. In third case, wavelet transform is adopted and in fourth case the coding method is kept same and changed the transforms.

ILSK shows PSNR improvement of 0.08-0.55 dB with respect to SPIHT while using DCT in both the coding schemes on Lena image for the considered bit rates. Similarly HLDTT shows 0.10-0.33 dB PSNR improvement between 0.0156-0.25 bpp over DTT_SPIHT. However, the PSNR of HLDTT reduces by 0.05-0.08 dB between 0.5-1.0 bpp with respect to DTT_SPIHT. HLDTT shows an improvement of 0.01-0.15 dB on the considered bit rates with respect to DTT_LSK. JPEG2000 exhibit a coding gain of 0.07-1.13 dB on Lena image while comparing with HLDTT(16×16)[No AC]. DWT_ILSK outperforms JPEG2000 by 0.27-1.13 dB on all the considered bit rates. Similarly, DWT_ILSK outperforms DWT_SPIHT by 0.02-0.72 dB below 0.125 bpp. DWT_SPECK and IWT_SPIHT show lower performance in most of the bit rates than the DWT_ILSK. Similar trends are also observed in Barbara image. Jasper-JPEG2000 encoder [98] is used to find the PSNR values of all images while bypassing the arithmetic coding stage. Figure 3.7(a) and (b) show the the MSSIM performance improvement of ILSK algorithm over SPIHT on Lena and Barbara images using different kinds of transform techniques.

Coding results of Boat, Mandrill and Texmos1 images are presented in Table 3.6. ILSK shows a PSNR improvement of 0.07-0.8 dB with respect to SPIHT by keeping DCT fixed in Boat image. Similarly, HLDTT outperform SPIHT by 0.31-0.73 dB while keeping DTT fixed. Keeping the coding method same (i.e., ILSK), DTT shows some PSNR improvement at very low bit rates. However, the MSSIM performance due to the reason of DTT is better over DCT. This can be observed in Figure 3.7(c). Similar trends are also observed for Mandrill and Texmos1 images.

The PSNR performance of 256-level-test-pattern and Ruler images are shown in Table 3.7. While keeping the transform fixed (say, DCT), the PSNR performance of ILSK is 0.01-0.39 dB higher than SPIHT in 256-level-test-pattern image.

Table 3.5: Performance comparison (PSNR in dB) between various algorithms on Lena and Barbara Images

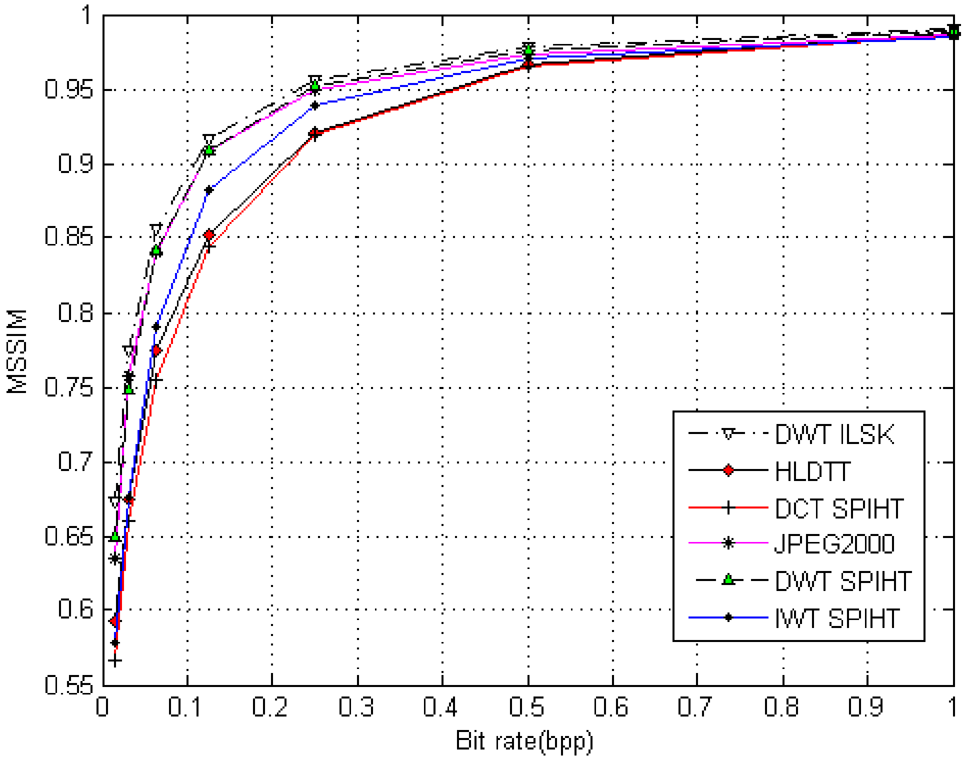
Coder	Lena(512 × 512)						
	0.0156	0.0313	0.0625	0.125	0.25	0.5	1.0
		Block	transform	(DCT)	(AC)		
EZDCT [41]	-	-	-	28.50	32.27	35.98	39.60
MRDCT[79]	-	-	-	29.26	32.55	35.99	39.49
JPEG [4]	-	-	-	28.00	31.60	34.90	37.90
EQDCT [43]	-	-	-	29.07	32.35	35.90	39.50
DCT_SPIHT [43]	22.62	24.18	26.61	28.95	32.31	36.10	39.72
DCT_ILSK	23.17	24.52	26.77	29.22	32.39	36.18	39.80
		Block	transform	(DTT)	(AC)		
DTT_SPIHT	22.72	24.20	26.51	28.89	32.19	36.00	39.61
DTT_LSK	23.10	24.53	26.75	29.21	32.29	35.95	39.53
HLDTT (8 × 8)	23.25	24.58	26.79	29.23	32.31	36.00	39.54
HLDTT (16 × 16)	23.19	24.81	27.20	29.65	32.98	36.42	39.65
HLDTT (16 × 16)[No AC]	22.47	24.31	26.44	29.12	32.30	35.72	39.21
		Wavelet	transform		(No AC)		
JPEG2000	22.58	25.06	27.47	30.25	33.27	36.30	39.28
DWT_SPIHT	22.99	25.19	27.61	30.54	33.62	36.80	39.99
DWT_SPECK	22.71	24.98	27.32	30.37	33.39	36.51	39.66
IWT_SPIHT($K = \sqrt{2}$)	22.50	24.85	27.28	29.99	32.93	36.07	39.19
IWT_SPIHT(OSS scheme)	21.95	24.52	26.98	29.75	32.62	35.39	37.51
DWT_ILSK	23.71	25.67	27.90	30.56	33.54	36.62	39.75
	Barbara (512 × 512)						
		Block	transform	(DCT)	(AC)		
EZDCT [41]	-	-	-	24.07	26.83	30.82	36.10
MRDCT [79]	-	-	-	24.22	26.84	30.61	35.89
JPEG [4]	-	-	-	23.20	25.20	28.30	33.10
EQDCT [43]	-	-	-	24.28	27.02	30.74	35.89
DCT_SPIHT [43]	21.16	22.05	22.92	24.57	27.12	30.93	36.42
DCT_ILSK	21.45	22.22	23.22	24.85	27.15	31.12	36.48
		Block	transform	(DTT)	(AC)		
DTT_SPIHT	21.25	22.18	23.05	24.52	27.09	30.82	35.98
DTT_LSK	21.42	22.21	23.20	24.80	27.03	30.82	36.20
HLDTT (8 × 8)	21.51	22.26	23.25	24.83	27.08	30.83	36.21
HLDTT (16 × 16)	21.35	22.39	23.46	24.98	27.22	31.14	36.38
HLDTT (16 × 16)[No AC]	20.77	21.98	23.15	24.61	27.03	30.63	35.75
		Wavelet	transform		(No AC)		
JPEG2000	20.81	22.34	23.39	25.06	27.82	31.42	36.49
DWT_SPIHT	21.24	22.49	23.53	24.82	27.54	31.58	36.79
DWT_SPECK	21.10	22.25	23.43	24.62	27.10	30.82	35.61
IWT_SPIHT($K = \sqrt{2}$)	20.87	22.27	23.26	24.36	26.58	30.31	35.00
IWT_SPIHT(OSS scheme)	20.50	22.13	23.20	24.24	26.44	30.13	33.87
DWT_ILSK	21.68	22.63	23.64	25.05	27.43	31.48	36.72

AC: With arithmetic coding, No AC: Without arithmetic coding

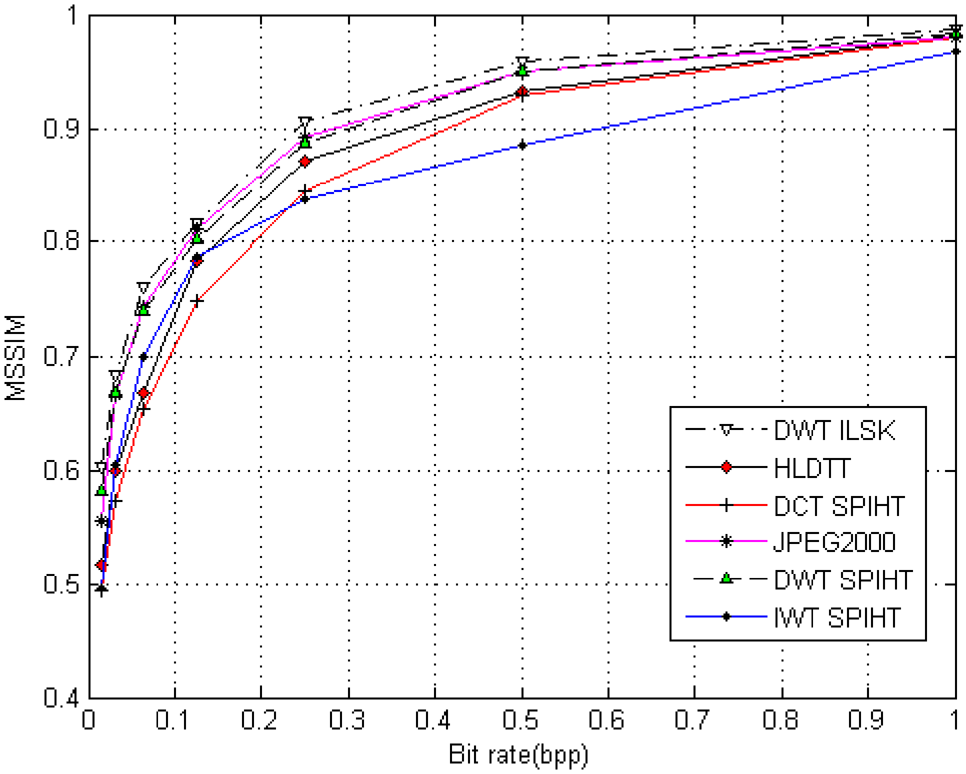
Similarly, the PSNR performance of HLDTT is 0.05-0.32 dB higher than DTT_SPIHT for 0.0156-0.5 bit rates. In case of DWT, the performance of ILSK is 0.05-0.47 dB higher than SPIHT for 0.0156-0.5 bit rates. However, JPEG2000 shows a PSNR gain up to 3.05 dB compared to HLDTT for the considered bit rates. While keeping the coding method same (say, SPIHT), DTT shows up to 0.04-0.48 dB gain than DCT for 0.25-1.0 bit rates in 256-level-test-pattern image. A slight decrease in PSNR ($\sim$ 0.05 dB) is observed below 0.25 bpp for the same image.

Table 3.8 shows the rate distortion performances of various color images. The results are based on 4:4:4 chroma sub sampling with non-interleaved color plane scan pattern. The bit rate is varied from 0.005 to 1.0 bpp to effectively show the coding efficiency of the proposed techniques at low ($\leq$ 0.1 bpp) to medium ($\geq$ 0.1 bpp to $\leq$ 0.5 bpp) bit rates. Lena image shows a very significant (0.07-3.11 dB) improvement of PSNR values at low bit rates (i.e., form 0.005-0.1 bpp). Similar improvements are also observed in other smooth images at low bit rates. At medium to higher bit rates, a slight reduction of coding gain is observed. Baboon image shows a gain of 0.05-0.91 dB at low bit rates and marginal loss at medium to high bit rates. HLDTT also shows a good improvement of coding gain over DTT_LSK at lower bit rates. Improvement at lower bit rates is highly desirable for transmission of images through a narrow bandwidth channel where a significant amount of information can be transmitted at the outset. It is worth noting form experimental results that the coding performance of color images is higher than gray scale images. The coding performance of color images can be further improved by exploiting the interdependency between color planes. Back-end arithmetic coding using contexts and joint encoding generally improves an additional gain of 0.5 to 1 dB. The same amount of improvement in HLDTT is expected as well.

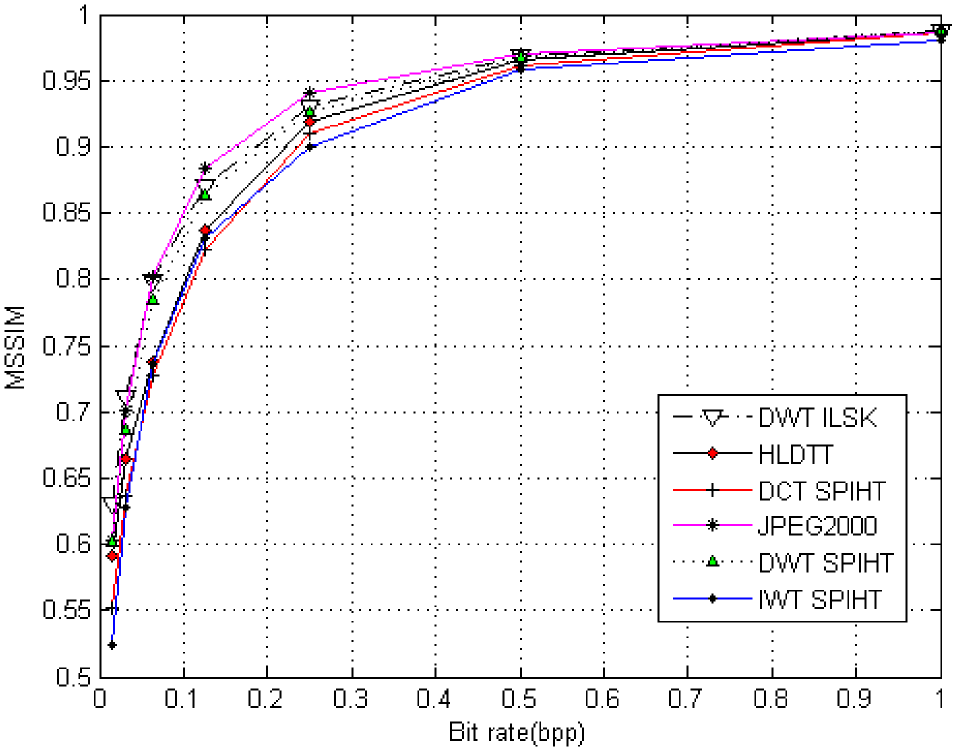
Figure 3.7 shows the MSSIM vs. bit rate plot for 6 standard test images. It can be observed from the figures that the decoded images by HLDTT consistently show better perception qualities than DCT_SPIHT decoded images irrespective of the bit rate considered. It is also well known that block based embedded coders have lower perceptual qualities than wavelet based embedded coders at lower rates. This is due to the blocking effect manifest in lower rate compressed images. Therefore, HLDTT has lower MSSIM than DWT_SPIHT and JPEG 2000 coders at lower rates. The blocking effect gradually vanishes at higher rates in block based embedded coders. This increases the MSSIM of HLDTT decoded images at higher rate. For images having sharp edges such as ruler and 256-level-test-pattern, HLDTT performs better than DCT_SPIHT and IWT_SPIHT on all the considered bit rates. The reason is that DTT has higher energy compaction properties for unnatural images than DCT.


MSSIM
Bit rate(bpp)
1
0.95
0.9
0.85
0.8
0.75
0.7
0.65
0.6
0.55
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
DWT ILSK
HLDTT
DCT SPIHT
JPEG2000
DWT SPIHT
IWT SPIHT

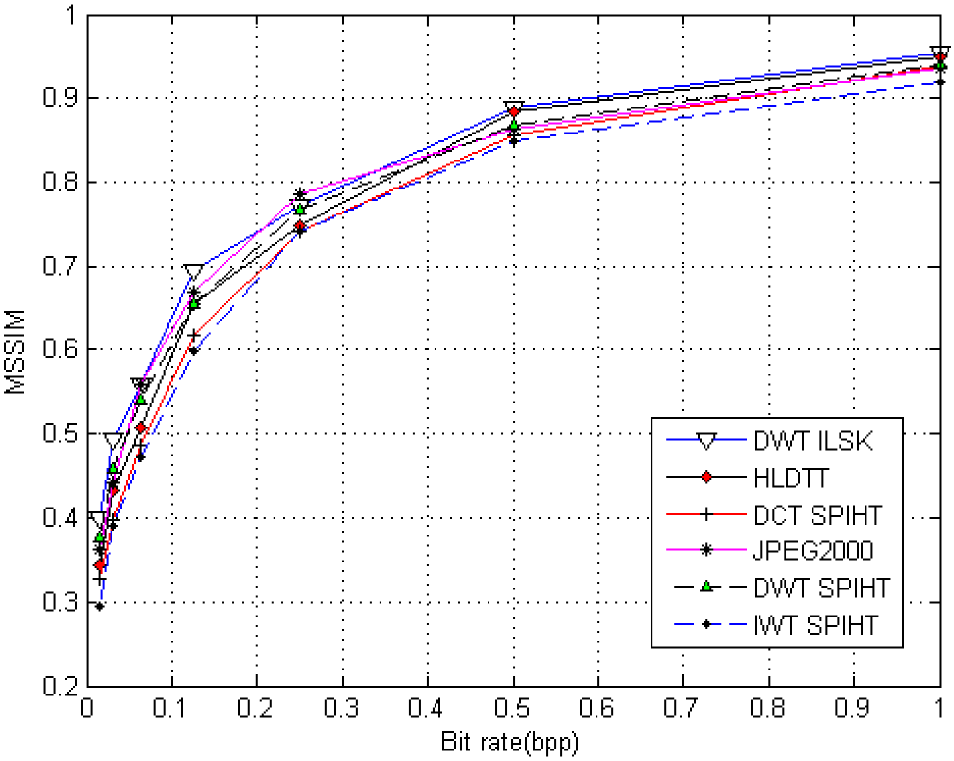
(a)


MSSIM
Bit rate(bpp)
1
0.9
0.8
0.7
0.6
0.5
0.4
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
DWT ILSK
HLDTT
DCT SPIHT
JPEG2000
DWT SPIHT
IWT SPIHT

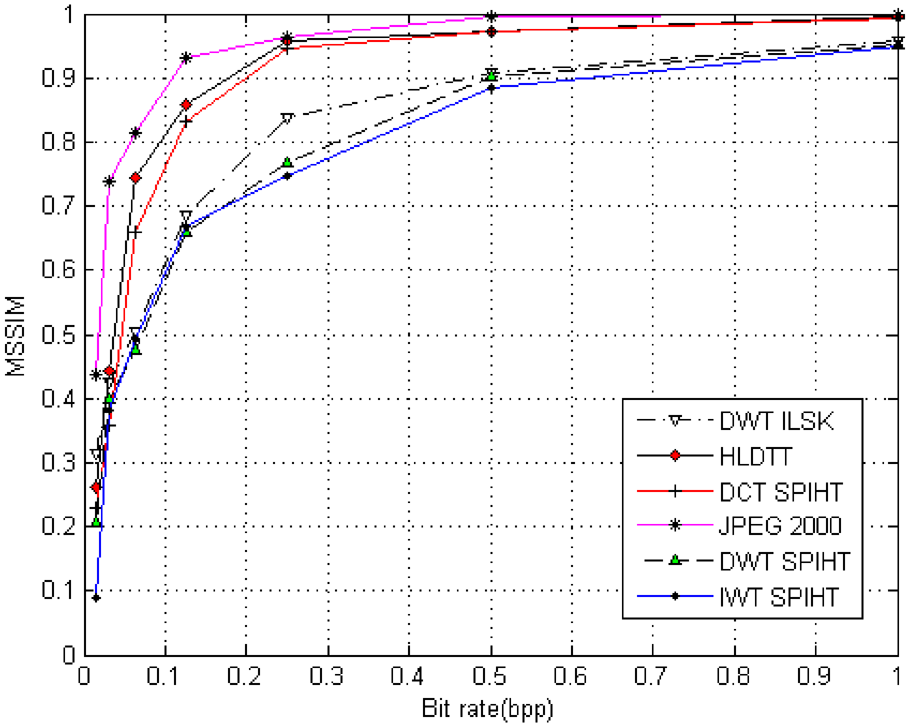
(b)


MSSIM
Bit rate(bpp)
0.5
0.55
0.6
0.65
0.7
0.75
0.8
0.85
0.9
0.95
1
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
DWT ILSK
HLDTT
DCT SPIHT
JPEG2000
DWT SPIHT
IWT SPIHT

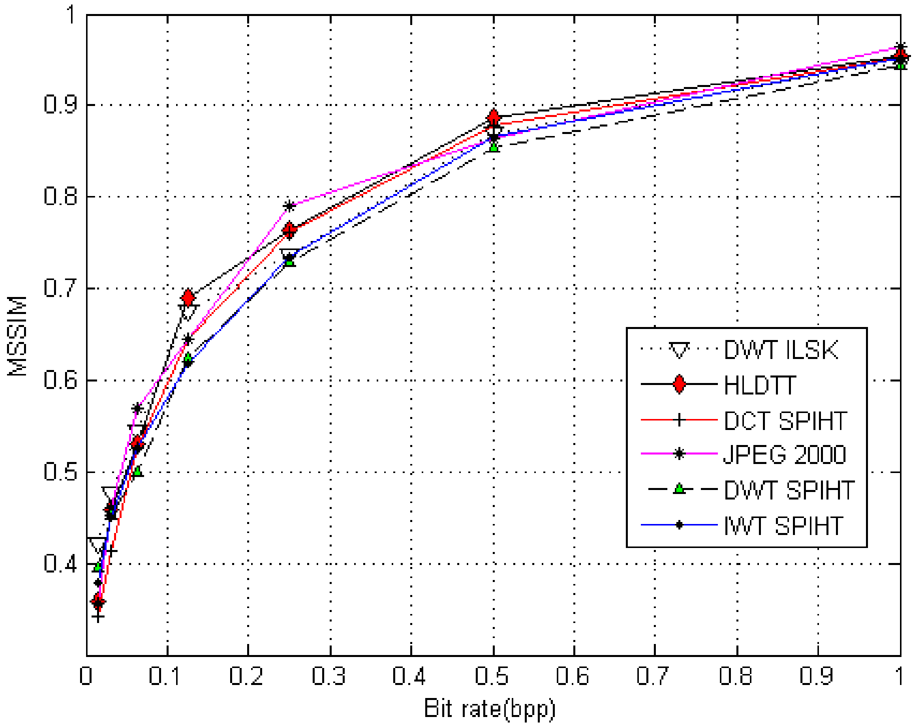
(c)


MSSIM
Bit rate(bpp)
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
DWT ILSK
HLDTT
DCT SPIHT
JPEG2000
DWT SPIHT
IWT SPIHT

(d)



(e)



(f)

Figure 3.7: MSSIM vs. bit rate on (a) Lena (b) Barbara (c) Boat (d) Mandrill (e) Ruler and (f) 256 Level Test Pattern images.

Figure 3.8 shows the decoded Barbara and Mandrill images which are compressed at 0.5 bpp using DCT_SPIHT and HLDTT algorithm. It is clearly evident that the visual quality of HLDTT decoded images are much better than DCT_SPIHT decoded images. The higher values of MSSIM in HLDTT decoded images indicate the result, even though the PSNR values of DCT_SPIHT in both the images at the specified bit rate are higher compared to HLDTT.

Figure 3.9 shows the coding efficiency of the proposed algorithms at very low (0.05 bpp) bit rate. While comparing with DCT_SPIHT decoded image (Figure 3.9(b)), HLDTT decoded image (Figure 3.9(c)) conveys a significant amount of detail information than DCT_SPIHT at much lower bit rates. This clearly demonstrates the enhanced recognizability of the proposed coders. Figure 3.9(f) is the reconstructed image by DWT_ILSK algorithm using CDF 9/7 bi-orthogonal wavelet [90]. Figures 3.9(d) and (e) are the images decoded by DWT_SPIHT and JPEG 2000 respectively. It is observed that DWT_ILSK has better PSNR gain than DWT_SPIHT and JPEG 2000. Therefore, the enhanced low bit rate performance on wavelet based coders further confirms the effectiveness of the proposed ILSK algorithm.

Figure 3.10 shows the decoded ruler image compressed at 1.0 bpp using DWT_SPIHT, JPEG2000, DWT_ILSK, DCT_SPIHT and HLDTT algorithms. It is observed that HLDTT performs significantly better over DCT_SPIHT and DWT_SPIHT by 1.85 dB and 9.34 dB respectively. Further DWT_ILSK outperforms DWT_SPIHT by 0.11 dB. This concludes that the performance of DWT/DCT based coders are poor for images having high intensity variations compared to DTT based coders. JPEG 2000 performs significantly better than all algorithms. Similar kind of improvement in performance is observed in 256-level-test-pattern image.

3.4.2 Statistical Analysis

It is can be shown that the proposed method outperforms other coding techniques by giving a probabilistic model of the average gains. Figure 3.11 shows the cropped portion of Lana, Barbara and Zelda images of two different sizes. Table 3.9 illustrates a comparison between cumulative number of bits generated in top 7 passes as well as the PSNR improvements of HLDTT over DCT_SPIHT and DTT_LSK for the cropped parts of images.

The coding gain of HLDTT can be written as:

$$Y_k(p) = Y_{HLDTT}(p) - y_k(p). \tag{3.14}$$

where $y_k(p)$ is the PSNR value at p^{th} cumulative pass when a specified coding scheme

Table 3.6: PSNR(dB) comparison of various standard gray scale images without backend arithmetic coding. All the block based transforms use blocks of size 8 × 8.

Coder	Bit rates (bpp)						
	0.0156	0.0313	0.0625	0.125	0.25	0.5	1.0
	Boat						
	Block transform (DCT)						
DCT_SPIHT	20.94	22.61	24.27	26.50	29.44	33.12	37.84
DCT_ILSK	21.74	23.04	24.67	26.81	29.58	33.23	37.91
	Block transform (DTT)						
DTT_SPIHT	21.04	22.67	24.26	26.37	29.21	32.82	37.50
DTT_LSK	21.70	23.02	24.66	26.80	29.58	33.13	37.81
HLDTT	21.77	23.05	24.67	26.81	29.59	33.14	37.81
	Wavelet transform						
IWT_SPIHT ($K = \sqrt{2}$)	21.36	23.08	25.06	27.25	29.69	33.13	37.89
IWT_SPIHT (OSS scheme)	20.54	22.57	24.47	26.63	28.96	31.91	36.08
JPEG2000	21.55	23.35	25.41	27.42	30.45	33.88	38.46
DWT_SPIHT	21.66	23.34	25.43	27.61	30.28	33.72	38.35
DWT_ILSK	22.33	23.67	25.51	27.72	30.39	33.78	38.38
	Mandrill						
	Block transform (DCT)						
DCT_SPIHT	19.02	19.55	20.13	21.08	22.44	24.70	28.19
DCT_ILSK	19.29	19.75	20.33	21.26	22.62	24.71	28.25
	Block transform (DTT)						
DTT_SPIHT	19.31	19.62	20.14	21.06	22.34	24.50	28.00
DTT_LSK	19.30	19.70	20.29	21.18	22.51	24.63	28.06
HLDTT	19.31	19.71	20.30	21.19	22.52	24.64	28.07
	Wavelet transform						
IWT_SPIHT ($K = \sqrt{2}$)	19.15	19.59	20.14	21.10	22.36	24.47	27.91
IWT_SPIHT (OSS scheme)	19.04	19.58	20.04	20.66	21.78	23.25	26.99
JPEG2000	19.35	19.77	20.44	21.33	22.81	25.07	28.64
DWT_SPIHT	19.40	19.80	20.36	21.38	22.78	25.03	28.55
DWT_ILSK	19.52	19.94	20.59	21.40	22.90	24.77	28.30
	Texmos1						
	Block transform (DCT)						
DCT_SPIHT	11.27	11.70	12.31	13.44	15.03	17.38	20.57
DCT_ILSK	11.46	11.89	12.69	13.67	15.36	17.53	20.68
	Block transform (DTT)						
DTT_SPIHT	11.35	11.77	12.40	13.44	15.00	17.12	20.20
DTT_LSK	11.48	11.88	12.66	13.59	15.18	17.20	20.27
HLDTT	11.49	11.90	12.67	13.60	15.19	17.21	20.28
	Wavelet transform						
IWT_SPIHT ($K = \sqrt{2}$)	11.20	11.64	12.42	13.55	15.19	17.25	20.33
IWT_SPIHT (OSS scheme)	11.17	11.56	12.31	13.36	14.77	16.82	19.05
JPEG2000	11.10	11.66	12.38	13.85	15.63	17.80	21.13
DWT_SPIHT	11.37	11.92	12.68	13.91	15.50	17.67	20.83
DWT_ILSK	11.56	12.05	12.91	14.05	15.76	17.84	20.88

Table 3.7: PSNR(dB) comparison of various standard gray scale images without backend arithmetic coding. All the block based transforms use blocks of size 8 × 8.

Coder	Bit rates (bpp)						
	0.0156	0.0313	0.0625	0.125	0.25	0.5	1.0
		256	Level	Test	Pattern		
		Block	transform	(DCT)			
DCT_SPIHT	14.88	15.55	16.32	17.32	19.10	21.90	27.35
DCT_ILSK	15.27	15.84	16.62	17.52	19.40	22.08	27.36
		Block	transform	(DTT)			
DTT_SPIHT	14.88	15.53	16.27	17.36	19.13	22.11	27.84
DTT_LSK	15.18	15.78	16.52	17.52	19.42	22.15	27.57
HLDTT	15.20	15.80	16.54	17.54	19.43	22.16	27.58
		Wavelet	transform				
IWT_SPIHT ($K = \sqrt{2}$)	15.16	15.70	16.16	17.03	18.32	20.99	26.51
IWT_SPIHT (OSS scheme)	15.16	15.68	16.12	17.04	18.29	21.14	26.64
JPEG2000	15.08	15.74	16.57	17.66	19.77	23.46	30.08
DWT_SPIHT	15.36	15.74	16.11	17.11	18.64	21.62	27.16
DWT_ILSK	15.50	15.86	16.50	17.34	19.11	21.67	27.03
			Ruler				
		Block	transform	(DCT)			
DCT_SPIHT	10.30	10.44	11.30	13.00	16.26	22.98	28.61
DCT_ILSK	10.43	10.82	11.78	13.44	16.38	22.95	27.96
		Block	transform	(DTT)			
DTT_SPIHT	10.30	10.49	11.58	13.75	19.27	23.14	31.05
DTT_LSK	10.38	10.88	12.02	14.32	19.42	23.20	30.45
HLDTT	10.41	10.90	12.09	14.35	19.43	23.21	30.46
		Wavelet	transform				
IWT_SPIHT ($K = \sqrt{2}$)	10.21	10.84	11.16	12.07	13.43	14.74	18.45
IWT_SPIHT (OSS scheme)	10.11	10.75	10.96	11.88	12.92	14.40	17.55
JPEG2000	10.99	12.28	13.29	15.85	23.16	35.01	42.48
DWT_SPIHT	10.37	10.30	11.10	11.80	12.84	15.64	21.12
DWT_ILSK	10.59	10.85	11.18	11.93	13.35	15.81	21.23

Table 3.8: PSNR(dB) comparison of standard color images without back end arithmetic coding

Coder	Bit rates (bpp)						
	0.005	0.01	0.05	0.1	0.2	0.5	1.0
			Lena				
DCT_SPIHT	23.78	26.36	30.66	32.94	35.74	39.83	42.85
DTT_LSK	25.68	27.24	30.83	33.00	35.68	39.48	42.51
HLDTT	26.89	27.27	30.85	33.01	35.68	39.48	42.51
			Baboon				
DCT_SPIHT	22.63	23.77	25.12	25.92	26.98	29.51	32.42
DTT_LSK	23.50	24.03	25.24	25.97	27.00	29.22	32.23
HLDTT	23.54	24.05	25.25	25.97	27.01	29.22	32.33

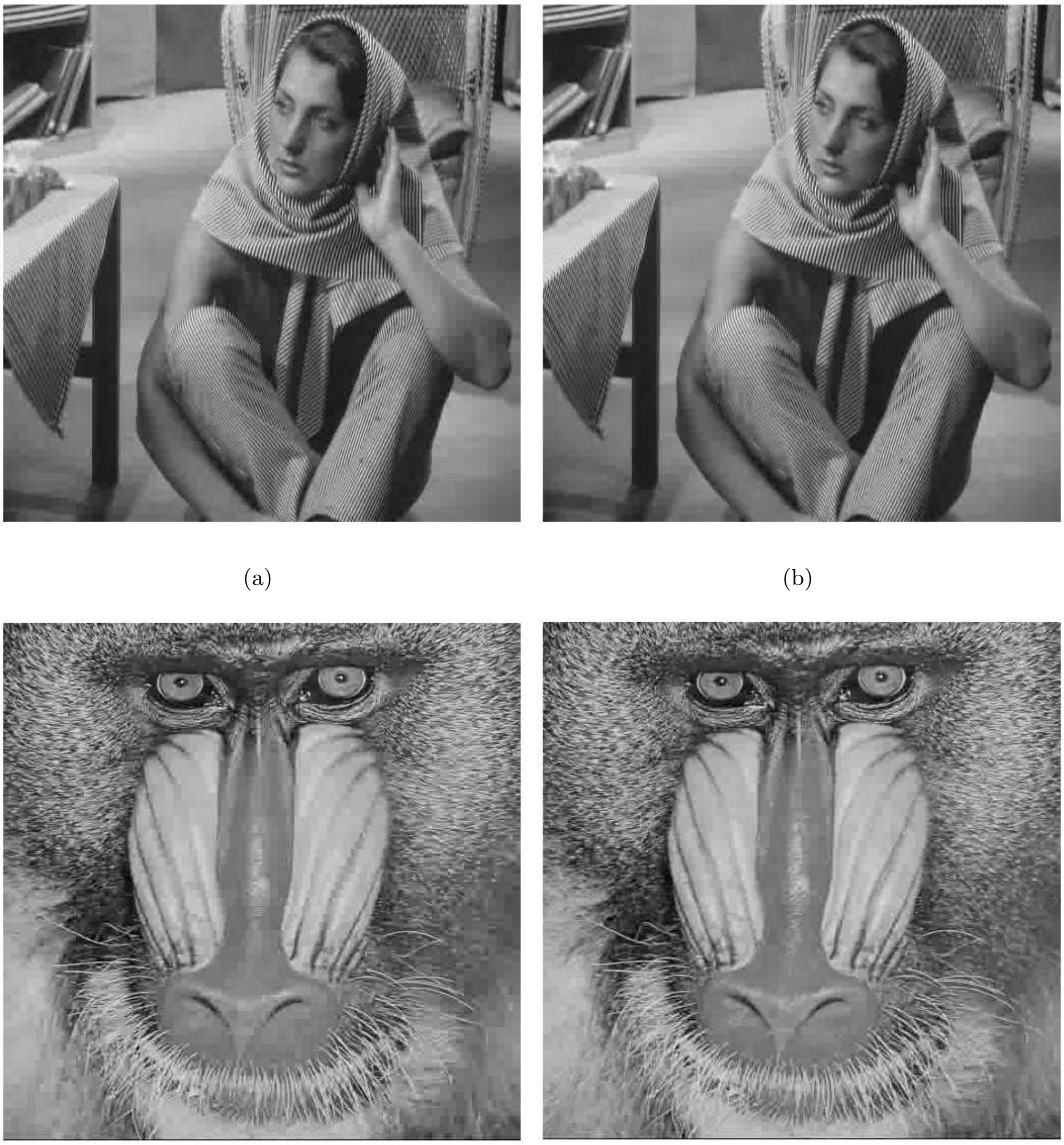
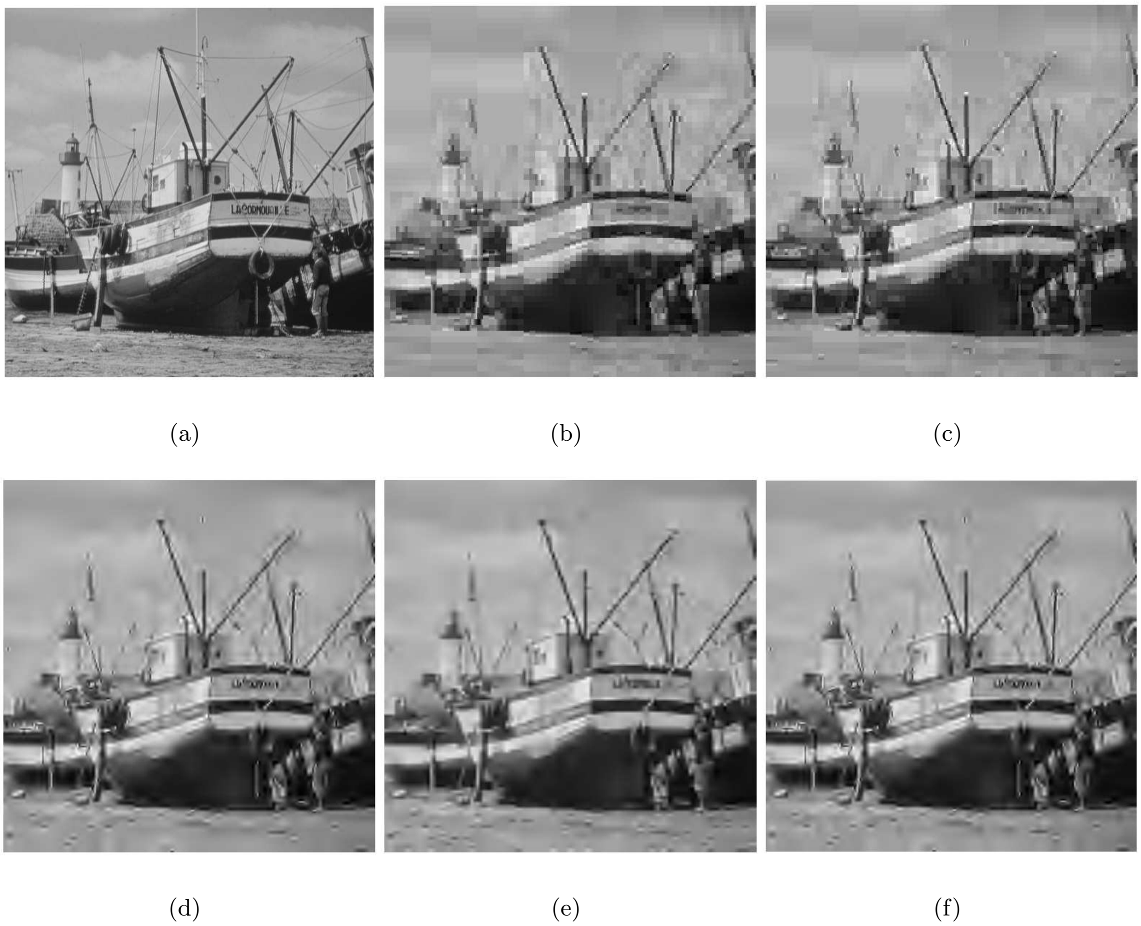

(a) (b) (c) (d)

Figure 3.8: Barbara and Mandrill images compressed at 0.5 bpp using (a),(c) DCT_SPIHT (MSSIM=0.9298 for Barbara, MSSIM=0.8572 for Mandrill) (b),(d) HLDTT (MSSIM=0.9318 for Barbara, MSSIM=0.8837 for Mandrill)

is applied and k={DCT_SPIHT, DTT_LSK}. $Y(p)$ represents the PSNR difference between HLDTT and one other coding technique. For example, $Y_{DCT\_SPIHT}(1)$ means the PSNR difference between HLDTT and DCT_SPIHT on pass no. 1. If $Y(p) > 0$, it means that HLDTT has better coding gain than the other coding schemes at the same pass number. It is assumed that $Y(p)$ is normally distributed and μ and σ are

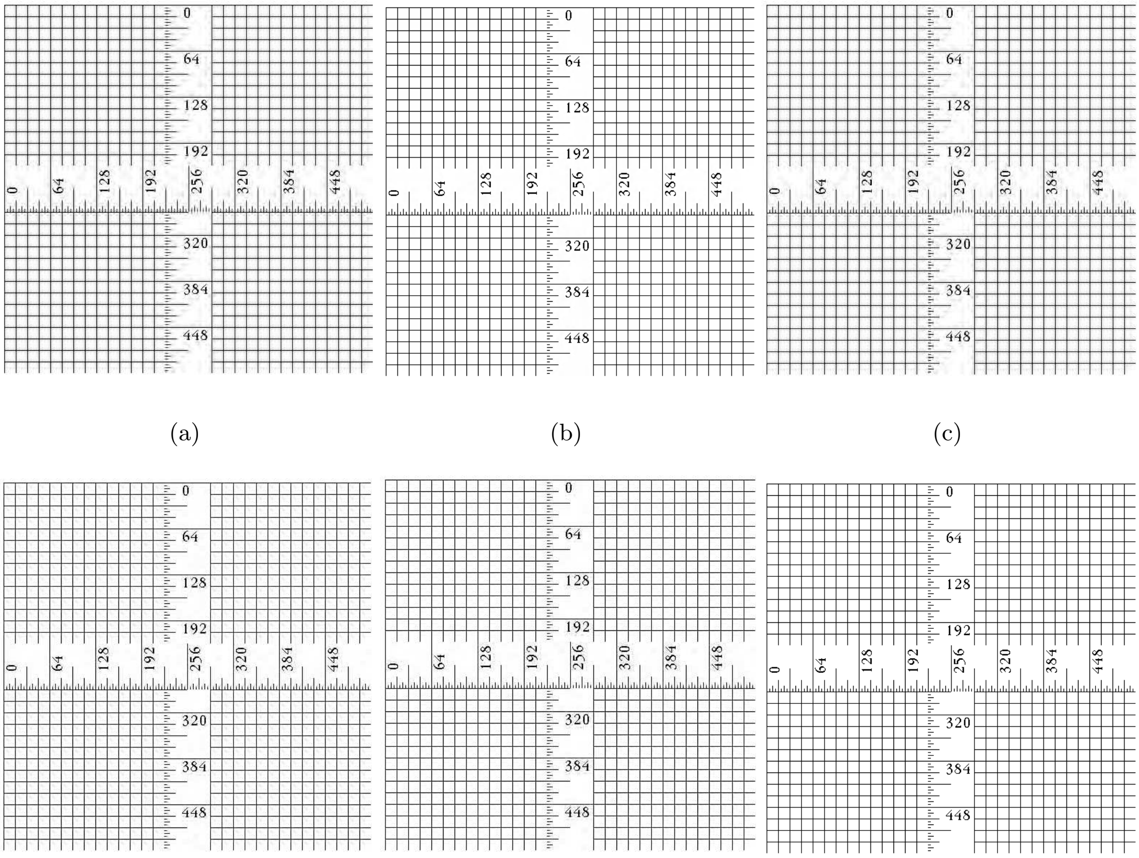

(a) (b) (c)

(d) (e) (f)

Figure 3.9: (a) Original boat image reconstructed at a target bit rate of 0.05 bpp using (b) DCT_SPIHT (PSNR=23.74 dB, MSSIM=0.6987) (c) HLDTT (PSNR=24.02 dB, MSSIM=0.7172) (d) DWT_SPIHT (PSNR=24.77 dB, MSSIM=0.7537) (e) JPEG 2000 (PSNR=24.70 dB, MSSIM=0.7754) and (f) DWT_ILSK (PSNR=24.98, MSSIM=0.7659) respectively.

mean and standard deviations. From the simulation results reported in Table 3.9, we have listed μ, σ and the corresponding 95% confidence bound. Taking a population size of 21 in each case, the following PSNR gains are obtained.

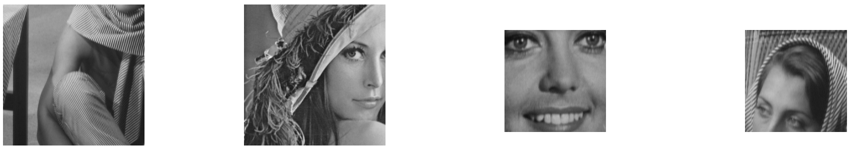
- Case_1: PSNR improvement of HLDTT over DCT_SPIHT on 256×256 images: μ=2.2547 dB, σ=2.4818 dB, 95% confidence interval=2.2547 ± 1.1698 dB.
- Case_2: PSNR improvement of HLDTT over DTT_LSK on same set of 256×256 images:
  μ=0.2798 dB, σ=0.2786 dB, 95% confidence interval=0.2798 ± 0.1313 dB.
- Case_3: PSNR improvement of HLDTT over DCT_SPIHT on 128×128 size



(a) (b) (c)

(d) (e) (f)

Figure 3.10: Reconstructed ruler images at a target bit rate of 1.0 bpp using (a) DWT_SPIHT (PSNR=21.12 dB, MSSIM=0.9530) (b) JPEG 2000 (PSNR=42.48 dB, MSSIM=0.9991) (c) DWT_ILSK (PSNR=21.23 dB, MSSIM=0.9585) (d) DCT_SPIHT (PSNR=28.61 dB, MSSIM=0.9936) (e) HLDTT (PSNR=30.46 dB, MSSIM=0.9954) and (f) Original image respectively.



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

Figure 3.11: (a) Table and left side part of Barbara (b) Hair and face part of Lena (c) Facial portion of Zelda (d) Hair and warps part of Barbara images.

images:
μ=0.3924 dB, σ=0.2857 dB, 95% confidence interval=0.3924 ± 0.1347 dB.

- Case_4: PSNR improvement of HLDTT over DTT_LSK on same set of 128 × 128 images:
  μ=0.2805 dB, σ=0.3051 dB, 95% confidence interval=0.2805 ± 0.1438 dB.

Table 3.9: Cumulative number of bits generated in the first seven p^{th} passes of DCT_SPIHT, DTT_LSK and HLDTT techniques. Symbols $A_0(S_0), A_1(S_1), A_2(S_2), \alpha A_{20}$ and βA$_{21}$ are same as defined for Table 3.2.

p^{th} pass no.	$A_0(S_0)$	$A_1(S_1)$	$A_2(S_2)$	αA_{20}	βA_{21}	$Y_{DCT\_SPIHT}(p)$	$Y_{DTT\_LSK}(p)$
Barbara(warp part), (size=256 × 256)							
1	121(5)	39(5)	23(5)	67.8	41	8.530	0.910
2	244(16)	84(16)	52(16)	78.7	38	2.320	0.418
3	363(18)	140(18)	101(18)	72.2	27.9	2.331	0.263
4	517(27)	280(29)	235(29)	54.5	16	1.431	0.241
5	747(56)	517(57)	464(57)	37.9	9.7	0.878	0.086
6	1632(174)	1237(165)	1187(165)	27.3	4.0	0.452	0.056
7	6229(626)	4374(565)	4326(565)	30.5	1.0	0.189	0.046
Lena(face and Hair portion), (size=256 × 256)							
1	120(4)	38(4)	18(4)	85	52.6	7.501	1.041
2	244(16)	84(16)	49(16)	80	41.7	2.320	0.418
3	356(16)	118(16)	66(16)	81.4	44.1	2.451	0.286
4	484(27)	211(27)	152(27)	68.6	28	1.727	0.323
5	769(69)	515(69)	450(69)	41.5	12.8	0.831	0.116
6	1748(190)	1594(210)	1531(210)	12.4	4.0	0.413	0.084
7	5494(727)	5132(723)	5071(723)	7.7	1.2	0.221	0.016
Zelda(facial portion), (size=128 × 128)							
1	36(4)	34(4)	19(4)	47.2	44.2	0.811	0.810
2	92(12)	91(9)	68(9)	26.1	25.3	0.299	0.478
3	268(38)	283(39)	259(39)	3.4	8.5	0.092	0.101
4	929(139)	930(149)	906(149)	2.5	2.6	0.063	0.205
5	2383(363)	2423(386)	2401(386)	-0.7	0.9	0.076	0.104
6	4763(737)	4773(745)	4751(745)	0.25	0.4	0.115	0.032
7	8685(1313)	8817(1337)	8795(1337)	-1.2	0.3	0.114	0.063
Barbara(Hair and warps part), (size=128 × 128)							
1	33(1)	23(1)	7(1)	78.8	69.6	0.733	0.732
2	69(4)	60(4)	37(4)	50.7	38.4	0.866	0.865
3	127(15)	118(15)	84(15)	33.9	28.8	0.450	0.420
4	401(55)	359(57)	324(57)	19.2	9.7	0.172	0.133
5	1718(208)	1466(208)	1435(208)	16.5	2.1	0.273	0.086
6	5046(714)	4749(706)	4718(706)	6.5	0.66	0.330	0.057
7	10344(1619)	10803(1734)	10772(1738)	-4.1	0.3	0.190	0.016

Therefore, it is observed form all 4 cases that there is 95% probability that the PSNR gain of HLDTT is significantly high (i.e., 0.1467-3.4245 dB) compared to other coding techniques. It is also observed from Table 3.9 that HLDTT has more number of encoded bits compared to DCT_SPIHT at pass no. 5 and 7 in Zelda (face portion) image. Similar trend is also observed in Barbara (Hair and warp part) image on pass no. 7. In spite of that, the PSNR improvement in HLDTT is higher than DCT_SPIHT. This is because the number of significant coefficients are higher in HLDTT than

DCT_SPIHT for the specified number of passes. It can be easily verified that the bit rate corresponding to pass 7 can be between 0.1-0.2 bpp for the considered set of images.

3.4.3 Computational Complexity

The run time of an algorithm depends on computational platform and number of operations. In this chapter, computational complexity is analyzed in terms of (A) Number of arithmetic and logical operations, and (B) Run time of the algorithm(s)[16],[33],[34]. These two methods are discussed below:

(A) Number of arithmetic and logical operations:

In HLDTT, markers are employed to encode the coefficients in coarsest subband. It was shown in Table 3.4 that the number of significant coefficients are very few on MSB plane in most of the natural images. Therefore, HLDTT skips a large insignificant set/block in the highest level/subband using fixed markers $MF[k]$ and $M[k]$. Unlike HLDTT, DTT_LSK and DCT_SPIHT compare and code each coefficient present in the highest subband. This indicates that the number of insignificant blocks encoded by HLDTT is lesser than DTT_LSK and DCT_SPIHT. Therefore, the initial encoding/decoding cost of HLDTT is much lesser than LSK. Assuming one comparison operation per insignificant block/set, HLDTT has significantly less than N_{LL_6} comparisons and DTT_LSK has at least $(N_{LL_6}+3L)$ comparisons with MSB plane, where N_{LL_6} is the number of coefficients in the highest wavelet level.

In case of DCT_SPIHT, LIP is initialized to all the coefficients present in the coarsest subband. LIS contains the root of all trees. Therefore, the number of coefficient comparisons is at least $\{(4 \times N_{LL_6}) + \frac{3}{4} \times (N_{LL_6})\}$ with the MSB plane. This clearly indicates that the number of comparisons in DCT_SPIHT on top bit plane pass are significantly higher compared to HLDTT. As demonstrated in Table 3.4 that the number of encoded bits is lower in HLDTT than DTT_LSK and DCT_SPIHT. This is almost lower by 21.9-96% compared to DCT_SPIHT and 0.5-82% compared to DTT_LSK in Barbara image for top six passes. Since, the number of comparisons with significant coefficients and number of insignificant blocks/sets to be encoded are directly proportional to string lengths, it is obvious that HLDTT requires lesser comparison operations than DCT_SPIHT and DTT_LSK at lower rates. The mathematical proof of the complexity (in worst case) between SPIHT, ILSK and LSK is deferred to Appendix A.

(B) Run time of the algorithms:

Computational complexity is assessed in terms of the run time of encoder and decoder. Table 3.10 shows the encoding and decoding timings in seconds for the three algorithms on Lena image at various bit rates. It is observed that the encoding/decoding time of DTT_LSK and HLDTT are approximately proportional to bit rates. i.e., doubles for doubling the bit rates. In case of DCT_SPIHT the encoding/decoding times are about 2-4 times slower below 0.125 bpp, and 5-15 times slower than HLDTT within 0.125-1.0 bit rates. The reduction of time complexity is a highly desirable feature for image transmission from hand held mobile devices where power consumption and processing speed are the limiting factors. It may be noted that the encoding time of HLDTT is slightly more than DTT_LSK. This is because of overheads required to test the significance of a group of levels (IL or IGL pass) on earlier bit plane passes. At the decoder end, this overhead is being compensated because of efficient skipping of a number of subbands. This reduces the decoding time of HLDTT compared to DTT_LSK.

Table 3.10: Encoding and Decoding times (msec) on Lena image

Coder	Bit rates (bpp)						
	0.0156	0.0313	0.0625	0.125	0.25	0.5	1.0
			Encoding	time			
DCT_SPIHT	302	371	741	1612	3529	8081	29610
DTT_LSK	119	133	210	354	620	1241	2376
HLDTT	87	137	236	392	702	1507	2898
			Decoding	time			
DCT_SPIHT	71	96	245	680	2289	5609	24602
DTT_LSK	42	71	128	233	453	867	1755
HLDTT	38	66	123	215	420	869	1750

Table 3.11: PSNR(dB) performance on Lena image with varying block sizes without backend arithmetic coding

Coder	Bit rates(bpp)						
	0.0156	0.0313	0.0625	0.125	0.25	0.5	1.0
			(16×16)	blocks			
DCT_SPIHT	21.52	23.88	26.29	29.06	32.27	35.79	39.35
DTT_LSK	22.32	24.29	26.43	29.11	32.28	35.71	39.20
HLDTT	22.47	24.31	26.44	29.12	32.30	35.72	39.21
			(32×32)	blocks			
DCT_SPIHT	18.25	20.96	24.94	28.65	32.50	36.13	39.52
DTT_LSK	18.10	21.04	24.80	28.11	31.47	35.23	38.76
HLDTT	18.16	21.14	24.82	28.13	31.48	35.24	38.76

3.4.4 Effect of Block Size

In order to understand the impact of block sizes on the performance of the proposed techniques, the coding efficiency and encoding/decoding time are measured

Table 3.12: Encoding and Decoding time of Lena image with varying block sizes

Coder	Bit rates(bpp)						
	0.0156	0.0313	0.0625	0.125	0.25	0.5	1.0
	Encoding time(msec)						
	(16 × 16) blocks						
DCT_SPIHT	228	360	700	1447	3506	8691	30602
DTT_LSK	91	128	210	364	656	1233	2369
HLDTT	86	136	238	416	783	1520	2881
	(32 × 32) blocks						
DCT_SPIHT	211	341	653	1317	2746	7561	31901
DTT_LSK	88	111	117	338	639	1263	2433
HLDTT	116	138	231	422	774	1562	3017
	Decoding time (msec)						
	(16 × 16) blocks						
DCT_SPIHT	60	98	219	617	1963	6837	28456
DTT_LSK	44	73	129	235	455	899	1800
HLDTT	39	67	125	235	453	917	1774
	(32 × 32) blocks						
DCT_SPIHT	54	95	204	532	1613	5406	28604
DTT_LSK	43	74	134	249	464	888	1750
HLDTT	39	65	123	234	443	575	1720

with blocks of 8 × 8, 16 × 16 and 32 × 32 sizes. The results of PSNR values, encoding and decoding times using 8 × 8 blocks on Lena image are summarized in Table 3.5 and 3.10 respectively. The results of 16 × 16 and 32 × 32 blocks on Lena image are summarized in Table 3.11 and 3.12. It is observed that as the block size increases from 8 × 8 to 16 × 16, the rate distortion performance from medium to higher bit rates increases. This is because as the bit rate increases, the number of significant map increases with increase of block size. For low bit rates, the performance deteriorates because the number of significant map reduces. Block size of 32 × 32 shows superior performance over 8 × 8 and 16 × 16 block sizes at higher bit rates ($\geq$ 0.5 bpp) on DCT_SPIHT algorithm. The reason is that the number of significant map increases in DCT than DTT with block of large sizes. Therefore, DCT with SPIHT show superior performances with increase of block size as well as with increase of bit rates. The encoding and decoding times show some marginal improvement with increase of block size on the proposed coders. Similar trends are also observed for other standard images.

3.5 Conclusions

In this chapter, two low complexity embedded image coders DTT_LSK and HLDTT are presented. The image reconstruction performance, encoding and decoding times of both gray scale and color images are compared with some state-of-the-art DCT based embedded coders. HLDTT exploits the inefficiency of DTT_LSK and DCT_SPIHT coders at low bit rates by efficiently encoding large clusters of zero blocks at initial bit plane passes. The proposed coder HLDTT not only shows a significant improvement in PSNR values at lower bit rates but also shows a significant reduction of encoding and decoding time over DCT based embedded coders. It is also demonstrated that HLDTT shows better MSSIM performance even at higher rates, though the PSNR values are slightly lower than DCT_SPIHT in most of the images. In order to have a fair comparison with wavelet based SPIHT and JPEG2000, ILSK is combined with DWT (acronym DWT_ILSK). It is observed that DWT_ILSK shows significant improvements in PSNR/MSSIM over DWT_SPIHT and JPEG2000 at very low bit rates. The proposed coders do not use list structures like DCT/DWT based embedded coders. This reduces the dynamic memory requirement by 91.17% compared to SPIHT. Further, HLDTT provides better coding efficiency, low computational complexity and reduced memory requirements. These features facilitate HLDTT to be an attractive candidate for image communications where limited channel capacity, stroge and computational complexity would be decisive factors.

Chapter 4

Reduced Memory Listless Scalable Embedded Image Compression Algorithms

Preview

In this chapter, two simple yet efficient embedded block based image compression algorithms has been proposed. The first algorithm compresses gray scale images, whereas the second algorithm is for the color images. The proposed algorithms not only improve the rate distortion performance of set partitioning in hierarchical trees (SPIHT) and set partitioning in embedded block coder (SPECK) at lower bit rates, but also reduce the dynamic memory requirement by 91%. The former objective is achieved by encoding a large cluster of zeros using few symbols at earlier bit plane passes and the later objective is realized by improved listless implementation of the algorithms. The proposed algorithms explicitly perform breadth first search like SPECK. In addition, the algorithms generate feature rich bit streams which are absent in SPIHT/SPECK. The reduction of encoding and decoding times as well as improvement in coding efficiency at lower bit rates make these coders attractive candidates for real time multimedia applications.

4.1 Introduction

Some of the embedded coding algorithms which are primarily designed for coding gray scale images are simply extended for coding of three planes of color images [26], [99]. For efficient coding, usually color images are transformed to luminance-chrominance (YUV or YCbCr) color planes. The extended color coding algorithms for SPIHT and SPECK are CSPECK [26] and CSPIHT [99] respectively, which generate fully embedded bit streams. However, being tree based algorithm, these are memory intensive. Block based coder such as EBCOT [27] generates feature rich bit stream with low memory requirements, but it is highly complex because of multiple coding passes per bit plane, use of context adaptive arithmetic coding and post compression rate distortion optimization. Block tree algorithm presented in [34] shows a significant improvement in coding efficiency by integrating set partitioning strategies of hierarchical trees and blocks into a single algorithm. However, the encoder and decoder are more complex because of additional steps of set partitioning.

Therefore, two low complexity wavelet based block partitioning algorithms has been proposed. These are Listless Embedded Block Partitioning (LEBP) for gray scale images and Color Listless Embedded Block Partitioning (CLEBP) for color images. These algorithms encode large clusters of zero blocks (e.g., many wavelet subbands) by few symbols in contrast to SPECK so as to improve the low bit rate performance. The proposed algorithms incorporate the feature of LSK to reduce the overall memory requirements. Although wavelet based image coding algorithms supports signal-to-noise-ratio scalable and progressive (by quality) image compression, these algorithms do not support spatial scalability and do not provide bit stream which can be adapted easily according to the type of scalability desired by the user. The ability to reorder the encoded bit stream into arbitrary spatial resolution and signal-to-noise-ratio (SNR) is an important feature to access images in heterogeneous networks having large variation in bandwidth and user device capabilities. Danyali and Mertins [30] proposed scalable SPIHT suitable for heterogeneous networks. However, multiple list array structure for each spatial resolution may be a memory intensive. Xie *et al.* [31] enabled SPECK to have full scalability (known as S-SPECK) based on the idea of quality layer formation similar to post compression rate distortion (PCRD) in JPEG 2000. It uses multiple list array structure like FS-SPIHT for set partitioning. Furthermore, the seek time of a block in S-SPECK depends on the block position. S-EZBA [100] uses less number of overhead bits compared to S-SPECK. Our proposed scalable LEBP (S-LEBP) uses marker table instead of list arrays. The same set of markers in the table are reused for encoding region of interest (ROI) unlike multiple list arrays in FS-SPIHT ans S-SPECK. The use of markers in stead of multiple list arrays reduce the memory intensive operation without sacrificing the PSNR performance. Further, S-LEBP uses the idea of bi-section method [101] to reduce average seek time and less number of overhead bits to improve compression efficiency.

In order to test the efficiency with DCT based embedded coders presented in [39]-[44]. The combination of LEBP with DCT with varying block sizes has been made. By using variety of test images, it is observed that the proposed coder (acronym DCT_LEBP) outperforms most of the state-of-the-art DCT based embedded coders.

4.2 Listless Embedded Block Partitioning Algorithm for Gray Scale Images

The LEBP algorithm is quite similar to ILSK algorithm proposed in Chapter 3. The only difference is that LEBP uses Lifting wavelet transform instead of discrete Tchebichef transform. LEBP is inherently resolution scalable in contrast to HLDTT because LEBP uses wavelet transformed coefficients.

4.2.1 Comparison with LSK and NLS Algorithms

The main differences between LSK and LEBP are:

1. The encoding technique used in coarsest (LL_6) subband.
2. The method of skipping large clusters (i.e. several/single wavelet levels) of insignificant coefficients.
3. Significant memory reduction at initial passes. The reason is that M[k] skips MF[k] and MV[k] markers corresponding to lower insignificant wavelet levels if all the coefficients are insignificant.
4. LEBP provides additional desirable features such as resolution scalability and random access decodability to the compressed bit stream by simply reordering the bit string according to scale and resolution level.

While comparing with NLS algorithm,

1. LEBP does explicit breadth first search (zero blocks), while NLS does depth first search (zero trees).
2. The markers are placed in the lower levels of insignificant trees in NLS, whereas markers are placed in leading indices of each subband/level in LEBP.
3. The markers are updated during block partitioning in LEBP, whereas the markers are updated during tree partitioning in NLS.
4. Scalability and random access decodability features are introduced in LEBP. Whereas, these features are lacking in NLS.

4.2.2 Memory Allocation

In LEBP, the precomputed maximum length array, L_{max} has length $[I - (2^L + 1)]$ similar to LSK. If W bytes are needed for each sub band coefficients, then the bulk storage memory required is IW for the subband data and $RC/2$ (each marker is of half byte) for the state table $MV[k]$. For L level of wavelet decomposition, $MF[k]$ needs $\lceil (3L+4)/2 \rceil$ bytes (the number of fixed markers are $(3L+4)$ and each marker is of half byte) and $M[k]$ state table needs $\lceil (L/2) \rceil$ bytes. Therefore, the total memory required for LEBP is:

$$M_{LEBP} = IW + RC/2 + \lceil (3L+4)/2 \rceil + \lceil (L/2) \rceil. \tag{4.1}$$

The memory required for LSK is:

$$M_{LSK} = IW + RC/2 \tag{4.2}$$

It is to be noted that $MF[k]$ markers do not update. This is used as a reference for $MV[k]$ markers to check the significance of a level/subband. When comparing Eqn. 4.1 with Eqn. 4.2, M_{LEBP} is $(2L+2)$ bytes higher than M_{LSK}. Assuming L=6, it is only 14 bytes.

Table 4.1: Distribution of wavelet coefficient magnitudes along thresholds and frequency levels(Highest: 6, Lowest: 1) for Lena(512×512) image

Threshold, 2^n	Magnitude range	Frequency level					
		6	5	4	3	2	1
2^{12}	$2^{12}, ..., (2^{13}-1)$	1	0	0	0	0	0
2^{11}	$2^{11}, ..., (2^{12}-1)$	23	0	0	0	0	0
2^{10}	$2^{10}, ..., (2^{11}-1)$	45	9	0	0	0	0
2^{9}	$2^{9}, ..., (2^{10}-1)$	92	81	15	0	0	0
2^{8}	$2^{8}, ..., (2^{9}-1)$	104	158	131	30	1	0
2^{7}	$2^{7}, ..., (2^{8}-1)$	122	212	395	343	43	0
2^{8}	$2^{6}, ..., (2^{7}-1)$	119	291	644	950	615	5

4.2.2.1 Dynamic Memory Comparison

Comparing Eqn. 4.1 and Eqn. 4.2, it is concluded that M_{LEBP} requires slightly more memory than M_{LSK} at the end of all passes. However, the memory required by M_{LEBP} is significantly lower than M_{LSK} in top 6 bit plane passes. Table 4.1 shows the distribution of wavelet coefficient magnitudes along thresholds and frequency levels for Lena image. This indicates a few coefficients are significant at higher threshold level while comparing with lower levels of threshold. For example, only 1 coefficient is significant above threshold 2^{12}, but a total of 2624 coefficients are significant above threshold 2^8. By using this statistical property (i.e., decaying coefficient spectrum on higher passes) markers M[k] in LEBP algorithm skip the markers (MF[k] and MV[k]) corresponding to several lower subbands/levels on top six passes. An estimation of approximate memory requirement between M_{LEBP}, M_{SPIHT} and M_{LSK} on top 7

Table 4.2: Comparison of memory requirement between LSK, SPIHT and ILSK algorithms on top 7 passes.

No. of sorting passes	Lena Image		
	M$_{LSK}(kB)$	M$_{SPIHT}(kB)$	M$_{ILSK}(kB)$
1	128	0.8985	0.0313
2	128	0.9004	0.125
3	128	0.9375	0.5
4	128	5.0088	2.0
5	128	10.039	32.0
6	128	43.872	32.0
7	128	82.532	128.0

passes is shown in Table 4.2.

It is observed in Table 4.3 that the significant coefficients above most significant bit (MSB) plane, lie in top $1/4^{th}$ portion of highest level in Lena image. An excellent demonstration regarding the location of significant coefficients in various bit plane passes is presented in [101]. The M[k] marker, which points to leading node of next subband skips MF[k] and MV[k] markers pointing to the lower levels/subbands. So, LEBP algorithm simply skips to next lower significant bit plane from the MSB plane. Therefore, the worst case dynamic memory required is $\frac{8\times8}{2} = 32$ bytes for MSB plane, since the top $1/4^{th}$ of highest level is having a dimension of 8×8. The memory required by other passes is calculated in a similar procedure. The maximum bit rate correspond to 6 passes of LEBP can be up to 0.05-0.06 bpp, where the significant detail of an image can be visualized. M_{LSK} needs a fixed amount of memory, i.e, $\frac{RC}{2}$ (assuming 4 bits per marker in 6 levels of wavelet decomposition), irrespective of the number of passes [36]. However, the memory required by the proposed algorithm LEBP at higher passes are fixed (typically 6 and above passes it is approx. same as that of LSK) because most of the coefficients at finest subbands are likely to become significant. It may be noted that while making dynamic memory comparison, the memory required for initialization of markers/lists are not considered.

In SPIHT coders, the number of elements in auxiliary lists represent the required dynamic memory. The SPIHT uses LIP, LIS and LSP arrays. LIS additionally requires type 'A' or 'B' information to recognize the nodes.
Let, N_{LIP} is the number of nodes in LIP, N_{LSP} is the number of nodes in LSP, N_{LIS} is the number of nodes in LIS, and W is the number of bits to store the addressing information of a node. Then, the total memory required (in bytes) due to auxiliary lists is [33]

$$M_{SPIHT} = [W(N_{LIP} + N_{LIS} + N_{LSP}) + N_{LIS}]/8 \tag{4.3}$$

Since, the memory size increases in each bit plane pass, in worst case the values of Eqn. 4.3 becomes

$$N_{LIP} + N_{LSP} = M \times N, N_{LIS} = (M \times N)/4. \tag{4.4}$$

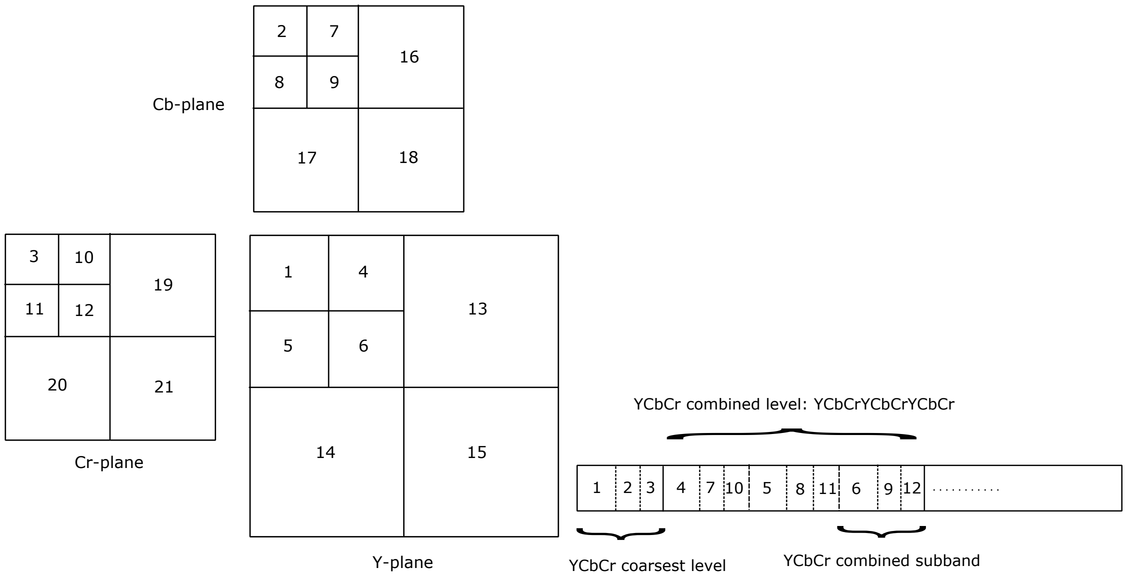
For a 512 × 512 image using 2 bytes per coefficient and 6 levels of wavelet like arrangement, while using the optional pre-computed maximum length array, the worst case memory requirement is $\frac{(512\times512)}{2} \times (8\text{bits}) + (2L+2) \simeq 128$ kB for LEBP, 128 kB for LSK and 1450 kB for SPIHT. Therefore the proposed algorithm is suitable for a fast and simple implementation. The memory required for wavelet transform is not calculated here. This part can be efficiently handled by Lifting wavelet transform [10],[102].

4.3 Listless Embedded Block Partitioning Algorithm for Color Images

The proposed CLEBP algorithm for color images uses two types of state table markers in contrast to three markers in LEBP algorithm. These are $MF[k]$ and $MV[k]$, where k's are starting indices of each wavelet levels calculated in a zigzag scan manner. Each marker holds 4 bits per coefficients to keep track of set partitions.

$MF[k]$ markers are used to keep track of these composite levels in CLEBP algorithm. If a composite level is insignificant, entire level can be skipped at once by coding using a single zero in contrast to 9 zeros in CSPECK and LSK. $MV[k]$ markers keep track of set partitions within a composite level. At any given time, $MV[k]$ markers are updated by splitting the wavelet composite level if a wavelet level is significant with respect to threshold. The meaning of each symbols are same as described for HLDTT algorithm in Chapter 3.

Considering YCbCr color space (4:2:0 format), each plane is separately wavelet transformed using 9/7 filter [90] to the same number of decomposition levels. The Figure 4.1(a) shows two levels of wavelet decompositions for ease of illustration whereas, four levels of decompositions are used in our simulations for (512 × 512) and CIF images. In QCIF images, three levels of wavelet decomposition are used. Wavelet subbands of Y, Cb and Cr planes are interleaved using the numbering pattern as shown in Figure 4.1(a). Then, two dimensional subbands are mapped to one dimensional arrangement as in Figure 4.1(b). At higher bit plane passes, this arrangement facilitates to encode a single symbol to a composite level, in contrast to many symbols required to encode in CSPIHT or CSPECK algorithm.



(a) (b)

Figure 4.1: (a) Numbering pattern across correlated wavelet subbands in 4:2:0 YCbCr color planes (b)One dimensional arrangement of Y, Cb and Cr subbands to form composite levels.

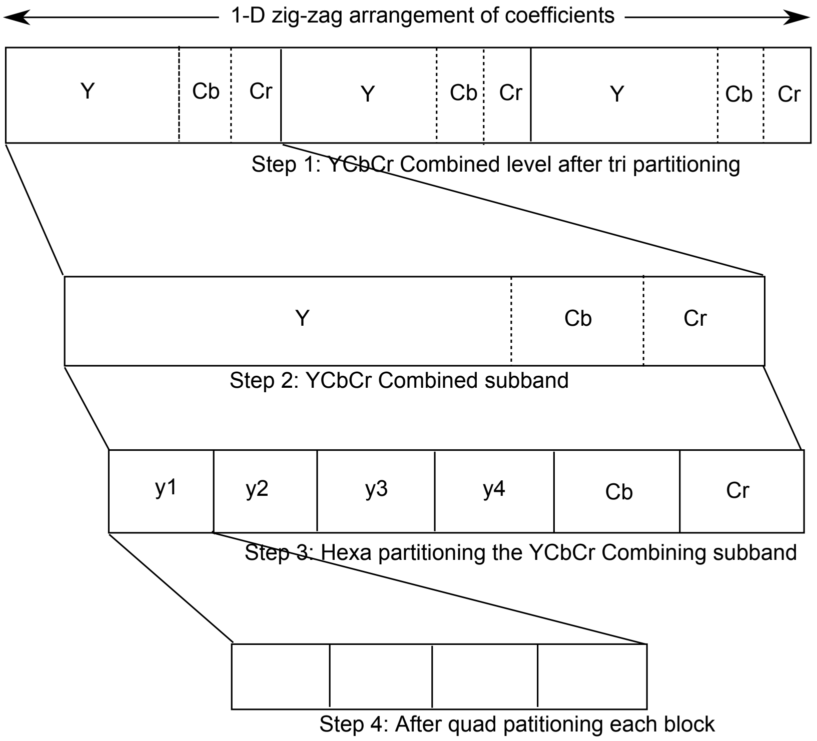

1-D zig-zag arrangement of coefficients
Y
Cb
Cr
Y
Cb
Cr
Y
Cb
Cr
Step 1: YCbCr Combined level after tri partitioning
Y
Cb
Cr
Step 2: YCbCr Combined subband
y1
y2
y3
y4
Cb
Cr
Step 3: Hexa partitioning the YCbCr Combining subband
Step 4: After quad patitioning each block

Figure 4.2: Steps for partitioning a combined wavelet level

4.3.1 Pseudocode of CLEBP Algorithm

```
Pass1 : Insignificant Pixel Pass :
    Same as Pass1 of HLDTT algorithm.
Pass2 : Insignificant Set Pass :
  1.  If, $(MV[k] = MF[k])\&(MV[k] \neq SIP)$        (Test significance
                                                     of a combined level)
        If k ∈ (YCbCr coarsest level)
        output($d = \Gamma_n(L_L)$).
          If d, Split $L_L$ by 6 parts.
          Else, Move to the next combined level.
        Else, output($d = \Gamma_n(L_L)$).
          If d, Split the combined level into 3 parts.
          Else, move past the combined level.
  2.  If, $MV[k] = (MF[k] - 1)\&MV[k] \neq SIP$        (Test significance
                                                     of a combined subband)
        if k ∈ (YCbCr combined subband)
        output($d = \Gamma_n(val[k : end of (L_L)])$).
          if d, split $L_L$ by 6 parts.
          Else, move to next combined level.
  3.  If, $(MV[k] \neq MF[k])\&(MV[k] \neq SIP)$        (Test significance
                                                     of a block/pixel)
        output($d = \Gamma_n(subband/block)$)
        if d, QuadSplit( ).
          if (block size = pixel size)
            Move to Pass1.
          Else, move past the subband/block.
Pass3 : Refinement Pass :
      If $MV[k] = SSP$
        Output($\Gamma_n(val[k])$)
        Move past the pixel.
      Elseif, $MV[k] = SNP$
        $MV[k] = SSP$
        Move past the pixel.
      Elseif, Check all the steps of (1, 2, and 3) of Pass2.
      Else, move to the next subband/block/set.
```

In the pseudocode of the algorithm, the meaning of the following points are defined as

- $(MV[k] = MF[k])$ & $(MV[k] \neq SIP)$ indicate YCbCr combined level is (Figure 4.1(b))to be tested for significance.
- $(MV[k] = MF[k] - 1)$ & $(MV[k] \neq SIP)$ indicate YCbCr combined subband is to be tested for significance.
- $(MV[k] \neq MF[k])$ & $(MV[k] \neq SIP)$ indicate a block within YCbCr combined subband is to be tested for significance.

Partitioning rule

The partitioning rule of a YCbCr combined level is shown in Figure 4.2. In step 1, if any wavelet coefficient is found significant for a certain threshold, the entire

YCbCr combined level is partitioned in to 3 parts. Each part is a YCbCr combined subband which is shown in step 2. Further, the combined subband is split in to 6 parts if any coefficient is found significant to the same threshold level as shown in step 3. Proceeding from step 4, each partitioned part recursively quad split to search for significant coefficient. Then, each significant coefficient is coded and transmitted along with the sign bit. The entire partitioning rule is presented in terms of pseudocode in section 4.3.1.

In SPECK algorithm, initially the image is partitioned into two sets, i.e., set S and set I. If set S is significant with respect to current threshold level, it is quad split, otherwise a single symbol (e.g., zero) is being coded for set S. Further, set I is tested for significance. if I is significant, it is partitioned into three S type contiguous sets and a new I type set. At the same resolution level, if these three new S type sets are insignificant with respect to current threshold, these will be coded with three zeros. In the case of color images, $3 \times 3 = 9$ insignificant subbands will be coded by 9 zeros in CSPECK algorithm. The proposed CLEBP algorithm codes these 9 insignificant subbands by a single symbol (i.e., zero). This, saves a considerable amount of symbols at early bit plane passes. As the coding process proceeds towards the finer resolution, the proposed CLEBP algorithm performs like CSPECK.

4.3.2 Complexity Analysis

The complexity of a coding algorithm depends upon target rates, number of bit planes to be processed and the number of list arrays to be processed. Encoding algorithm performs basic operations such as memory access, magnitude comparison against a threshold and some input/output operations. On the other hand, decoding algorithm performs bit manipulations, input/output and memory access operations. It is to be noted that the decoding algorithms are faster than encoding algorithms because:

- No comparison operations are required.
- Efficient skipping of insignificant sets/blocks/subbands.

The complexity of an algorithm also depends upon target rate as more number of operations are required at higher target rates.

In the list based coders, the processing of the lists accounts for much heavier computations. SPIHT, SPECK and proposed algorithms (LEBP/CLEBP) are bit plane based which require multiple passes across bit planes to reconstruct a coefficient. SPIHT and SPECK algorithms access coefficients multiple times through lists to reconstruct itself. Coefficients in LSP of SPIHT are further processed by the refinement pass. A bit of wavelet coefficient is added to the compressed bit stream for each entry of LSP.

In SPIHT, the information about the significance of a coefficient is managed by LIP. A coefficient once entered in LIP remains there until it becomes significant and moves to LSP. The proposed coders perform an explicit breadth first search without using lists. State informations are kept in fixed size arrays to enable fast scanning of bit planes. In LEBP/CLEBP, special markers are placed on the lower nodes of subbands or levels or group of levels instead of searching for the trees to find predictable insignificance. These markers are updated when new significant sets/blocks are formed by magnitude comparison. With these sparse marking, a large sections of blocks (subband, level or group of levels) are skipped at once as the breadth first scan moves through the lower nodes of wavelet decomposition array. If a block is insignificant with respect to a threshold, it is just skipped; otherwise, it is partitioned recursively until a significant coefficient is found. The probability that a block/subband is insignificant at higher bit planes is relatively high. Therefore, the proposed algorithms have the advantages of quickly finding significance of a coefficient by recursively partitioning scheme and efficient block skipping. Hence, it has the advantage of memory access time. This indicates that the encoder is faster than SPIHT and SPECK.

At the decoder, significance of a coefficient is reconstructed from the received bit map. When a coefficient/block/subband/level is insignificant, the decoder skips all operations and moves to test the significance of the next coefficient/block/subband/level. However, in SPIHT, the reconstruction of a significant coefficient in the larger sets is a more time consuming operation. This is because of quad or set partitioning. Therefore, the decoder of SPIHT is more complex than LEBP.

4.4 Simulation Results and Analysis

The performance of the proposed LEBP and DCT_LEBP coding algorithms are evaluated on two sets of standard monochrome images (8 bits/pixel). The first set includes Barbara and Lena images each of size 512 × 512. The second set includes two higher resolution images from JPEG 2000 test set, which are Bike and Woman images of size 2048 × 2560. The performance of the proposed algorithm is evaluated in terms of coding efficiency and encoder/decoder complexity. Since the coders are embedded, the results of various bit rates are obtained from a single encoded bit stream.

Investigation on the effects of various block sizes to the performance of DCT_LEBP algorithm and performance comparison with some state-of-the-art DCT based embedded coders has been made. The proposed image coding algorithms are implemented

Table 4.3: Cumulative number of bits generated in the first six passes of SPIHT, LSK and proposed LEBP technique

No. of sorting passes	Barbara						
	$A_0(S_0)$ SPIHT	$A_1(S_1)$ LSK	$A_2(S_2)$ LEBP	αA_{20}	βA_{21}	BRI_{20}	BRI_{21}
1	455(1)	85(1)	21(1)	96	76	0.0017	0.0003
2	923(23)	231(23)	152(23)	84	34	0.0029	0.0003
3	1445(72)	551(72)	464(72)	68	16	0.0037	0.0003
4	2353(208)	1411(208)	1319(208)	44	7	0.0039	0.0004
5	4547(532)	3634(534)	3539(534)	22	3	0.0038	0.0004
6	11702(1524)	10436(1539)	10345(1539)	11	1	0.0052	0.0003
	Lena						
1	453(1)	83(1)	15(1)	97	82	0.0017	0.0003
2	924(24)	230(24)	143(25)	85	38	0.0030	0.0003
3	1451(68)	538(68)	442(67)	70	18	0.0038	0.0004
4	2421(218)	1440(218)	1336(216)	45	8	0.0041	0.0004
5	4689(535)	3764(536)	3665(536)	22	3	0.0039	0.0004
6	10741(1392)	9863(1403)	9762(1403)	9	1	0.0037	0.0004

Table 4.4: Performance comparison (PSNR in dB) between various algorithms on Lena and Barbara Images. No back end arithmetic coding employed.

Coder	Bit rates (bpp)						
	0.0156	0.0313	0.0625	0.125	0.25	0.5	1.0
	Lena (512×512)						
JPEG 2000	22.58	25.06	27.47	30.25	33.27	36.30	39.28
SPECK	22.71	24.98	27.32	30.37	33.39	36.51	39.66
SPIHT	22.99	25.19	27.61	30.54	33.62	36.80	39.99
NLS	23.22	25.40	27.70	30.41	33.42	36.49	39.62
LSK	23.64	25.62	27.85	30.54	33.53	36.62	39.75
LEBP	23.71	25.67	27.90	30.56	33.54	36.62	39.75
	Barbara(512×512)						
JPEG 2000	20.81	22.34	23.39	25.06	27.82	31.42	36.49
SPECK	21.10	22.25	23.43	24.62	27.10	30.82	35.61
SPIHT	21.24	22.49	23.53	24.82	27.54	31.58	36.79
NLS	21.44	22.50	23.48	24.48	27.06	31.13	36.31
LSK	21.65	22.61	23.62	25.03	27.42	31.48	36.72
LEBP	21.68	22.63	23.64	25.05	27.43	31.48	36.72

in MATLAB 7.10.0 under Window XP, Intel Core 2 Duo CPU with 3 GHz speed and 4 GB of RAM space.

4.4.1 Coding Performance

The coding performance for gray scale and color images are discussed in terms of percentage of bits saved and the PSNR improvements.

Table 4.5: PSNR(dB) comparison of JPEG2000 standard test images (Grayscale)

Coder	Bit rates (bpp)						
	0.0313	0.0625	0.125	0.25	0.5	1.0	2.0
			Bike	(2560 × 2048)			
JPEG 2000(AAC)	-	23.74	26.31	29.56	33.43	37.99	43.95
SPIHT(No AAC)	-	22.86	25.29	28.51	32.36	36.98	42.90
SPECK(No AAC)	-	22.53	24.79	27.96	31.74	36.24	42.28
WBTC(No AAC)	-	23.14	25.48	28.76	32.51	37.12	43.13
NLS(No AAC)	-	23.40	25.86	28.75	32.15	36.46	42.53
LSK(No AAC)	21.73	23.66	26.03	28.94	32.33	36.54	41.94
LEBP(No AAC)	21.75	23.67	26.08	28.96	32.34	36.54	41.95
			Woman	(2560 × 2048)			
JPEG 2000(AAC)	-	25.59	27.33	29.95	33.57	38.28	43.97
SPIHT(No AAC)	-	25.07	26.91	29.43	32.43	37.73	43.21
SPECK(No AAC)	-	24.84	26.91	29.10	32.40	37.02	43.19
WBTC(No AAC)	-	25.29	27.08	29.71	33.23	37.91	43.62
NLS(No AAC)	-	26.61	28.34	30.56	33.30	36.99	42.60
LSK(No AAC)	25.76	26.95	28.67	30.84	33.51	37.05	41.99
LEBP(No AAC)	25.80	26.96	28.68	30.89	33.51	37.05	42.00

4.4.1.1 Coding Performance for Gray Scale Images

In order to evaluate the coding performance, we first compare the cumulative number of bits generated in different pass of SPIHT, LSK and LEBP algorithms. Table 4.3 shows the cumulative number of bits generated in the first six passes of three algorithms on two different type of standard images (a) Barbara and (b) Lena.

Let us denote the symbols illustrated in Table 4.3 as follows:

- $A_0(S_0)$: A_0 is the cumulative number of bits generated on top n passes of SPIHT and S_0 is the number of significant coefficients corresponding to these n passes, where $n = 1, 2, 3, ...6$.
- $A_1(S_1)$: A_1 is the cumulative number of bits generated on top n passes of LSK and S_1 is the number of significant coefficients corresponding to these n passes.
- $A_2(S_2)$, A_2 is the cumulative number of bits generated on top n passes of LEBP and S_2 is the number of significant coefficients corresponding to these n passes.
- αA_{20} is the percentage of bit saving in LEBP with respect to SPIHT.
- βA_{21} is percentage of bit saving in LEBP with respect to LSK.
- BRI_{20} is the bit rate improvement of LEBP with respect to SPIHT.
- BRI_{21} is the bit rate improvement of LEBP with respect to LSK.

The bit rate at n^{th} pass is given by:

$$\text{Bit rate} = \frac{\text{Cumulative number of bits generated}}{\text{Original image size}}. \tag{4.5}$$

The BRI (bpp) can be found by taking the bit rate difference between any two algorithms. For example, at n^{th} pass

$$\begin{aligned} \text{BRI}_{20} &= \text{Bit rate(SPIHT)} - \text{Bit rate(LEBP)}, \\ \text{BRI}_{21} &= \text{Bit rate(LSK)} - \text{Bit rate(LEBP)}. \end{aligned} \tag{4.6}$$

It is evident form Table 4.3 that the proposed LEBP algorithm consistently outperforms the other two. The percentage of bits saved (αA_{20})by LEBP over SPIHT in the first six passes is about 96-97 %, 84-85 %, 68-70 %, 44-45 %, 22 % and 9-11 % respectively. Further comparing with LSK, LEBP saves (αA_{21}) 76-82 %, 34-38 %, 16-18 %, 7-8 %, 3.0 % and 1.0 % of bits in the first six bit plane passes respectively. Therefore, LEBP is more efficient in sorting significant coefficients than SPIHT and LSK. One of the main reason is that clusters of zeros are more likely to occur at the early passes when the threshold is high. The proposed algorithm encodes these clusters more efficiently than SPIHT and LSK.

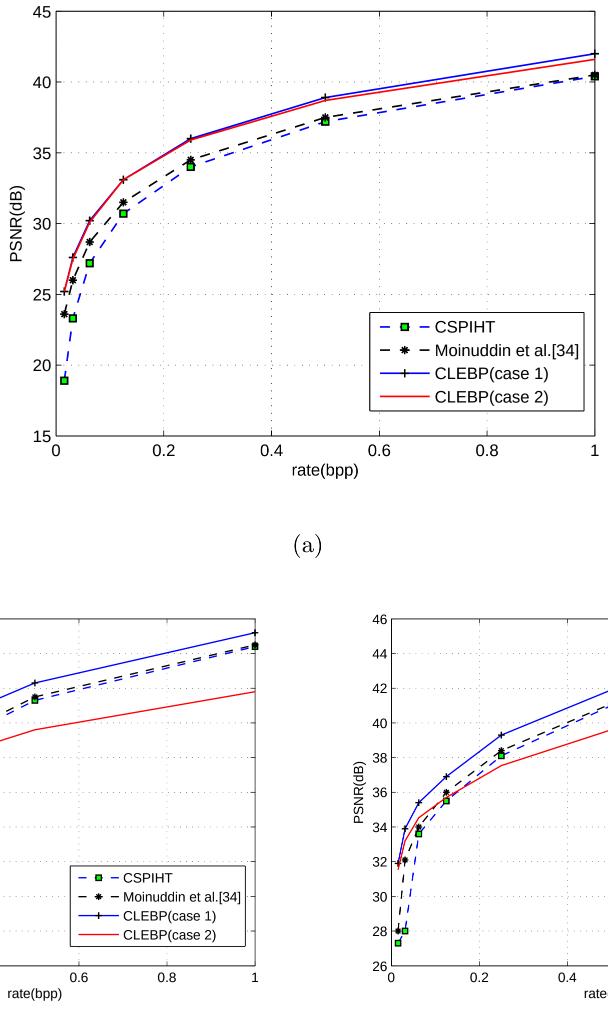
In Table 4.4, the PSNR results of JPEG 2000, SPECK, SPIHT and LSK algorithms on Lena and Barbara images are compared with LEBP algorithm. For fair comparison, the results are shown without arithmetic coding. It is observed that LEBP algorithm exhibits superior performance for the range of bit rates considered. LEBP brings a gain of approximately 0.58-1.0 dB, 0.29-0.72 dB and 0.05-0.07 dB over SPECK, SPIHT and LSK respectively on Lena image particularly for very low bit rates ($\leq$ 0.0625 bpp). Similarly, LEBP brings a gain of approximately 0.21-0.58 dB, 0.11-0.44 dB and 0.02-0.03 dB over SPECK, SPIHT and LSK respectively on Barbara image. At medium bit rates, the improvement in PSNR gain is slightly higher. At higher bit rates, the gain of the proposed algorithm reduces slightly than SPIHT. Considering the case with JPEG 2000, LEBP outperform 0.13-0.61 dB over the range of bit rates considered on Lena image. LEBP exhibit a coding gain of 0.25-0.87 dB below 0.0625 bpp in case of Barbara image. However, JPEG2000 shows PSNR improvement of 0.01-0.39 dB between 0.125-0.25 bpp. It is also observed that LEBP algorithm outperforms JPEG 2000 at lower bit rates ($\leq$ 0.0625 bpp) in most of the images.

In Table 4.5, ‘AAC’ in the parenthesis indicates that the results are obtained using arithmetic coding. ‘No AAC’ indicates that the results are obtained without using arithmetic coding. The meaning is followed consistently throughout the thesis. It is observed that LEBP outperforms SPIHT by 0.45-0.81 dB on bit rates $\leq$ 0.25 bpp, SPECK by 0.6-1.14 dB on bit rates $\leq$ 0.5 bpp, WBTC [33] by 0.2-0.53 dB on bit rates $\leq$ 0.25 bpp and LSK by 0.01-0.05 dB on bit rates $\leq$ 0.5 bpp. For medium (i.e., $\geq$ 0.5 bpp) to higher bit rates (i.e., $\leq$ 2 bpp) LEBP under performs SPIHT by 0.02-0.95 dB, SPECK by 0.33 dB, WBTC by 0.17-1.18 dB on Bike image.

On Woman image, LEBP outperforms SPIHT by 0.89-1.87 dB on bit rates $\leq$ 0.5 bpp, SPECK by 0.03-2.12 dB on bit rates $\leq$ 1.0 bpp, WBTC by 0.38-1.18 dB on bit rates $\leq$ 0.5 bpp and LSK by 0.01-0.05 dB on bit rates 0.0156-2 bpp. Similarly, for medium to higher bit rates, LEBP under performs SPIHT by 0.0.68-1.21 dB, SPECK by 1.19 dB and WBTC by 0.86-1.21 dB.

4.4.1.2 Coding Performance for Color Images

In order to understand the effectiveness of the proposed algorithm, simulation is carried out on standard still color images such as Lena of 512 × 512 size, first frame 'Hall-Monitor' of quarter common intermediate format (QCIF: 176 × 144 size) and 'News' of common intermediate format (CIF: 352×288 size) of the standard MPEG-4 test video sequences. The comparison results are shown in Figure 4.2, 4.3 and 4.4 respectively. The experimental results of the proposed algorithm has been carried out in YCbCr color space because of its higher compression performance than YUV color space. The results of CSPIHT and algorithm by Moinuddin *et al.* [34] are directly quoted from [34]. These results are reported in YUV color space. CLEBP(case 1) uses lanczos2 sub-sampling or up-sampling techniques in chrominance planes, whereas CLEBP(case 2) subsamples chrominance planes by discarding alternate rows and columns. It is observed that the PSNR values of CLEBP(case 1) outperform the PSNR results reported in [34] by 1.4-1.6 dB on Y plane and 0.6-3.9 dB on chrominance planes. Considering CLEBP(case 2), Lena image shows higher performance in Y-plane. However, in chrominance plane the performance goes down by 2 dB. Hall-Monitor and News images are compressed using a slight different procedure than Lena image since their dimensions are not equal. If the image is padded using border extension (replicate), then the algorithm is considered as CLEBP(case 3). The border is extended symmetrically in case of CLEBP(case 4). Using CLEBP(case 3) a gain of 0.1-0.2 dB in Y-plane and 0.1-3.6 dB in chrominance planes is observed on Hall-Monitor image. A slight degradation in PSNR values are observed in Y-plane between 0.0625-0.25 bit rates. News image shows a gain of 0-0.8 dB in Y-plane and gain up to 2.5 dB in chrominance planes over algorithm in [34]. The decoded Hall-Monitor image shows PSNR degradation up to 3 dB in Y-plane while compressed using CLEBP(case 4). The decoded images of Hall-Monitor are shown in Figure 4.5(b) and (c). It is observed that the decoded images using CLEBP(case 3) have better perceptual qualities than CLEBP(case 4). Similar kind of change in performance is observed in News image, shown in Figure 4.6(b) and (c).



(a)

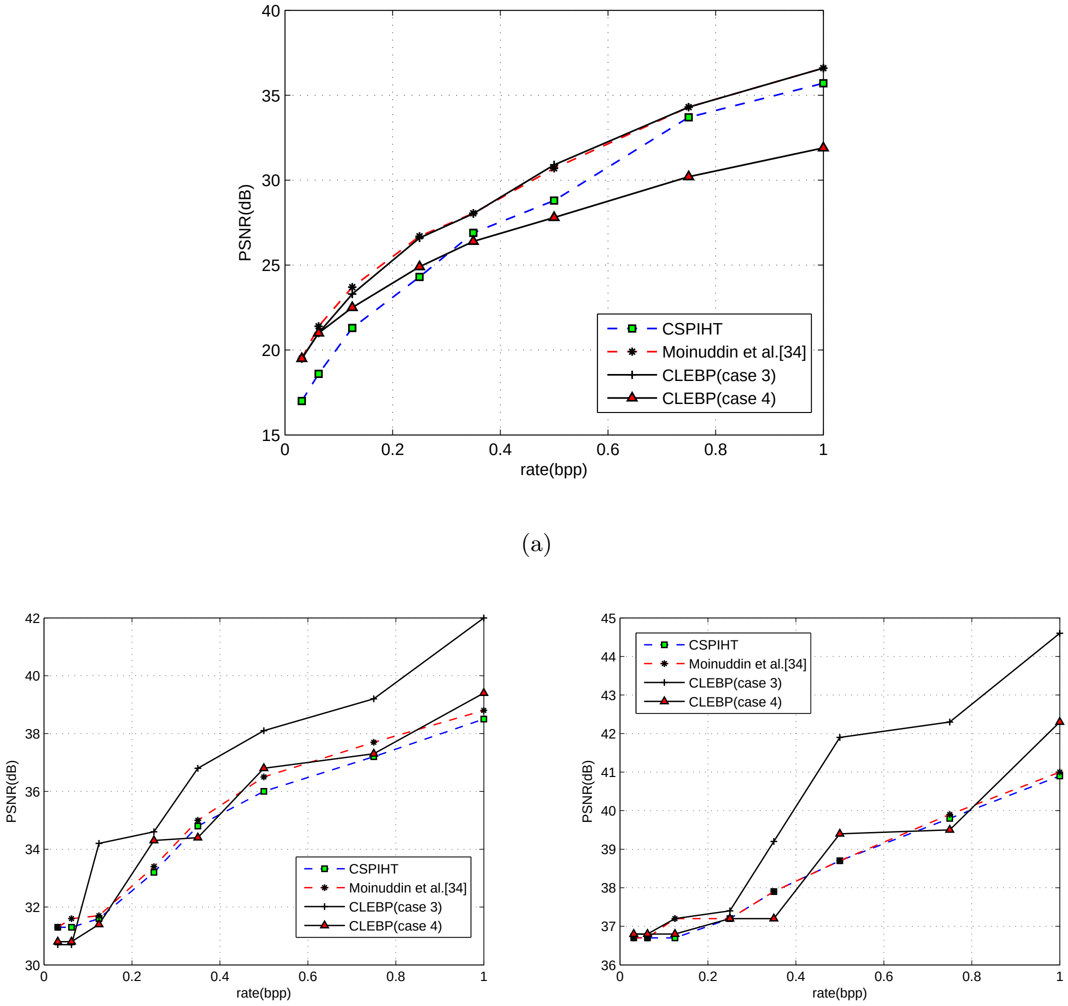
(b)

(c)

Figure 4.3: Rate-distortion performance comparison of Lena image on (a)Y-plane (b)U-plane for CSPIHT and algorithm by Moinuddin *et al.* [34], Cb-plane for CLEBP(case 1 and case 2) (c) V-plane for CSPIHT and algorithm by Moinuddin *et al.*, Cr-plane for CLEBP.

4.4.1.3 Coding Performances Between Gray scale and Color Images

While comparing the results between SPIHT and LEBP on gray scale images, LEBP shows poor results than SPIHT at higher rates (e.g., 0.25 to 1.0 bpp). However, in case of color images CLEBP(case 1) shows consistently better performance even in luminance plane at higher rates (Figure 4.2(a)). The possible reasons for inconsistency



(a)

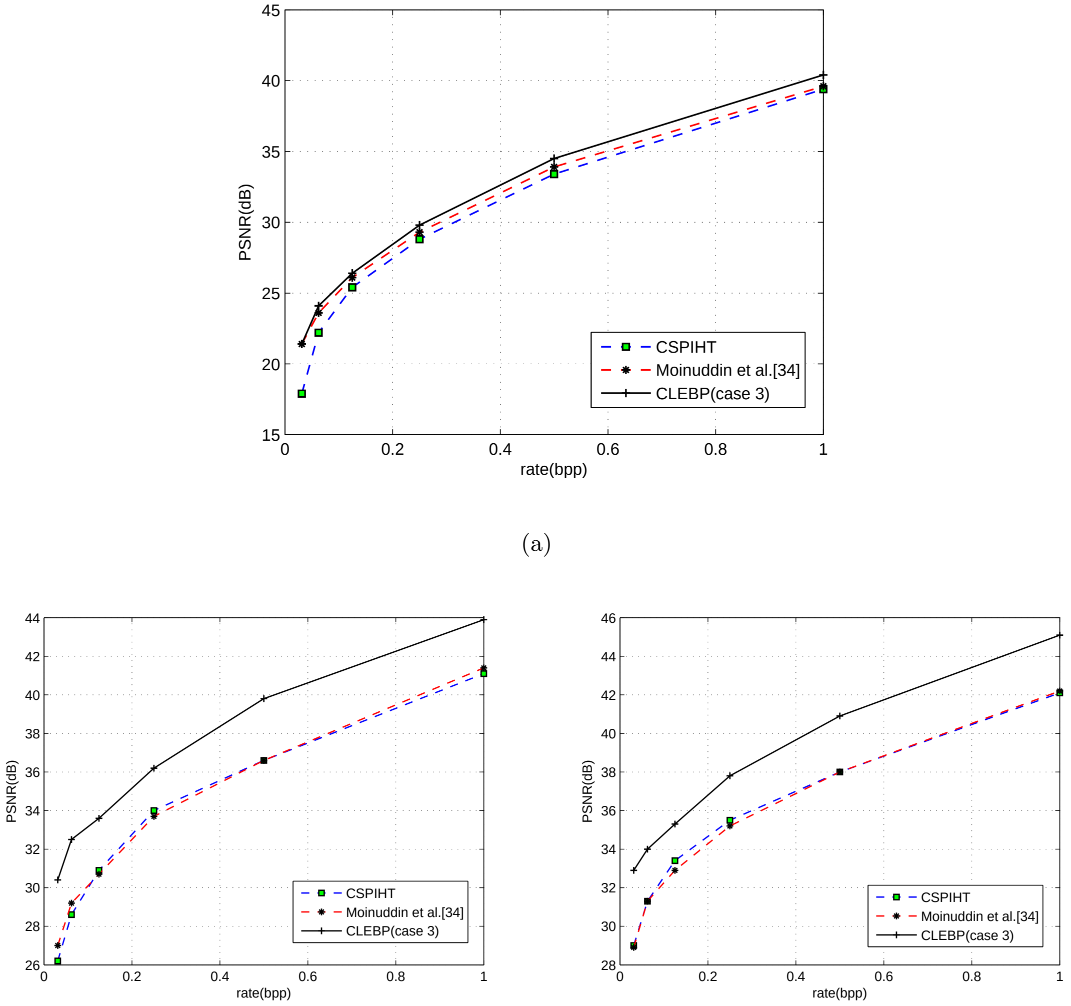
(b)

(c)

Figure 4.4: Rate-distortion performance comparison of Hall-Monitor image on (a)Y-plane (b)U-plane for CSPIHT and algorithm by Moinuddin *et al.*, Cb-plane for CLEBP (c)V-plane for CSPIHT and algorithm by Moinuddin *et al.*,Cr-plane for CLEBP.

between the two results can be as follows:

CLEBP(case 1) uses lanczos2 sub-sampling and up-sampling of the chrominance planes at the encoder and the decoder sides respectively. CSPIHT and algorithm in [34] uses the type of sub-sampling like the one in CLEBP(case 2). By simply discarding alternate rows and columns in chrominance planes, CLEBP(case 2) performs worst than CSPIHT and algorithm in [34], but better in luminance plane. From the simulation results, it is observed that the overall performance of CLEBP (case 2) is lower than the above two algorithms at higher rates while the performance of



(a)

(b)

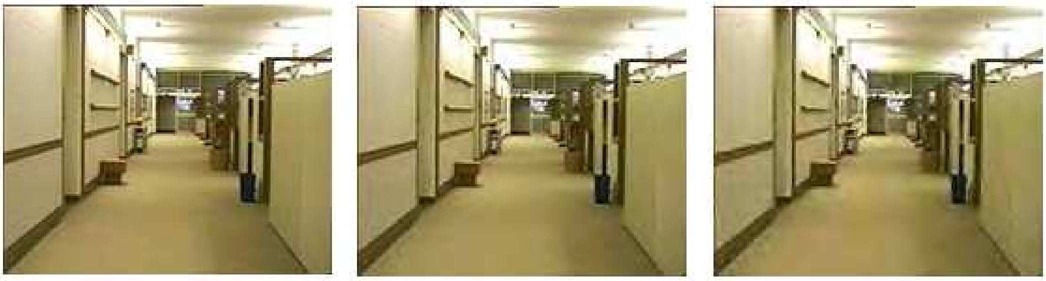
(c)

Figure 4.5: Rate-distortion performance comparison of News image on (a)Y-plane (b)U-plane for CSPIHT and algorithm by Moinuddin *et al.*, Cb-plane for CLEBP (c)V-plane for CSPIHT and algorithm by Moinuddin *et al.*, Cr-plane for CLEBP.

CLEBP(case 1) is higher on all the bit rates. Since the human eye is more sensitive to luminance components than the chrominance components, the decoded images using CLEBP(case 2) have better perceptual qualities than CSPIHT.

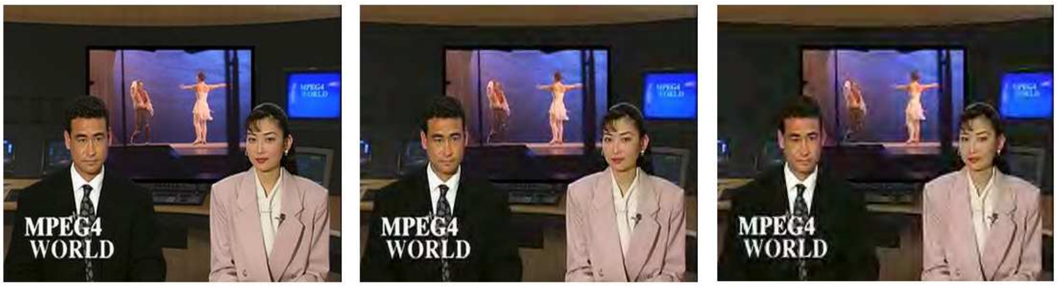
The procedure for encoding and decoding of CIF and QCIF images by CLEBP(case 3) algorithm are as follows:

- The image is padded by extending the border pixels(i.e., replicate padding), since the algorithm is highly dependent on images having equal dimensions and having dimensions integer powers of 2 (e.g. QCIF and CIF dimensions are changed to



(a) (b) (c)

Figure 4.6: (a) Original Hall monitor image compressed by CLEBP at 1 bpp using (b) replicate padding (case 3) (c) symmetric padding (case 4).



(a) (b) (c)

Figure 4.7: (a) Original News image compressed by CLEBP at 0.5 bpp using (b) replicate padding (case 3) (c) symmetric padding (case 4).

256 × 256 and 512 × 512 respectively). However, CSPIHT algorithm accesses the coefficients from the lists. Therefore, the image dimensions are need not to be equal or an integer power of 2.

- It is verified that the performance of CLEBP(case 3) is degraded by padding symmetrically/ circularly/ zero across border of QCIF or CIF images compared to replicate type of padding.
- CLEBP encodes luminance and chrominance planes by an efficient one dimensional clustering arrangement of all the subband coefficients.
- The PSNR is calculated by removing the padded portion from the decoded image at the decoder side.

The procedure for encoding/decoding color images in CLEBP (case 4) is similar to CLEBP(case 3).

Table 4.6: PSNR comparison for the SPIHT and LEBP coder at different spatial resolutions (Lena 512 × 512, 0.05 bpp)

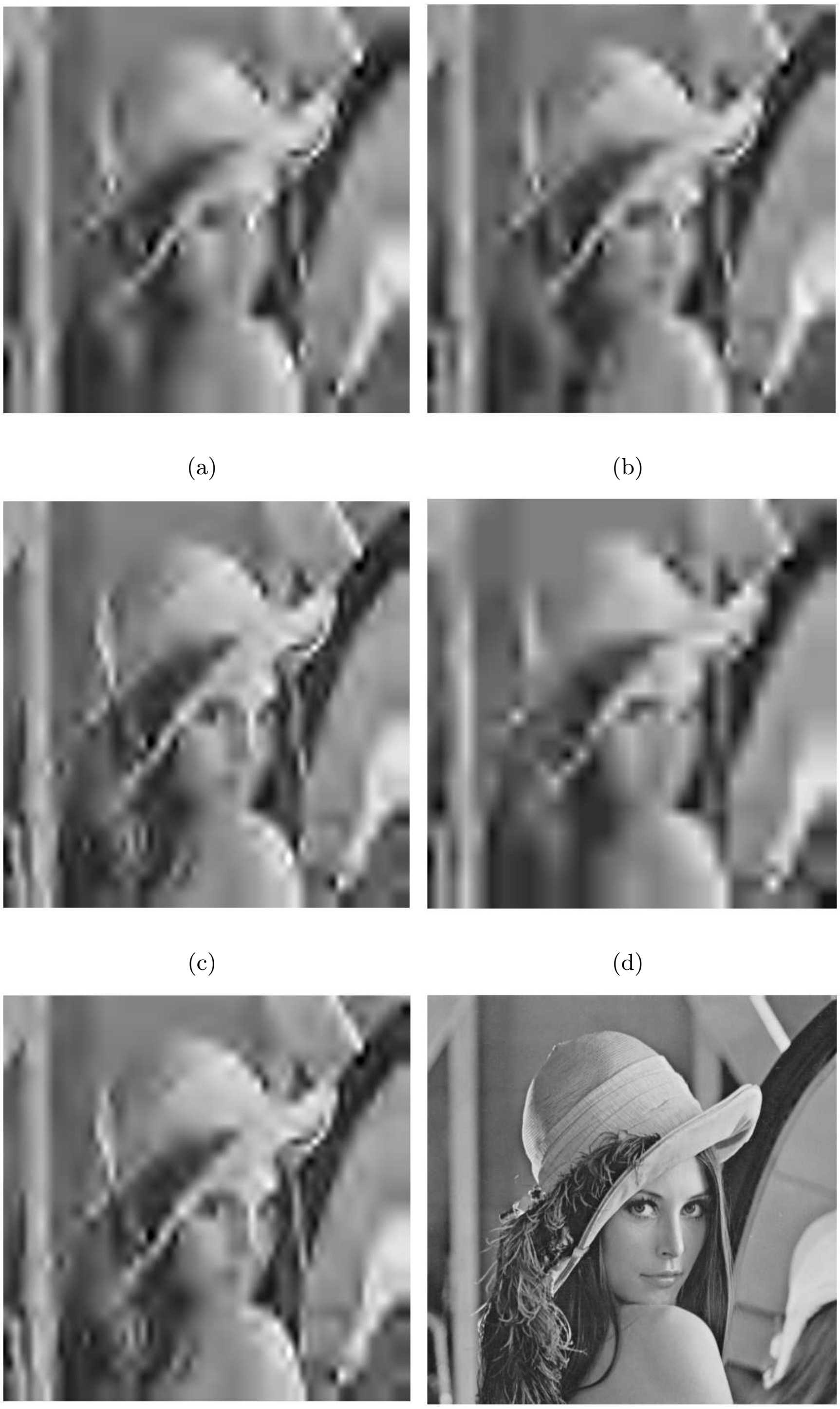
Spatial resolution	PSNR(dB)			
	SPIHT	NLS	LSK	LEBP
32 × 32	37.84	36.05	38.63	38.63
64 × 64	29.65	30.32	31.53	31.55
128 × 128	26.07	26.25	26.38	26.41
256 × 256	24.60	24.86	24.91	25.05

The results of CSPIHT and algorithm by Moinuddin *et al.* are directly quoted form [34]. CSPIHT does not need any kind of border extension to encode images because each coefficient in the coarsest subband constitutes a quad tree. The block-tree algorithm presented by Moinuddin *et al.* needs some kind of adjustment to form root blocks of equal dimensions. As a result, additional circuits are needed for image size adiustment in CIF/QCIF images. It is observed from the simulations that the type of sub-sampling/border extensions (e.g., symmetrical, replicate, circular or zero padding) has greater impact on coding performance in chrominance planes than luminance plane. The effect can be observed in Figures 4.3, 4.4 and 4.5. CSPIHT is tree based and it uses inter and intra subband correlations. Algorithm proposed in [34] also uses inter as well as intra subband correlations more efficiently than CSPIHT due to block-trees. CSPIHT and algorithm in [34] use list structure whereas, the proposed CLEBP uses markers. CLEBP exploits inter and intra subband correlations by efficiently clustering similar blocks and coding of the zero clusters in a one dimensional manner. It is expected that the coding performances of CLEBP and algorithm in [34] have similar rate-distortion performances at lower rates.

4.4.1.4 Progressive Transmission and Scalability Efficiency

The progressive transmission performance of LEBP relative to SPIHT and LSK is indicated by percentage of encoded bit saved at lower bit plane passes. As discussed in above sections, the percentage of bit saved at top 1 to 6 passes is significant. The difference decreases with increase in sorting passes (higher rates) as all coders include many wavelet coefficients to represent image. Therefore, a large amount of saving in encoded bit length in LEBP contributes to a substantial increase of progressive performance as the image is recognized at a much lower rates. For example, in figures 4.8 and 4.9, Lena and Cameraman images are compressed at 0.01 and 0.0313 bpp respectively using SPIHT, NLS, LSK, JPEG2000 and LEBP coders clearly demonstrate the enhanced recognizability of LEBP decoded images.

The spatial scalability efficiency of LEBP relative to the SPIHT, NLS and LSK

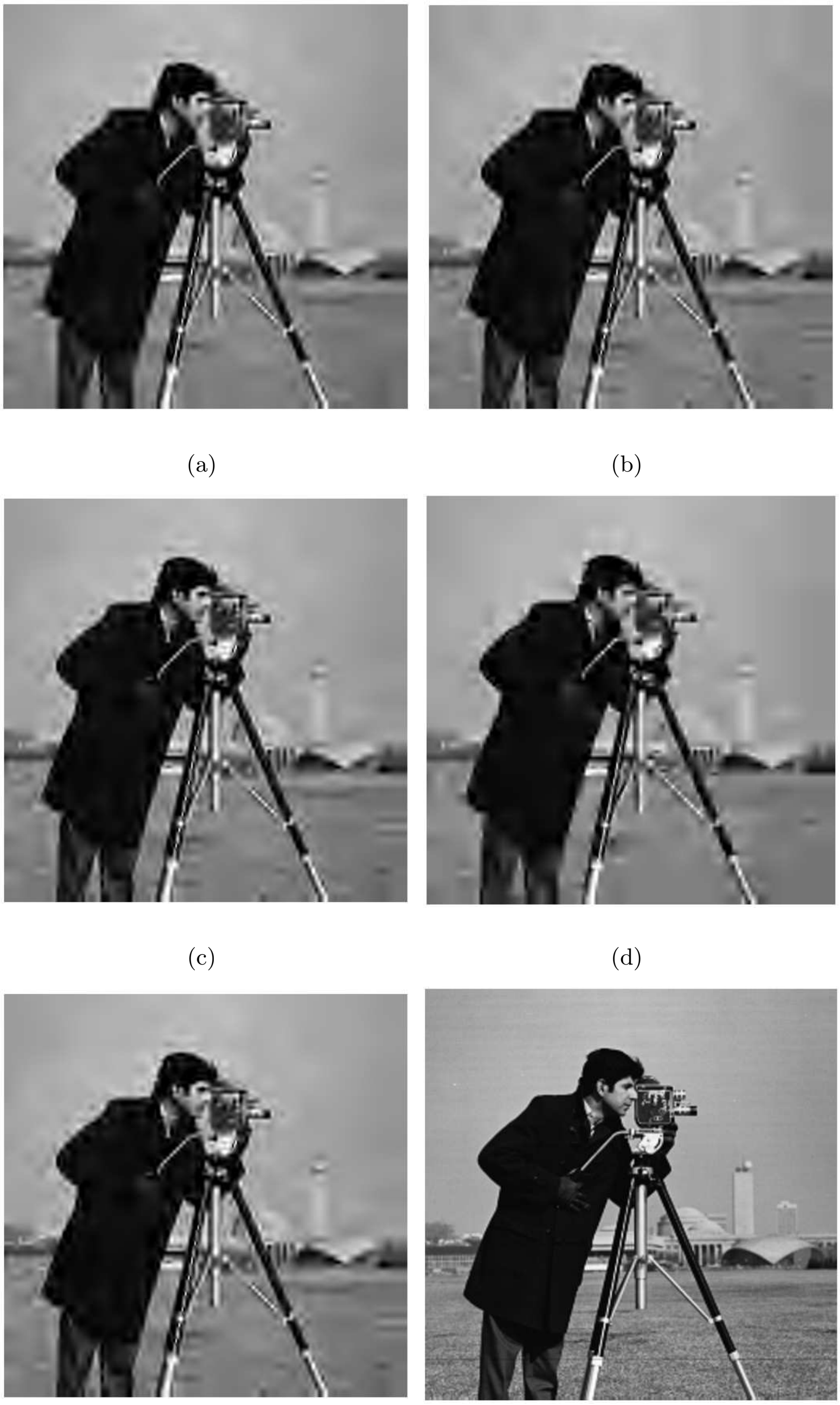


(a) (b)

(c) (d)

(e) (f)

Figure 4.8: Decoded Lena images compressed at bit rates of 0.01 using (a) SPIHT (PSNR=21.44 dB, MSSIM=0.5800), (b) NLS (PSNR=21.95 dB, MSSIM=0.6029), (c) LSK(PSNR=22.58 dB, MSSIM=0.6333), (d) JPEG2000(PSNR=21.01, MSSIM=0.5664),(e) LEBP(PSNR=22.71 dB, MSSIM=0.6386), and (f)Original image

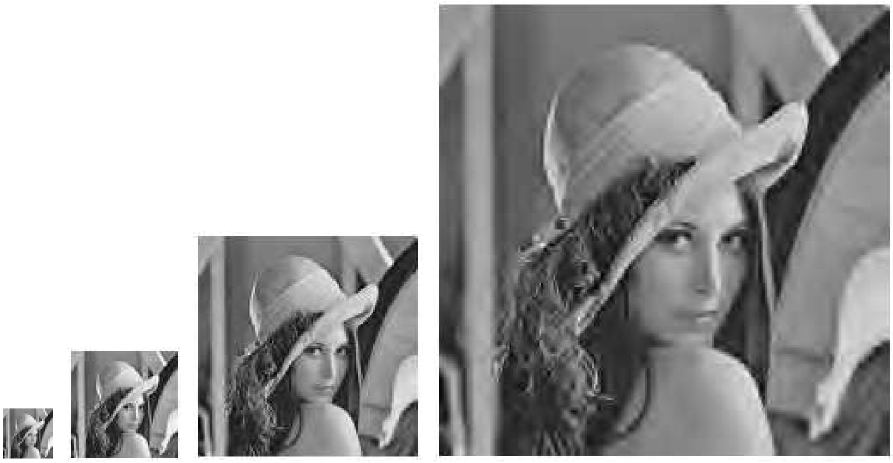


(a) (b)

(c) (d)

(e) (f)

Figure 4.9: Decoded Cameraman images compressed at bit rates of 0.0313 using (a) SPIHT (PSNR=24.88 dB, MSSIM=0.7447), (b) NLS (PSNR=25.09 dB, MSSIM=0.7561), (c) LSK(PSNR=25.30 dB, MSSIM=0.7626), (d) JPEG2000(PSNR=24.85, MSSIM=0.7543), (e) LEBP(PSNR=25.33 dB, MSSIM=0.7640), and (f) Original image



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

Figure 4.10: Resolution scalable Lena images decoded at a bit rate of 0.05 bpp by LEBP algorithm (a) 32×32(PSNR=38.63 dB,MSSIM=0.9971), (b) 64×64(PSNR=31.55 dB,MSSIM=0.9806), (c) 128×128(PSNR=26.41 dB,MSSIM=0.8905), (d) 256×256(PSNR=25.05 dB,MSSIM=0.7886)

is indicated by the quality of the decoded images at different spatial resolutions. Therefore, it can be evaluated by comparing the quality of the lower resolution versions of the image. Table 4.6 shows the PSNR comparison for the Lena image coded at 0.05 bpp with LEBP, NLS, LSK and SPIHT coders. The PSNR values are obtained by scaling the original image appropriately from the subband decompositions. It is observed that the PSNR values of LEBP are higher than SPIHT with decreasing spatial resolutions. The quality of decoded images between LSK and LEBP are nearly same at lower resolutions. Figure 4.10 shows the resolution scalable decoded Lena images.

4.4.1.5 Region of interest and Random access decoding

The user may desire certain parts of an image that are of greater importance to be encoded with higher fidelity compared to rest of the image. During decompression, the quality of the image can be adjusted depending on the degree of interest. In order to obtain region of interest (ROI), the wavelet coefficients of the transformed image are arranged in such way that a block of wavelet coefficients corresponds to a local region of an image has been shown in Figure 4.11. The encoding bit stream format is exactly similar to Figure 3.4 which is shown in chapter 3. The detail explanation of how to achieve ROI was also discussed in chapter 3. Figure 4.12(a)-(d) illustrate an example of LEBP's ROI retrievability scalability, where the facial portion of Lena image is retrieved from the coded bit stream at 4 different bit rates, i.e., 0.125 bpp, 0.25 bpp, 0.50 bpp and 1.0 bpp respectively.

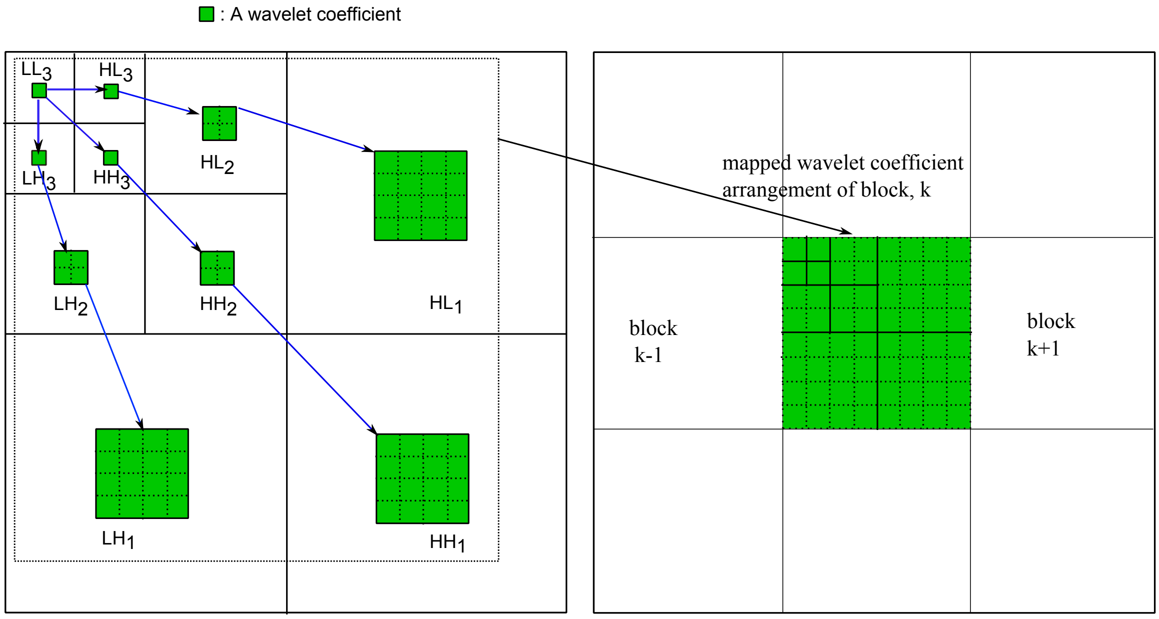
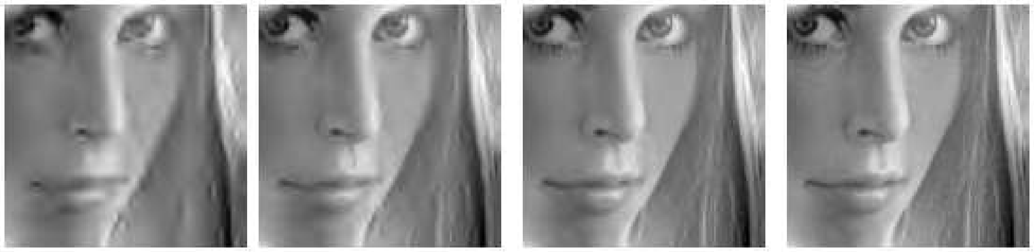

🟩 : A wavelet coefficient
LL3
HL3
LH3
HH3
HL2
LH2
HH2
HL1
LH1
HH1
mapped wavelet coefficient arrangement of block, k
block k-1
block k+1

Figure 4.11: Rearrangement of wavelet transformed coefficients for ROI/random access decoding where correlated coefficients across different scales are mapped into a localized region of the source image.

Random access decoding scheme is for extracting the target image information for the compressed bit string with minimal decoding work. It is useful in interactive image browsing systems, where the user first browses coarse resolution images and then probably looks into details of these parts according to its interest. The random access decoding, naturally follows the same kind of bit stream format as that for ROI retrievability.



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

Figure 4.12: ROI of facial portion of Lena image (128×128 size) which is retrieved at a bit rate of (a)0.125 bpp, (b)0.25bpp, (c)0.5bpp, and (d)1.0bpp.

The beginning of the target bit stream block can be estimated by using the length information of every block which has been coded before the target block. This leads to minimal decoding time. The length information stored in each block causes an inevitable overhead and loss of coding efficiency. The average seek time is $Lp\frac{N_b}{2}$, where L is the number of resolution scales, p is the average skip time and N_b is the total number of image blocks. From [103], it is clear that the linear method has $O(n)$ seek time, while bi-section method has $O(log_2n)$ seek time. Therefore, the average seek time can be improved to an order of $O(\frac{n}{log_2n})$ faster using bi-section method [101].

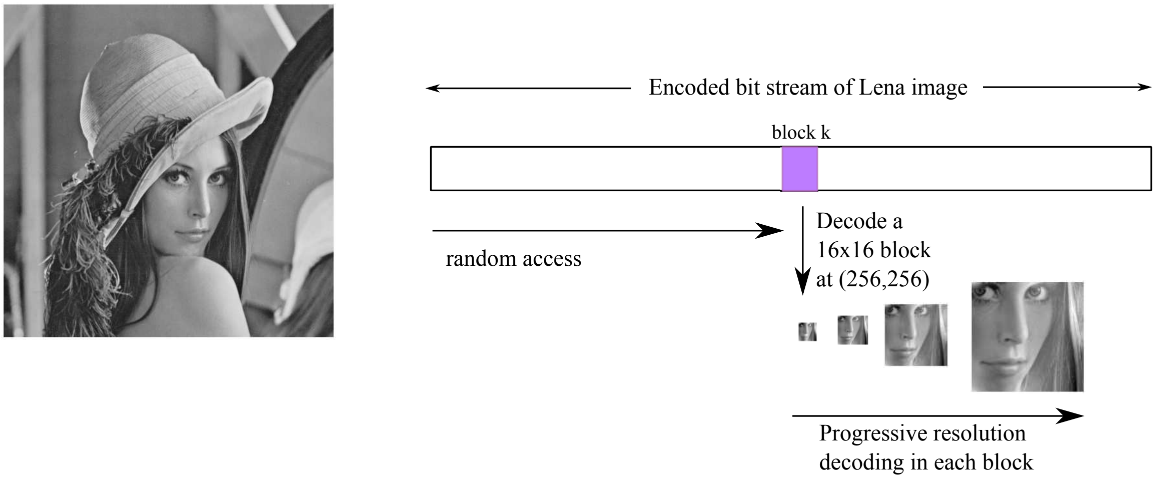


Figure 4.13: Random access decoding of k^{th} block at 1.0 bpp

In the example is shown in Figure 4.13, once the beginning of the target block is seeked, each bit after the point of bit stream is decoded until the requested resolution is achieved. A size of 16 × 16 block located at indices (256, 256) of a 512 × 512 Lena image is randomly accessed from the compressed bit stream, then three higher resolutions are progressively decoded.

4.4.2 Effect of Block Size

In order to better understand the impact of block sizes on the performance of the proposed techniques, the coding efficiency and encoding/decoding time are measured with blocks of 8 × 8, 16 × 16 and 32 × 32 sizes using the technique which is shown in Figure 4.14.

4.4.2.1 Impact of Coding Eficiency on Varying Block Sizes

The PSNR results of 8×8, 16×16 and 32×32 blocks on Lena and Barbara images are presented in Table 4.7, 4.8 and 4.9 respectively. It is observed that the path map bits of the significant blocks from medium to higher bit rates increases as the block size increases. For low bit rates, a reduction in coding performance is observed, because

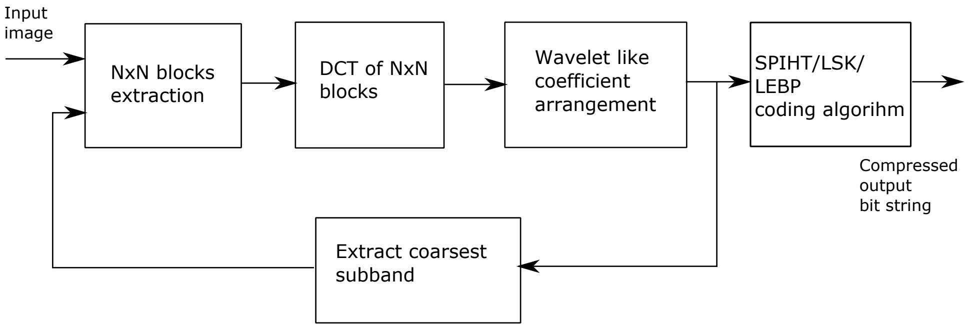

Input image
NxN blocks extraction
DCT of NxN blocks
Wavelet like coefficient arrangement
SPIHT/LSK/ LEBP coding algorihm
Compressed output bit string
Extract coarsest subband

Figure 4.14: Proposed block diagram of DCT based embedded coder using various block sizes

the number of significant map reduces in a larger block/set. Therefore, block size of 32 × 32 shows inferior performance over 8 × 8 and 16 × 16 block sizes at lower rates. This is clearly shown in Figure 4.15 on (a) Lena and (b) Barbara images for DCT_LEBP only. It can be observed that the overall rate-distortion performance of 16 × 16 block size is better over the bit rates considered.

Table 4.7: PSNR (dB) performance of various standard gray scale images without using back end arithmetic coding

Coder	Bit rates (bpp)						
	0.0156	0.0313	0.0625	0.125	0.25	0.5	1.0
	Lena						
EQDCT	18.12	20.91	24.70	27.98	31.56	35.40	39.27
DCT_SPIHT	22.01	23.77	25.99	28.56	31.88	35.68	39.36
DCT_LSK	22.51	24.05	26.25	28.79	31.99	35.71	39.22
DCT_LEBP	22.57	24.10	26.27	28.82	32.00	35.72	39.23
	Barbara						
EQDCT	17.75	19.98	22.25	24.32	27.05	30.85	36.25
DCT_SPIHT	20.56	21.68	22.63	24.40	26.90	30.66	36.09
DCT_LSK	20.88	21.90	23.04	24.69	27.30	31.05	36.09
DCT_LEBP	20.92	21.93	23.05	24.70	27.31	31.06	36.10
	Mandrill						
EQDCT	17.78	19.08	19.99	21.01	22.48	24.75	28.19
DCT_SPIHT	19.09	19.59	20.15	21.09	22.45	24.71	28.20
DCT_LSK	19.30	19.70	20.30	21.18	22.51	24.62	28.05
DCT_LEBP	19.31	19.71	20.31	21.19	22.52	24.63	28.06

Table 4.7 shows the PSNR performances of proposed techniques, viz. DCT_SPIHT, DCT_LSK, DCT_LEBP with Embedded quad-tree DCT (EQDCT) [43] on 8×8 block sizes. It is observed that DCT_LEBP outperforms EQDCT (0.32-4.45 dB over 0.0156 to 0.5 bpp) and DCT_SPIHT (0.04-0.56 dB over bit rate 0.0156-0.5 bpp). At higher rates ($\geq$ 1.0 bpp) slight reduction of PSNR value (approx. 0.04-0.13 dB) is observed in Lena image. On Barbara image a gain of 0.21-3.17 dB and 0.01-0.42 dB is observed

over the considered bit rates. Similar kinds of performance is observed on Mandrill image. The large PSNR gain on the above images at rates lower than 0.0313 bpp indicates the superior coding efficiency of the proposed DCT_LEBP algorithm at lower bit rates. Therefore, the proposed coder has better capability in sorting significant coefficients in earlier passes.

Table 4.8: PSNR(dB) performance on Lena image with varying block sizes

Coder	Bit rates (bpp)						
	0.0156	0.0313	0.0625	0.125	0.25	0.5	1.0
			(16×16)	blocks			
EQDCT(AAC)	21.82	24.18	26.66	29.56	32.90	36.40	39.75
Song [44](AAC)	-	-	27.57	30.28	33.36	36.64	39.93
DCT_SPIHT(No AAC)	21.52	23.88	26.29	29.06	32.29	35.79	39.35
DCT_LSK(No AAC)	22.50	24.41	26.77	29.60	32.87	36.23	39.58
DCT_LEBP(No AAC)	22.54	24.43	26.78	29.61	32.88	36.24	39.59
			(32×32)	blocks			
EQDCT(AAC)	18.96	21.89	25.39	29.08	32.72	36.28	39.59
DCT_SPIHT(No AAC)	18.25	20.96	24.94	28.65	32.50	36.13	39.52
DCT_LSK(No AAC)	18.80	21.89	25.36	28.99	32.68	36.06	39.29
DCT_LEBP(No AAC)	18.82	21.90	25.37	29.00	32.68	36.06	39.29

Table 4.8 shows the PSNR performance of Lena image using DCT_SPIHT, DCT_LSK and DCT_LEBP algorithms. Among the proposed coders, DCT_LEBP exhibits better rate distortion (PSNR) performance. Therefore, the coding performance of DCT_LEBP embedded coder is compared with coders proposed by Song and Cho [44] and EQDCT. It is to be noted that embedded coder proposed by Song and Cho consists of block sizes 32×32 together with five levels of wavelet like DCT coefficient arrangement, combined with SPIHT and context based arithmetic coding. Comparing with [44], coder DCT_LEBP (without arithmetic coded) with 16×16 block shows PSNR reduction of 0.4-0.79 dB over bit rates 0.0625-1.0 bpp. By using context based arithmetic coding, it is expected that DCT_LEBP coder can outperform the coder in [44]. Further, DCT_LEBP outperforms EQDCT by 0.05-0.72 dB in low bit rates. At medium bit rates, a slight reduction of PSNR value (0.02 to 0.16 dB) is observed.

In Table 4.9 results are shown for Barbara image, DCT_LEBP with 16×16 block shows PSNR reduction of 0.15-0.9 dB in comparison with [44] on 0.0625-1.0 bit rates. It is observed further that DCT_LEBP with 32×32 blocks has slightly more gain than 16×16 sizes at medium to higher bit rates. Therefore, the proposed algorithm DCT_LEBP with 32×32 block sizes has comparable performance with algorithm proposed by Song and Cho [44]. As far as algorithm complexity is concerned, DCT_LEBP is less complex as it do not use list structure and avoids complex kind of coefficient

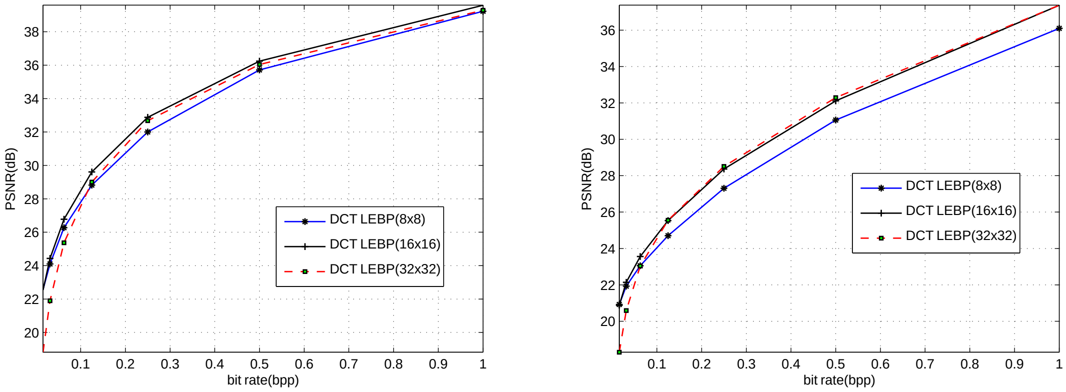
Table 4.9: PSNR(dB) performance on Barbara image with varying block sizes

Coder	Bit rates (bpp)						
	0.0156	0.0313	0.0625	0.125	0.25	0.5	1.0
			(16×16)	blocks			
EQDCT(AAC)	20.43	21.90	23.57	25.79	28.60	32.41	37.57
Song and Cho[44](AAC)	-	-	24.06	26.43	29.27	32.82	37.52
DCT_SPIHT(No AAC)	20.26	21.61	23.11	25.28	28.05	32.10	37.27
DCT_LSK(No AAC)	20.87	22.11	23.55	25.53	28.37	32.10	37.37
DCT_LEBP(No AAC)	20.88	22.14	23.56	25.53	28.37	32.11	37.37
			(32×32)	blocks			
EQDCT(AAC)	20.08	22.02	23.90	26.32	29.18	32.96	37.89
DCT_SPIHT(No AAC)	17.64	19.83	22.61	25.31	28.33	32.24	37.38
DCT_LSK(No AAC)	18.23	20.57	23.02	25.54	28.50	32.29	37.37
DCT_LEBP(No AAC)	18.30	20.59	23.04	25.55	28.51	32.29	37.37

sorting method like in [44].

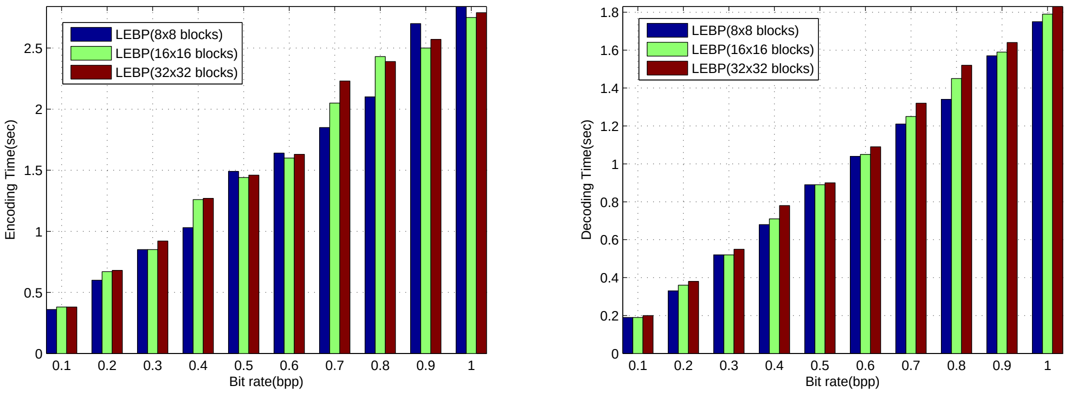
4.4.2.2 Impact of Encoding and Decoding Times on Varying Block Sizes

The results of encoding and decoding times using 8×8, 16×16 and 32×32 blocks of Barbara image are compared in Figure 4.16 for LEBP algorithm only. It can be observed that the encoding and decoding times slightly varies over changing block sizes. In fact, decoding times are slightly higher with increase of block sizes. This phenomenon is true on encoding times. However, encoding time is always higher than decoding time for a given bit rate. Similar trends are also observed in other standard images.



(a) (b)

Figure 4.15: Rate-distortion performance of DCT_LEBP algorithm on varying block sizes for (a)Lena and (b)Barbara images.



(a) (b)

Figure 4.16: Comparison of (a) Encoding time(sec) and (b)Decoding time(sec) on varying block sizes of LEBP algorithm on Barbara image.

4.4.3 Computational Complexity

The computational complexity is analyzed in terms run times and number of arithmetic operations. Table 4.10 summarizes the run times (seconds) for the four algorithms on standard gray scale images such as Lena and Barbara at various bit rates. It is observed that the encoding times of LEBP are about 1.5-2 times faster at lower rates, 3-15 times faster than SPIHT and SPECK at higher rates in Lena image. But it is slightly more (1.3 times) than LSK. The decoding times are approximately 2-4 times faster at lower rates and 4-20 times faster at higher rates than SPIHT and SPECK. Decoding times of LEBP are lesser than LSK. Similar trends are observed for other standard images. It is observed that the encoding time of the proposed algorithm is slightly more than LSK. This is because of extra overhead is required to test the significance of a group of levels (IL or IGL pass) on earlier bit plane passes. At the decoder end, this overhead gets compensated because of efficient skipping of a number of subbands from a larger set. This reduces the decoding times of LEBP compared to LSK. Considering the number of arithmetic operation, the mathematical proof of the complexity is deferred to Appendix A.

4.5 Conclusions

In this chapter, two low complexity embedded image coders i.e., LEBP and CLEBP are proposed. The image reconstruction performance on gray scale and color images

Table 4.10: Encoding and Decoding times of Lena and Barbara images with different bit rates

Coder	Bit rates (bpp)						
	0.0156	0.0313	0.0625	0.125	0.25	0.5	1.0
	Encoding time(sec)						
	Lena						
SPECK	0.16	0.25	0.51	1.12	2.64	8.15	29.13
SPIHT	0.24	0.39	0.71	1.43	3.26	9.07	36.56
LSK	0.09	0.11	0.18	0.32	0.60	1.17	2.27
LEBP	0.11	0.14	0.24	0.42	0.79	1.47	2.87
	Barbara						
SPECK	0.27	0.32	0.64	1.11	3.19	8.19	20.86
SPIHT	0.23	0.39	0.63	1.58	3.33	7.86	26.60
LSK	0.08	0.11	0.18	0.39	0.55	1.07	2.18
LEBP	0.12	0.14	0.24	0.35	0.66	1.32	2.73
	Decoding time (sec)						
	Lena						
SPECK	0.12	0.16	0.38	0.92	2.26	7.16	24.20
SPIHT	0.04	0.08	0.21	0.65	2.01	6.75	31.31
LSK	0.04	0.07	0.13	0.24	0.48	0.95	1.89
LEBP	0.03	0.05	0.11	0.23	0.45	0.90	1.78
	Barbara						
SPECK	0.12	0.16	0.37	0.63	2.17	5.97	16.33
SPIHT	0.04	0.08	0.19	0.74	2.14	6.05	22.76
LSK	0.04	0.07	0.13	0.26	0.52	0.99	1.98
LEBP	0.03	0.06	0.11	0.21	0.45	0.91	1.87

are compared with some state-of-the-art wavelet and DCT based embedded coders. For a fair comparison, back-end arithmetic coding is not employed in the proposed coders. LEBP exploits the inefficiency of tree based SPIHT/NLS and block based SPECK/LSK coders at low bit rates by efficiently encoding a large clusters of zero blocks with few symbols at early passes. The proposed coders are fully embedded and do not use list structures like SPIHT and SPECK. This reduces memory/hardware requirements.

LEBP has better progressive-in-resolution feature capability than others. This feature is particularly useful while transmitting images in narrow band channels. The LEBP is made scalable in terms of pixel accuracy and resolution using a minor rearrangement to the bit string. These are very useful features for transmission of images in heterogeneous network having different access bandwidth and computational need. Other important features, such as random access and region of interest naturally follow in the scalable LEBP. It is also demonstrated that CLEBP exhibit better rate distortion trade-off than CSPIHT in almost all the bit rates using replicate padding and lanczos2 interpolation techniques. Like scalable LEBP, scalability in CLEBP can be realized using similar kind of rearrangement to the output bit string.

Chapter 5

Listless Block-Tree Set Partitioning Algorithm for Very Low Bit Rate Embedded Image Compression

Preview

This chapter presents a listless implementation of wavelet based block tree coding (WBTC) algorithm of varying root block sizes. WBTC algorithm improves the image compression performance of set partitioning in hierarchical trees (SPIHT) at lower rates by efficiently encoding both inter and intra scale correlation using block trees. Though WBTC lowers the memory requirement by using block trees compared to SPIHT, it makes use of three ordered auxiliary lists. This feature makes WBTC undesirable for hardware implementation as it needs a lot of memory management when the list nodes grow exponentially on each pass. The proposed listless implementation of WBTC algorithm uses special markers instead of lists. This reduces memory requirement by 88% with respect to WBTC and 89% with respect to SPIHT. The proposed algorithm is combined with discrete cosine transform (DCT), discrete Tchebichef transform (DTT) and discrete wavelet transform (DWT) to show its superiority over DCT and DWT based embedded coders including JPEG 2000 at lower rates. The compression performance on most of the standard test images is nearly same as WBTC and outperforms SPIHT by a wide margin particularly at lower bit rates.

5.1 Introduction

Recently, low bit rate image compression has assumed a major role in applications such as processing and storage on handheld mobile or portable devices, wireless transmission, streaming data on the Internet and transmission in narrow band channels.

Contemporary wavelet based coding provides substantial improvement in progressive picture build up qualities at lower rates. In addition, it supports a wide range of functionalities either with increase computational complexities, e.g., JPEG 2000 or increase memory requirements, e.g., EZW, SPIHT, S+P [17], SLCCA and [104].

SPIHT uses list structure to keep track of set partitions. However, the use of lists in these coders causes a variable data dependent memory management as the list nodes are updated on every passes. At high rates, it is possible that the number of list nodes can be more than the number of coefficients. This is an undesirable feature for hardware implementations. A variant of SPIHT called No list SPIHT (NLS) [35] which uses a state table with four bits per coefficients to keep track of set partitions is reported. Lanuzza *et al.* [105] proposed two modifications to NLS. In first case, refinement pass is executed at the first step instead of the last step in NLS and in second case, less number of markers are used. Then, the modified algorithm is implemented in a FPGA. Lian *et al.* [106] applied lifting wavelet transform to the image and replaced the lists used in SPIHT by flag maps. This lowers the memory requirements and improves the coding process. Recently, Li *et al.* [107] applied listless SPIHT for satellite image compression.

Though NLS is a low complexity image coding algorithm with performances nearly close to SPIHT, NLS does not fully exploit the coding performances at lower bit rates. A block-tree algorithm called wavelet block truncation coding presented in [33],[34] exploits both inter and intra subband correlation to improve the coding performance at very low bit rates with a slight increase of the decoder complexity. Pan *et al.* [78] presented a listless modified SPIHT algorithm which improves the low bit rate performance of SPIHT. However, it's performance drastically reduces from medium to higher bit rates. Our proposed algorithm named as Listless block-tree coding (LBTC) which not only reduces the dynamic memory requirement almost by 88-89% but also enhances the low to higher bit rate performance with an average reduction of encoder and decoder complexity.

The proposed algorithm performs explicit breadth and depth searches. State information is kept in a fixed size array which corresponds to the array of coefficient values with four bits per coefficient. Special markers are placed on lower levels of insignificant block trees when they are created. These markers are updated during tree partitioning. Efficient skipping of insignificant block tree is accomplished using a morton scan sequencing. In stead of addressing each coefficient using two indices, linear indexing scheme is used because this particular format has several computational and organizational advantages.

Though most of the research activities are focused on wavelet coders, DCT is still widely used in many practical applications such as JPEG [4],[108], MPEG-4 and H.264 [37] because of its compression performance and computational advantages. In the literature [39]-[44], DCT-based coders with innovative data organization strategies and representations of DCT coefficients have been reported. These coders exhibit high compression efficiency. Therefore, in order to evaluate the coding efficiency on the DCT based coders, the proposed LBTC algorithm is combined with DCT. The proposed new coder is named as Listless block-tree DCT (DCT_LBT). It is also discussed in previous chapters that discrete Tchebichef transform (DTT) has comparable coding performance as DCT. Therefore, another coder named as Hierarchical listless DTT (HLBT_DTT) is proposed which is similar to DCT_LBT. All these block based coders have some desirable attributes like progressive image compression, precise rate control and lower complexity. This makes DCT_LBT/HLBT_DTT to be an ideal candidate for modern multimedia applications.

5.2 The Proposed LBTC Algorithm

Generally, the encoding process of a wavelet transformed image is performed bit-plane by bit-plane. In order to do this, an appropriate threshold is pre-computed. Considering the transformed image as an indexed set of transformed coefficients c_{ij} located at i^{th} row and j^{th} column, the initial threshold is:

$$T = \left\lfloor log_2(max_{i.j}\{\,|c_{i,j}|\,\})\right\rfloor \tag{5.1}$$

The new coder presented here uses eight class of markers for partitioning a block tree. These are defined below:

- BIP: The pixel is insignificant or untested for this bit plane.
- BNP: The pixel is newly significant and it will not be refined for this bit plane.
- BSP: The pixel is significant and it will be refined for this bit plane.
- BDS: The pixel is the first (lowest index) child in a single tree consisting of all descendants of its parents
- BCP: Like BIP which is applied during partitioning of IS pass, but it will be tested for significance immediately during same IS pass.
- SD: The pixel is the first child (lowest index) in a composite tree consisting of all descendants of its parent block.

- SG: The pixel is the first grandchild (lowest index) in a set consisting of all grand descendants of its grandparent block.
- SN*: The following markers are used on the leading nodes of each generations of an insignificant block tree.
    - SN2: The pixel is first child of SD set. This pixel and its 16 neighbors (4×4 blocks) can be skipped.
    - SN3: The pixel is first grandchild of SD or child of SG set. This pixel and its 64 neighbors (8×8 blocks) can be skipped.
    - SN4: The pixel is first great grandchild of SD or grand child of SG set. This pixel and its 256 neighbors (16×16 blocks) can be skipped.

      ⋮
    - SN6: The pixel is 5th generation descendants of SD or 4th generation descendants of SG set. This pixel and its 4096 neighbors (64×64 blocks) can be skipped.

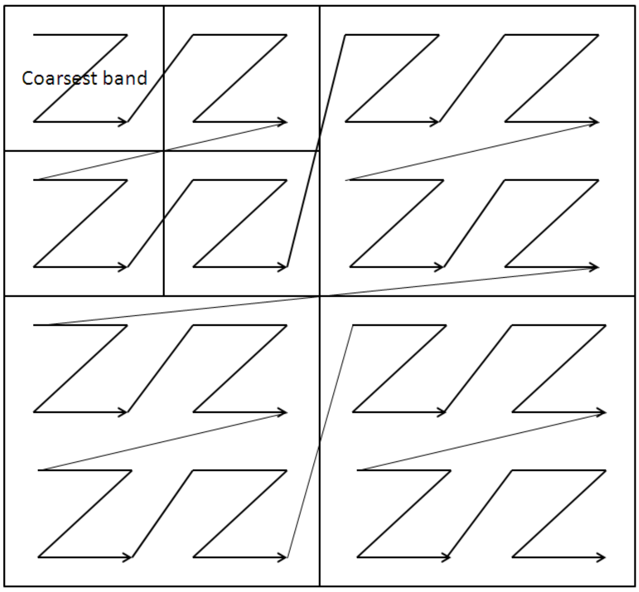


Figure 5.1: Two dimensional morton scan sequence of the transformed coefficients

The two dimensional arrangement of wavelet transformed coefficients are mapped to an one dimensional array of length I using morton scan sequencing which is shown in Figure 5.1, where two levels of wavelet decomposition has been illustrated. The algorithm scans from top most bit plane to the lowest bit plane while updating threshold (T) by half every time. In each pass, the coefficient array called *val* is examined for significance. A coefficient s is significant, if $s \geq 2^T$. The simplest method to test the significance is by bit wise AND operation. The decoder does exactly the reverse process. At the arrival of first significant bit of a coefficient, it is reconstructed as $\pm 1.5 \times 2^T$. The decoder add or subtract 2^{T-1} to its current reconstructed value depending on whether it inputs significant bit in the current pass or not. There is one to one correspondence between *markers* and *val*. This means that markers update its value each time *val* at i^{th} position becomes significance.

5.2.1 Initialization

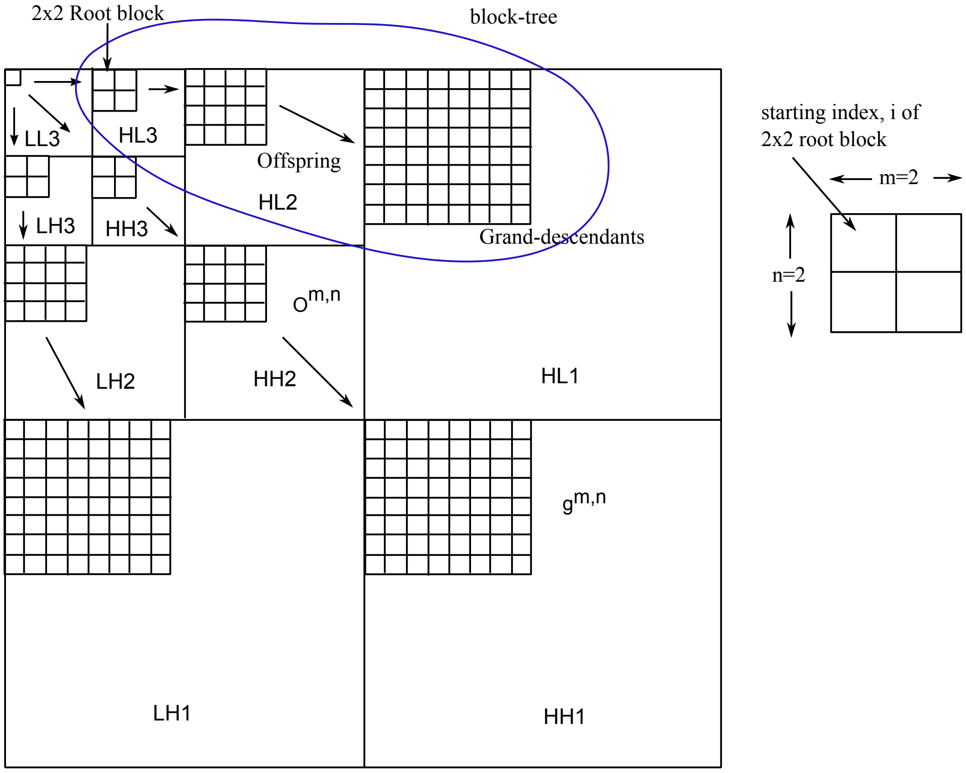


Figure 5.2: Parent-child relationship of a block-tree in a three level pyramid with initial root block size $m \times n$ shown in the right side.

Figure 5.2 shows the parent-child relationship of block-trees in a three level octave band decomposition. All the block-trees having root block size of 2×2 are shown for the sake of simplicity. During encoding, two maximum magnitude descendant arrays are computed. These are (a) $d_{max}^{m,n}(i)$ calculates the maximum descendant of a block tree which root block size is $m \times n$. Where, index 'i' is the lowest index of the root block and (b) $g_{max}^{m,n}(i)$ calculates the maximum grand descendant sets of the root block

$m \times n$. Therefore,

$$\begin{aligned} d^{m,n}_{max}(i) &= \max\Big[d^{m/2,n/2}_{max}(i),\ d^{m/2,n/2}_{max}(i+(m/2+n/2)), \\ &\qquad d^{m/2,n/2}_{max}(i+2(m/2+n/2)),\ d^{m/2,n/2}_{max}(i+3(m/2+n/2))\Big] \end{aligned} \tag{5.2}$$

$$\begin{aligned} \text{and,}\quad g^{m,n}_{max}(i) &= \max\Big[d^{m,n}_{max}(4i),\ d^{m,n}_{max}(4i+(m+n)), \\ &\qquad d^{m,n}_{max}(4i+2(m+n)),\ d^{m,n}_{max}(4i+3(m+n))\Big]. \end{aligned} \tag{5.3}$$

The descendants of a single tree is represented by:

$$d^{1,1}_{max}(i) = \max\left[val(4i),\ val(4i+1),\ val(4i+2),\ val(4i+3),\ g^{1,1}_{\max}(i)\right]. \tag{5.4}$$

Zero is substituted for $g^{m,n}_{max}(i)$ when $i \geq I/16$. The function *insert* is used to push markers SN* wherever needed by the lower tree levels when descendants are created. The *insert* function can be defined for 2 × 2 blocks as follows

$$\text{mark[4i]} = \text{SN2},\ \text{mark[16i]} = \text{SN3},\ \text{mark[64i]} = \text{SN4}, \text{...etc.} \tag{5.5}$$

For 4 × 4 blocks:

$$\text{mark[4i]} = 4 \times \text{SN2},\ \text{mark[16i]} = 4 \times \text{SN3},\ \text{mark[64i]} = 4 \times \text{SN4}, \text{...etc.} \tag{5.6}$$

A five levels of hierarchical tree have $1364 \times (m \times n)$ descendants, if the root block is of size $m \times n$. When the block tree is processed, SD marker and 5 SN* markers associated with the tree are encountered. Therefore, a large number of predictable insignificant coefficients can be skipped with little effort.

The DC (coarsest) subband block is coded using a slight different procedure than that in NLS. Assuming the number of coefficients inside DC subband are I_{dc}), NLS codes each coefficients in the DC band whereas, special markers are used for coding the DC subband. The number of markers required depend on the size of DC band. The pseudo-code for the initialization procedure is shown as follows:

Algorithm 2 DC subband initialization

```
while $i \leq I_{dc} - 1$ do
    $mark[i : I_{dc}/4 : end(i)] \leftarrow M^*$
end while
```

Algorithm 3 Initialization of remaining 3 highest subbands

```
for all subbands do
    $mark[I_{dc} : 16 : 4(I_{dc} - 1)] \leftarrow SD$
end for
```

Markers M* are updated and the corresponding coefficients inside the subband are coded each time these become significant. It can be noted that markers M* are only used to partition the DC subband. The coefficients from I_{dc} to $(4I_{dc} - 1)$ are marked as SD at a step of 16 or (4×4) blocks. This is because these are the offspring of block-trees having their root blocks (sizes are 2×2) lying in DC subband. A further compression is achieved as more blocks are likely to be insignificant on earlier passes using this techniques.

A function named 'skip' is used to indicate how many coefficients need to be skipped when a marker is encountered during scanning process.
For (2×2) initial root block, skip[SD]=16 if the marker is SD. Similarly, skip[S*P]=1 for BIP,BNP and BSP. skip[SG]=64 for marker SG; For SN*, skip[SN2]=16, skip[SN3]=64, ..., skip[SN6]=4096.
If the marker is SD, skip[SD]=64, skip[S*P]=1, skip[SG]=256, skip[SN2]=64, skip[SN3]=256, ... skip[SN6]=16385 for (4×4) initial root block.

It is to be noted that the pseudo-code presented in Section 5.2.2 is exclusively for LBTC having initial root block size of (2×2). Additional markers are to be inserted into the algorithm during partitioning a block tree having initial root block size of (4×4) to (2×2). Then, the block tree of initial root block size of (2×2) is to be processed until it becomes a single tree at a bit plane pass. Once it is partitioned into a single tree, the root of the descendant is to be marked as BCP. Then, the algorithm switches to the one proposed for LBTC(2×2).

Therefore, in general, the algorithm uses additional $(n - 1)$ markers for each increase of root block size $(2^n \times 2^n)$ blocks, where $n = 1, 2, 3...$ etc.

5.2.2 The Pseudo-code of LBTC(2×2) Encoder Algorithm

```
i = 0,  while i ≤ I
Step1 : Insignificant pixel pass

  if mark[i] ← BIP                                    if insignificant
    output(d ← $val[i]$AND s)                     send the coeff significance
    if d,                                                 if significance
      output(sign[i])
      mark[i] ← BNP                                 mark as newly significant
      i = i + 1                                         move past the coeff
    else,
      i ← i + skip(mark[i])                         skip to the next block/set
  end.
```

```
Step2 : Insignificant set pass

1.  If mark[i] $\leftarrow$ SD                                              set of descendants
     output (d $\leftarrow d^{m,n}_{max} \lfloor \frac{i}{4} \rfloor$ AND s)          send the block tree significance
     if d,                                                                     if significance
        mark[i] $=$ mark[i + (m × n)] $\leftarrow$ BDS                                    quad split
        mark[i + 2(m × n)] $=$ mark[i + 3(m × n)] $\leftarrow$ BDS
        mark[4i] $\leftarrow$ SG                                set of grand descendants block tree
     else,
        i = i + 4(m × n).                                         skip to the next block tree
2.  Elseif mark[i] $\leftarrow$ SG                                      if grand descendant block tree
     output(d $\leftarrow d^{m,n}_{max} \lfloor \frac{i}{16} \rfloor$ AND s)                           send the significance
     if d,
        mark[i] $=$ mark[i + 4(m × n)] $\leftarrow$ SD                          quad split and mark
        mark[i + 8(m × n)] $=$ mark[i + 12(m × n)]                      as descendant block trees
        insert(i), insert(i + 4(m × n)),                        mark boarders of these new sets
        insert(i + 8(m × n)), insert(i + 12(m × n))
     else,
        i $=$ i + 16(m × n)                       skip to the next grand descendant block tree
3.  Elseif mark[i] $\leftarrow$ BDS                                                if it is a single tree
        output(d $\leftarrow d^{1,1}_{max} \lfloor \frac{i}{4} \rfloor$ AND s)                            check for significance
     if d,                                                                    if significance
        mark[i] $=$ mark[i + 1] $\leftarrow$ BCP                                  quad split and mark each
        mark[i + 2] $=$ mark[i + 3] $\leftarrow$ BCP                                   coeff as insignificant
        mark[4i] $\leftarrow$ SD
     else, i $=$ i + 4                                     skip to the next tree
4.  Elseif mark[i] $\leftarrow$ BCP                                if insignificant during current pass
        output(d $\leftarrow val[i]$AND s)                                        send the significance
        if d, output(sign[i])                               if significant, send the sign bit
           mark[i] $\leftarrow$ BNP                                            mark as newly significant
        else, mark[i] $\leftarrow$ BIP                                               else insignificant
        i = i + 1                                                   go to the next coefficient
5.  Else
        i $=$ i + 1                                                              skip to the next coeff

    Step3 : Refinement pass

    i = 0, while i $\leq$ I
    1.  If mark[i] $\leftarrow$ BSP                                                 significant coeff
            output(val[i]AND s)                                                  refine the coeff
            i $=$ i + 1                                                     move past the coeff
    2.  Elseif, mark[i] $\leftarrow$ BNP                                            newly significant coeff
            mark[i] $\leftarrow$ BSP                                             significantfor next pass
            i $=$ i + 1                                                      move past the coeff
    3.  Elseif, mark[i] $\neq$ BIP                                                            if SN$*$
            i $=$ i + skip(mark[i])                                   skip to the next block/set
    4.  Else, i $=$ i + 1                                                    move  past the coeff
         End.
```

5.3 The Proposed DCT_LBT Embedded Encoder

The block diagram of the proposed DCT_LBT embedded coder is shown in Figure 5.3. First, the input image is divided into non-overlapping $N \times N$ blocks. Then each block is transformed using DCT. The DCT coefficients of each block are arranged in a wavelet like hierarchical manner. Though there exists a number of possible arrangement of coefficients, few arrangements are described below.

Case-1: The DCT coefficients are arranged in M levels of wavelet like hierarchical arrangement (e.g., $M = 4$ if $N = 16$; $M = 3$ if $N = 32$ or 8). The decision logic roots the coefficients of coarsest subband to the input. The coarsest subband is further divided into $N \times N$ blocks. Another M_1 level of wavelet like hierarchical rearrangement

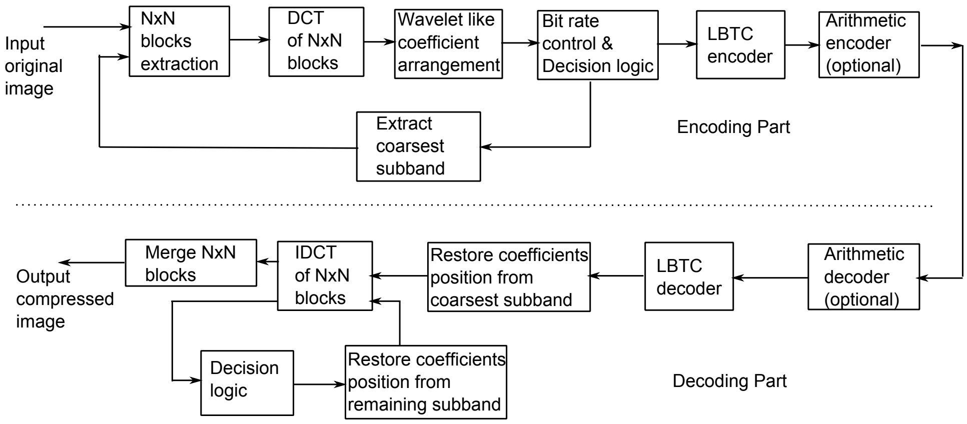

Input original image
NxN blocks extraction
DCT of NxN blocks
Wavelet like coefficient arrangement
Bit rate control & Decision logic
LBTC encoder
Arithmetic encoder (optional)
Extract coarsest subband
Encoding Part
Output compressed image
Merge NxN blocks
IDCT of NxN blocks
Restore coefficients position from coarsest subband
LBTC decoder
Arithmetic decoder (optional)
Decision logic
Restore coefficients position from remaining subband
Decoding Part

Figure 5.3: Block diagram of proposed DCT_LBT embedded image coder

(e.g., $M_1 = 1$ if $N = 16$; $M_1 = 2$ if $N = 32$ or 8) is made by reapplying DCT to the $N \times N$ blocks of coarsest subband. The resultant rearrangement of DCT coefficients makes the coarsest level to be of size 16×16 with an overall five levels of hierarchical arrangement in a standard 512×512 size monochrome image. The coefficients are converted to integers and quantized by the proposed LBTC algorithm. For image reconstruction, exactly the same reverse process is carried out at the decoder side.

Case-2: The input image is dived into 32×32 blocks. The DCT coefficients of each block are arranged into 5 levels of hierarchical arrangement. Coefficients present in the similar subband of all 32×32 blocks are grouped together. This makes an overall of 5 levels of arrangement in a standard 512×512 size image. This types of arrangement of coefficients has already been reported [40],[42],[41],[39],[44]. The function of decision logic is simply to root the overall coefficient arrangement to the proposed LBTC coding algorithm instead of rooting into the feedback path. The rate control logic decides the exact target rate at which the image is to be compressed.

5.3.1 Relation Between Transformed Coefficients

Figure 5.4 shows the arrangement of 8×8 DCT coefficients in a typical 3-level wavelet pyramid structure. After labeling 64 coefficients in a 8×8 block, the parent child relationship is explained as follows

The parent of coefficient i is $\lfloor \frac{i}{4} \rfloor$ for $1 \leq i \leq 64$, while the set of four children associated with coefficient j is $\{4j, 4j+1, 4j+2, 4j+3\}$ for $1 \leq j \leq 15$. The DC coefficient 0 is the root of DCT coefficients tree, which has only three children: coefficients 1,2 and 3. In the proposed structure, offspring corresponds to direct descendants in the same spatial location in the next finer band of the pyramid. A tree corresponds to a node having 4 children which always form a group of 2×2 adjacent pixels. In

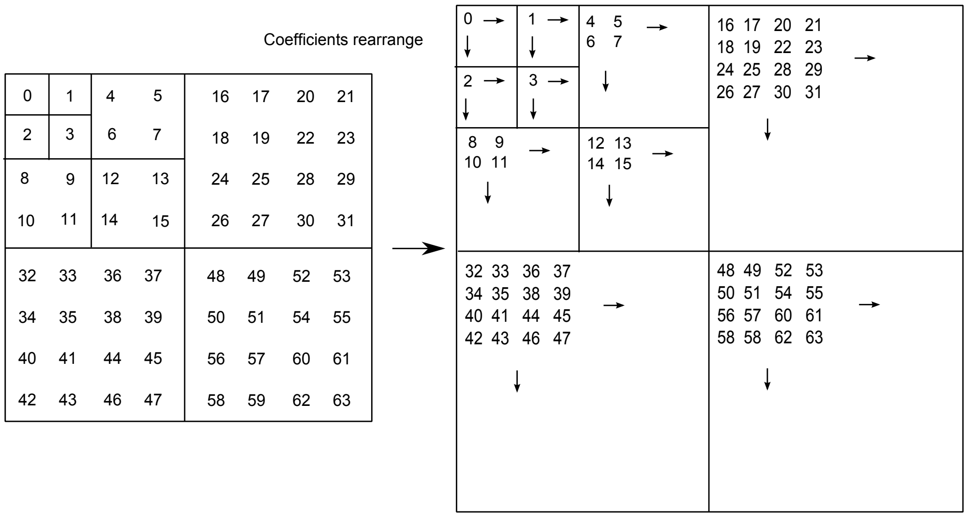


Figure 5.4: Algorithm for rearranging the transformed coefficients (A three levels of coefficient arrangement is shown here for illustration).

Figure 5.4, arrows indicate that the same index coefficients of other 8 × 8 blocks are grouped together. Further, one or two levels (depending upon the block size) of coefficient rearrangement is performed on the coarsest band in the proposed coefficient rearrangement algorithm. Therefore, five levels of DCT coefficients are subjected to be processed by the proposed algorithm.

5.4 The proposed HLBT_DTT Embedded Coder

The block diagram of HLBT_DTT embedded coder is similar to DCT_LBT embedded coder except the following differences:

- 16 × 16 DCT is replaced by 16 × 16 DTT.
- A hybrid kind of scanning order is used instead of morton scanning order.
- The coefficient arrangement algorithm is same as defined for DCT_LBT in Case 1. The refinement pass is executed in the first step instead of last one. This modification causes a reordering of the information into the encoded bit strings.

5.4.1 Coefficient Scanning Order

After arranging the DTT coefficients into five levels like the one for DCT_LBT, the coefficients are converted to integers and quantized by LBTC coding algorithm. LBTC coding algorithm scans the coefficients using the pattern as shown in Figure 5.5. Morton scanning is performed in the highest subbands. Vertical and horizontal snake scanning are performed in lower HL and LH subbands. Zigzag scanning is performed in lower HH subbands. The parent-child relationship among the coefficients among the different scales are same as mentioned in DCT_LBT. For image reconstruction, the reverse operation is done at the decoder side.

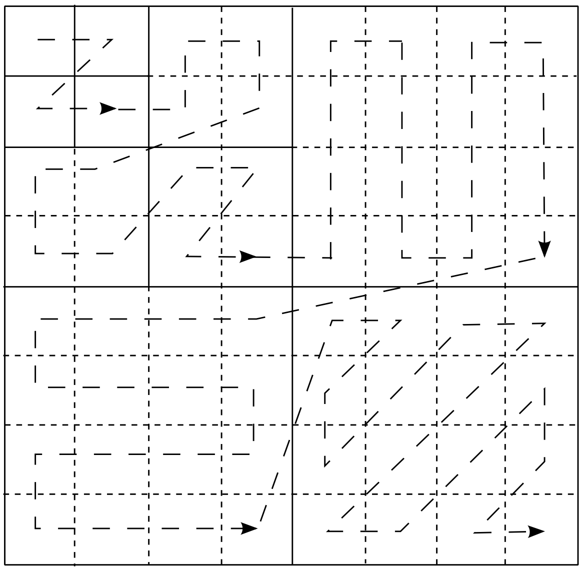


Figure 5.5: Scanning order of hierarchical subbands

.

5.5 Memory Allocation

The number of coefficients in the DC band is $I_{dc} = R_{dc} \times C_{dc}$, where $R_{dc} = R \times 2^{-L}$, $C_{dc} = C \times 2^{-L}$. R is the number of rows, C is the number of columns and L is the number of subband decomposition levels. The number of block trees in a root block of size (2×2), (4×4), and (8×8) required for LBTC is $1/4^{th}$, $1/16^{th}$ and $1/64^{th}$ of that required by NLS respectively. This reduces the initial cost by 25%, 6.25% and 1.25% respectively compared to NLS.

The coefficients are stored in one dimensional array of length I. If W bits are needed for each sub band coefficients, then the bulk storage memory required is IW for the subband data and $RC/2$ (half byte per pixel) for the eight state table markers. Therefore,

$$M_{LBTC} = IW + RC/2 \tag{5.7}$$

Hence, from Eqn. 5.7, the total memory required by LBTC is 704 kB (assuming W=18 bits/coefficients) which is 22.3% more than the image alone.

5.5.1 Dynamic Memory Requirement Comparison

In case of SPIHT, assuming total number of list entries (LIP and LSP) is twice the number of coefficients RC and LIS $\simeq RC/4$; at the end of all passes, the total memory required by SPIHT is

$$M_{SPIHT} = \frac{RC}{4} \times (8W + 1) \tag{5.8}$$

The total number of list entries are less than the number of coefficients because of block nature of WBTC. Therefore, total memory required at any time is less than SPIHT. Though memory required by WBTC is approximately $1/3^{rd}$ of SPIHT for some earlier bit plane passes, by the end of all passes the memory required is almost 85%-95% as that of SPIHT. Therefore, the worst case memory required by WBTC is

$$M_{WBTC} =\sim 0.95 \times M_{SPIHT}. \tag{5.9}$$

For a 512×512 image with 18 bits per coefficient, this is $\frac{(512\times512)}{1024}\times\frac{1}{2}$ (bytes/coeff)=128 kB for LBTC, 1160 kB for SPIHT and 1100 kB for WBTC.

5.6 Experimental Results and Performance Comparison

The performance of LBTC, DCT_LBT and HLBT_DTT algorithms are compared in terms of coding efficiency and computational complexity with other algorithms such as SPIHT, WBTC, NLS, JPEG 2000, Song and Cho[44] and Hou *et al.*[43] on standard monochrome images. The implementation is done in MATLAB 7.10.0 under Window XP, Intel Core 2 Duo CPU with 3 GHz speed and 4 GB of RAM space.

5.6.1 Coding Efficiency of LBTC Embedded Wavelet Coder

The performance of LBTC algorithm is evaluated on two sets of standard 8 bpp monochrome images. The first set includes Lena and Barbara images of size 512×512. The second set includes higher resolution images from JPEG 2000 test set which are Bike and Woman of size 2560 × 2048. Five levels of dyadic wavelet decomposition are carried out using 9/7-biorthogonal filter [90] with symmetric extension at the image edges.

In order to evaluate the coding performance, we first compare the cumulative number of bits generated in different passes of NLS and proposed LBTC(2 × 2) as well as LBTC(4×4). The results are shown in Table 5.1 for Lena and Barbara images. It is evident form the Table 5.1 that LBTC(2 × 2) saves 51.3%, 37.7%, 22.3%, 9.8%, 3.6%, 1.4% of bits, and LBTC(4 × 4) saves 58.9%, 43.1%, 25%, 10.9%, 3.8%, 10%, 1.4% of bits respectively on top six bit plane passes. This gives rise to a Peak-signal-to-noise-ratio (PSNR) improvement of (1.69-2.01 dB), (1.68-1.79) dB, (0.80-0.88) dB and (0.22-0.24) dB over NLS. On Barbara image, LBTC(2×2) saves 64.6%, 42%, 23.4%, 10%, and LBTC(4×4) saves 72.3%, 42%, 26.3% of bits respectively on top four passes. Therefore, the PSNR improvement is (2.08-2.29) dB, (1.62-1.83) dB, (0.47-0.51) dB, and 0.22 dB for the corresponding passes respectively. This indicates that LBTC is more efficient in sorting significant coefficients than NLS. It can be estimated form Figure 5.6 that the percentage of bit saving in LBTC is nearly same as WBTC in most of the images.

Table 5.1: Comparison of encoded string length between NLS and LBTC for Lena and Barbara Images on top six cumulative bit plane passes

Images	No. of sorting passes	NLS encoding string-length(no. of significant coeff.)	LBTC(2×2) encoding string-length(no. of significant coeff.)	LBTC(4×4) encoding string-length(no. of significant coeff.)	PSNR(dB) improvement LBTC(2×2) vs. NLS	PSNR(dB) improvement LBTC(4×4) vs. NLS
Lena	1	481(29)	234(29)	198(29)	1.69	2.01
	2	1041(106)	649(106)	593(106)	1.68	1.79
	3	2015(262)	1566(262)	1511(262)	0.80	0.88
	4	4296(571)	3874(571)	3825(571)	0.22	0.24
	5	10483(1422)	10105(1422)	10087(1422)	0.17	0.17
	6	23754(3367)	23430(3367)	23420(3367)	0.07	0.07
Barbara	1	469(19)	166(19)	130(19)	2.08	2.29
	2	1023(98)	593(98)	532(98)	1.62	1.83
	3	1952(256)	1496(256)	1440(256)	0.47	0.51
	4	4139(564)	3725(564)	3670(564)	0.22	0.22
	5	11444(1544)	11100(1544)	11045(1544)	0.06	0.07
	6	44476(17756)	44150(17756)	44095(17756)	0.04	0.04

Table 5.2 shows comparison of PSNR values between SPIHT, NLS, WBTC(2×2), LBTC(2×2), LBTC(4×4) and LBTC(8×8) for the same set of images at very low bit rates. It can be observed that the average PSNR loss of LBTC(2×2) over WBTC(2×2) is 0.126 dB on Lena image, whereas the average PSNR gain of LBTC(2×2) over WBTC(2×2) on Barbara image is 0.473 dB on 0.005-0.1 bit rates. However, the average rate distortion (R-D) performance increases slightly over WBTC by increasing the root block size of LBTC.

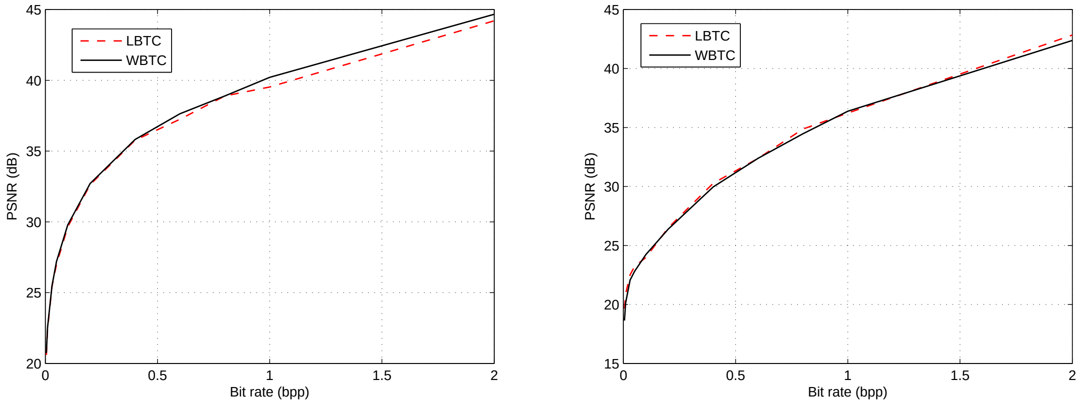
In Figure 5.6, it is observed that the R-D performance of Lena and Barbara images using LBTC(2×2) algorithm is nearly same as WBTC (2×2) on (0.0-2.0) bpp. However, on Lena image, a slight fall in performance is observed in LBTC at higher bit rates. On the contrary, LBTC shows an improvement in performance over all the considered bit rates in Barbara image.

Table 5.3 shows the comparisons of LBTC algorithm with SPIHT, NLS and WBTC algorithms for (0.0625-2.0) bit rates on Woman and Bike images without arithmetic coding. It can be seen that LBTC shows a coding gain of (0.08-1.42) dB with respect to WBTC, and (0.38-1.67) dB with respect to SPIHT over bit rates (0.0625-0.5) bpp

Table 5.2: Comparison of PSNR(dB) values between SPIHT, NLS, WBTC and LBTC at very low bit rates (All the results are without arithmetic coding)

Images	Algorithm	Bit rate (bpp)				
		0.005	0.01	0.03	0.05	0.1
Lena	SPIHT	19.46	21.94	25.22	27.06	29.71
	NLS	19.39	21.97	25.20	26.97	29.61
	WBTC(2 × 2)	20.74	22.52	25.48	27.19	29.78
	WBTC(4 × 4)	20.80	22.54	25.49	27.20	29.79
	WBTC(8 × 8)	20.82	22.55	25.50	27.20	29.79
	LBTC(2 × 2)	20.60	22.40	25.44	27.00	29.64
	LBTC(4 × 4)	20.77	22.51	25.45	27.00	29.64
	LBTC(8 × 8)	20.79	22.51	25.45	27.00	29.64
Barbara	SPIHT	17.99	19.84	21.91	22.58	23.95
	NLS	18.86	20.53	22.42	23.18	23.98
	WBTC(2 × 2)	18.64	20.12	22.06	22.80	24.23
	WBTC(4 × 4)	18.82	20.22	22.21	22.89	24.25
	WBTC(8 × 8)	19.00	20.32	22.29	22.95	24.29
	LBTC(2 × 2)	19.66	20.88	22.51	23.21	23.98
	LBTC(4 × 4)	19.71	20.93	22.52	23.22	23.98
	LBTC(8 × 8)	19.73	20.95	22.52	23.22	23.98

on Woman image. However, the gain in LBTC falls by (0.91-1.0) dB and (0.59-0.73) dB over rates (1.0-2.0) bpp on WBTC and SPIHT respectively. Comparing with NLS, LBTC shows a coding gain of (0.01-0.1) dB over the considered bit rates on Woman image. In Bike image, LBTC shows a coding gain of (0.01-0.43) dB on WBTC and (0.26-0.63) dB on SPIHT for bit rates (0.0625-0.25) bpp. However, the coding gain reduces to approximately (0.34-0.65) dB in WBTC and (0.19-0.51) dB in SPIHT over bit rates (0.5-2.0) bpp. LBTC consistently shows a PSNR gain between (0.01-0.09)



(a) (b)

Figure 5.6: Rate distortion performance comparison of WBTC(2 × 2) and LBTC(2 × 2) algorithms on (a) Lena and (b) Barbara images. (No back-end arithmetic coding employed)

dB over NLS in all the bit rates on Bike image.

Table 5.3: Comparison of PSNR(dB) values between SPIHT, NLS, WBTC and LBTC algorithms (All the results are without arithmetic coding).

Images	Algorithm	Bit rate (bpp)					
		0.0625	0.125	0.25	0.5	1.0	2.0
Woman	SPIHT	25.07	26.91	29.43	32.93	37.73	43.21
	NLS	26.61	28.34	30.56	33.30	36.99	42.60
	WBTC	25.29	27.08	29.71	33.23	37.91	43.62
	LBTC	26.71	28.38	30.59	33.31	37.00	42.62
Bike	SPIHT	22.86	25.29	28.51	32.36	36.98	42.90
	NLS	23.40	25.86	28.75	32.15	36.46	42.53
	WBTC	23.14	25.48	28.76	32.51	37.12	43.13
	LBTC	23.49	25.91	28.77	32.17	36.47	42.55

Table 5.4 shows the PSNR performance comparisons of LBTC with JPEG 2000, SPIHT, NLS and WBTC algorithms using arithmetic coding [109] for the same set of images. Comparing with JPEG 2000, LBTC shows a significant gain (0.28-1.46) dB over (0.0625-0.5) bpp on Woman image. However, LBTC shows a coding gain of 0.12 dB at 0.0625 bpp and a loss of (0.04-0.72) dB over (0.125-2.0) bpp in Bike image. WBTC shows poor performance than LBTC below 0.5 bpp. At higher rates (approx. ≥ 1.0 bpp), WBTC outperforms LBTC by a wide margin. In nutshell, LBTC outperforms most of the algorithm at lower rates (typically ≤ 1.0 bpp) and under performs at higher rates in most of the images. One possible reason for reduction of PSNR values at higher rates is that more bits are required to indicate the significance of block-tree as well as each tree inside a block-tree as more and more coefficients became significant with decreasing thresholds. This leads to a cumulative increase of bit string length at the encoder output. Bits indicating significance of block-trees are required, but these bits don't carry any information related to PSNR. This leads to reduction of zero distortion with non zero increase of bit rate. This effect is more dominant in images of higher dimensions. The reason is that higher dimension images have more block-trees as compared to lower dimension images.

5.6.2 Coding Efficiency of DCT_LBT Embedded Coder

Table 5.5 shows a comparison of cumulative number of bits generated on top six bit plane passes of Lena and Barbara images using DCT_LBT coder. It is observed that DCT_LBT(4×4) encoder saves 6.0%-82.8% of bits on Lena and 3.8%-92.5% of bits on Barbara images for the considered range of bit plane passes. This gives rise to a PSNR gain of (0.27-2.85) dB and (0.17-2.66) dB on Lena and Barbara images respectively.

Input image partitioned into (8×8) blocks, (16×16) blocks, and (32×32) blocks-I

Table 5.4: Comparison of PSNR(dB) values between JPEG 2000, SPIHT, NLS, WBTC and LBTC algorithms (All the results are with arithmetic coding).

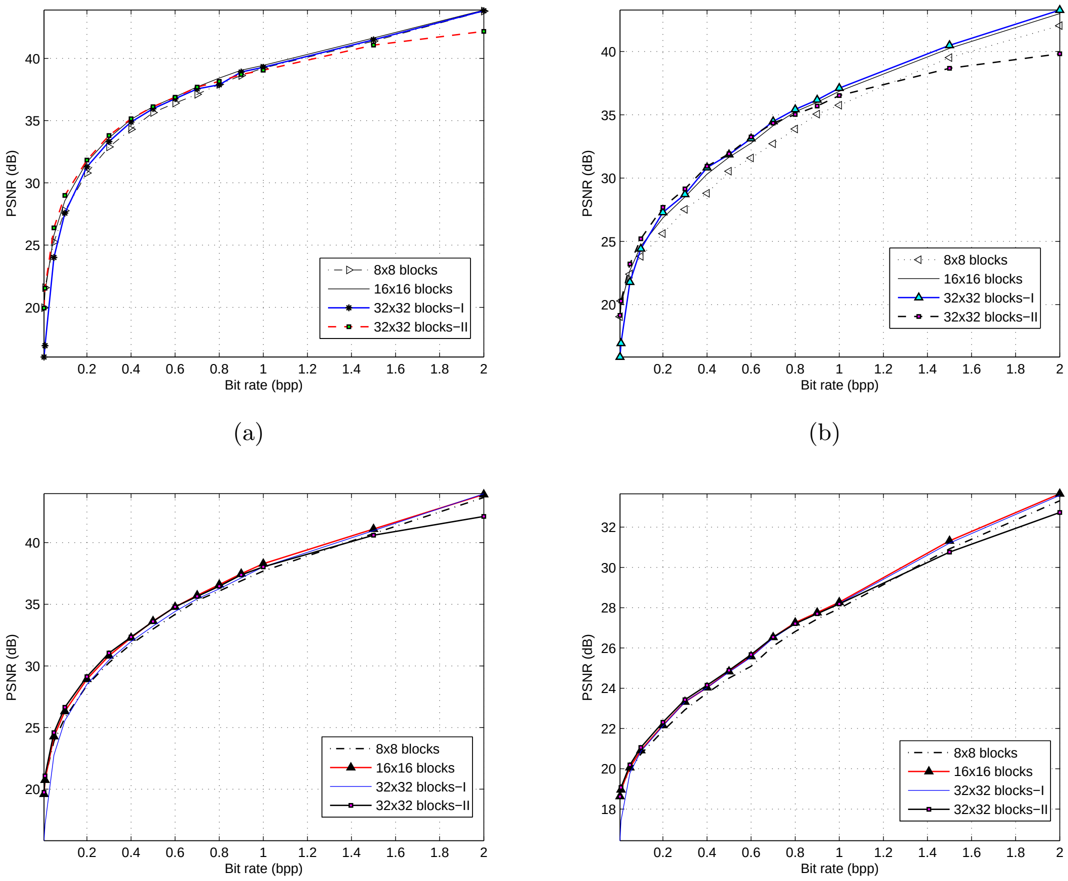
Images	Algorithm	Bit rate (bpp)					
		0.0625	0.125	0.25	0.5	1.0	2.0
Woman	JPEG2000	25.59	27.33	29.95	33.57	38.28	43.97
	SPIHT	25.50	27.33	29.94	33.59	38.28	43.99
	NLS	26.89	28.74	30.86	33.77	37.88	43.60
	WBTC	25.67	27.34	30.05	33.73	38.30	44.10
	LBTC	27.05	28.98	30.96	33.85	37.92	43.62
Bike	JPEG2000	23.74	26.31	29.56	33.43	37.99	43.95
	SPIHT	23.44	25.89	29.12	33.01	37.70	43.80
	NLS	23.70	26.20	29.25	32.75	37.22	43.23
	WBTC	23.61	25.97	29.29	33.17	37.82	43.93
	LBTC	23.86	26.27	29.31	32.91	37.27	43.27

Table 5.5: Comparison of cumulative encoded string length between DCT_NLS and DCT_LBT for Lena and Barbara images on top six bit plane passes

Images	No. of sorting passes	DCT_NLS encoding string-length(no. of significant coeff.)	DCT_LBT(2 × 2) encoding string-length(no. of significant coeff.)	DCT_LBT(4 × 4) encoding string-length(no. of significant coeff.)	% of bit saving DCT_LBT(4 × 4) vs. DCT_NLS	PSNR(dB) improvement
Lena	1	458(6)	115(6)	79(6)	82.8	2.85
	2	925(25)	301(25)	229(25)	75.3	2.46
	3	1473(76)	653(76)	561(76)	62.0	1.43
	4	2799(243)	1949(243)	1853(243)	33.8	0.74
	5	6544(761)	5708(761)	5623(761)	14.1	0.54
	6	15082(1963)	14268(1963)	14188(1963)	6.0	0.27
Barbara	1	453(1)	70(1)	34(1)	92.5	2.66
	2	926(21)	240(21)	172(21)	81.4	2.31
	3	1452(70)	554(70)	461(70)	68.3	1.32
	4	2873(252)	1980(252)	1884(252)	34.4	0.75
	5	8280(822)	7469(822)	7376(822)	10.9	0.34
	6	23375(2486)	22577(2486)	22484(2486)	3.8	0.17

use the coefficient rearrangement algorithm as explained in Case-1; whereas, (32×32) blocks-II uses the coefficient arrangement as explained in Case-2 of Section 5.3. Figure 5.7 shows the coding performance of DCT_LBT(4×4) algorithm on (a) Lena, (b) Barbara, (c) Boat and (d) Mandrill images using the above block sizes associated with their respective coefficient arrangements. It is clearly evident from the results that (16×16) block sizes show a good trade-off of coding performance over the range of considered bit rates on most of the images. (32×32) blocks-II shows slightly better performance at lower rates. However, its performance is seriously affected at higher rates which is typically ≥ 1.0 bpp in most of the standard images.

Figure 5.8 (a)-(d) displays the decoded cropped portions of Bike image at 0.1 bpp using DCT_SPIHT, DCT_NLS, DCT_LBT(2×2) and DCT_LBT(4×4) coding schemes. It is to be noted that the blocking effect is manifested in the above simulations. These effects vanishes gradually with the increase of bit rate. Figure 5.9 (a)-(d)



(a)

(b)

(c)

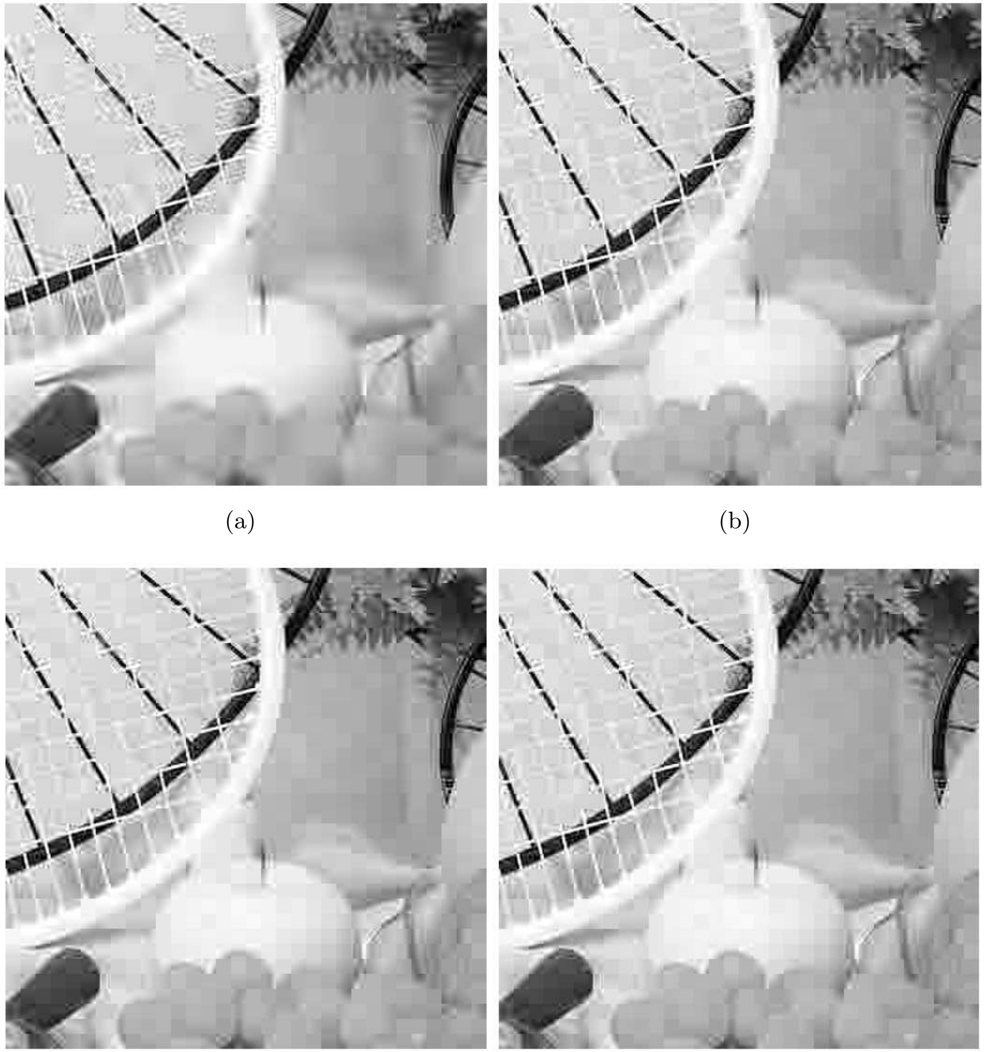
(d)

Figure 5.7: Rate distortion performance comparison of DCT_LBT (4 × 4) algorithm for input block sizes of (8 × 8), (16 × 16), (32 × 32)blocks-case I, (32 × 32)blocks-case II on (a) Lena (b) Barbara (c) Boat and (d) Mandrill images without using arithmetic coding.

shows the decoded cropped portions of Woman image when coded using DCT_SPIHT, DCT_NLS, DCT_LBT(2 × 2) and DCT_LBT(4 × 4) at 0.05 bpp. It is observed that the decoded images using DCT_LBT coding schemes have better perceptual qualities than other coding schemes.

Table 5.6 and 5.7 show a comparison of R-D performance of DCT_LBT(4 × 4) algorithm with other state-of-the-art DCT coders presented by Hou *et al.* and Song and Cho including JPEG 2000 for (512 × 512) size test images (Peppers, Boat, Mandrill, Lena and Barbara) and (2560 × 2048) size images (Woman, Bike and Cafe). The proposed algorithm uses input block size of (16 × 16). These binary coded version results (i.e. arithmetic coding) of other algorithms are taken directly from [44].

When compared with the results by Song and Cho in Table 5.6, it is clear that the proposed algorithm shows a PSNR gain of (0.3-1.33) dB between (0.0625-0.5)



(a) (b)

(c) (d)

Figure 5.8: Decoded Bike images compressed at 0.1 bpp using (a)DCT_SPIHT, (b)DCT_NLS, (c)DCT_LBT(2×2), (d)DCT_LBT(4×4) respectively.

bpp and a loss of (0.37-0.76) dB between rates (1.0-2.0) bpp on Woman image. In case of bike image, the gain is (0.06-1.27) dB over (0.0625-0.5) bpp and a loss of (0.9-0.98) dB above 0.5 bpp is observed. Similarly, a PSNR gain of (0.37-1.56) dB is achieved over (0.0625-2.0) bit rates in Cafe image. The proposed technique shows approximately same or better performance at lower bit rates ($\leq$ 0.5 bpp) and under performs at higher rates ($\geq$ 1.0 bpp) in most of the images when compared with JPEG 2000,. Improvement at lower rates is a desirable feature for browsing images over wireless lines where a significant amount of information is required at the earlier stages of transmission. Though the coding performances are lower at higher rates, the subjective quality of the image does not show any significant change even for a PSNR loss over 1.0 dB.

Table 5.6: Comparison of PSNR(dB) values between Hou *et al.*, JPEG 2000, Song and Cho and DCT_LBP(4×4) algorithms on 2560×2048 size images(with arithmetic coding)

Images	Algorithm	Bit rate (bpp)					
		0.0625	0.125	0.25	0.5	1.0	2.0
Woman	Hou *et al.*	24.90	26.69	28.63	31.70	35.98	41.34
	JPEG2000	25.59	27.35	29.99	33.62	38.42	43.99
	Song and Cho	25.60	27.29	29.84	33.38	37.89	43.77
	DCT_LBT	26.85	28.62	30.87	33.68	37.52	43.01
Bike	Hou *et al.*	22.18	24.47	27.35	30.73	34.77	39.91
	JPEG2000	23.80	26.36	29.62	33.51	38.10	43.98
	Song and Cho	23.24	25.75	28.60	32.08	36.57	42.57
	DCT_LBT	23.65	26.32	29.52	32.65	36.86	42.72
Cafe	Hou *et al.*	18.21	19.61	21.50	24.49	28.66	34.30
	JPEG2000	19.05	20.76	23.13	26.81	32.03	39.07
	Song and Cho	18.73	20.46	22.83	25.87	30.50	37.21
	DCT_LBT	19.74	21.52	23.91	27.43	31.12	37.58

In Table 5.7, DCT_LBT(4×4) shows (0.3-1.28) dB PSNR improvement over Hou *et al.*, PSNR loss of (0.04-0.8) dB with respect to JPEG 2000 and a gain of (0.03-0.50) dB with respect to Song and Cho in Pepper image. Similarly, when compared with Hou *et al.*, the proposed algorithm shows a coding gain of (0.03-1.97) dB, (0.08-0.41) dB, (0.15-0.55) dB and (0.19-0.53) dB respectively on Lena, Barbara, Mandrill and Boat images over (0.0.0625-2.0) bit rates. It can be calculated from Table 5.7 that the proposed techniques show an average reduction of 0.004 dB with [44] and 0.2 dB with JPEG 2000 over the five set of considered images. It is also worth mentioning that DCT_LBT coder shows a significant improvement of PSNR values at lower bit rates than most of the DCT based embedded coders reported in [39],[40],[41],[42],[43].

Table 5.7: Comparison of PSNR(dB) values between Hou *et al.*, JPEG 2000, Song and Cho and DCT_LBT(4 × 4) algorithms on 512 × 512 size images(All the results are with arithmetic coding)

Images	Algorithm	Bit rate (bpp)					
		0.0625	0.125	0.25	0.5	1.0	2.0
Peppers	Hou *et al.*	26.31	29.16	31.89	34.52	37.28	41.84
	JPEG2000	26.93	30.48	33.37	35.84	38.32	43.07
	Song and Cho	26.67	27.47	32.09	34.67	37.61	42.49
	DCT_LBT	26.61	30.44	32.57	35.17	37.67	42.52
Boat	Hou *et al.*	25.17	27.47	30.08	33.70	38.40	44.26
	JPEG2000	25.45	27.81	30.92	34.50	39.21	44.69
	Song and Cho	25.54	27.71	30.47	34.09	38.60	44.51
	DCT_LBT	25.52	27.66	30.52	34.12	38.93	44.66
Mandrill	Hou *et al.*	20.40	21.25	22.99	25.03	28.48	34.03
	JPEG2000	20.60	21.60	23.12	25.53	29.00	34.73
	Song and Cho	20.59	21.66	23.11	25.56	28.97	34.78
	DCT_LBT	20.55	21.56	23.21	25.43	28.66	34.58
Lena	Hou *et al.*	-	29.42	32.88	36.37	39.68	-
	JPEG2000	27.80	30.79	33.97	37.24	40.34	-
	Song and Cho	27.57	30.28	33.36	36.64	39.93	-
	DCT_LBT	27.49	30.17	33.41	36.72	40.08	44.82
Barbara	Hou *et al.*	-	25.43	28.54	32.29	37.05	-
	JPEG2000	23.17	25.25	28.30	32.17	37.12	-
	Song and Cho	24.06	26.43	29.27	32.82	37.52	-
	DCT_LBT	23.52	25.84	28.62	32.39	37.41	43.81

5.6.3 Coding Efficiency of HLBT_DTT Embedded Coder

The performance of HLBT_DTT algorithm is evaluated on two sets of standard monochrome images having bit depth 8 bpp. The first set includes Lena, Barbara and Mandrill images of size 512 × 512. The second set includes higher resolution images from JPEG 2000 test set which are Bike and Woman of size 2048 × 2560.

Table 5.8 compares the cumulative number of bits generated on top seven bit plane passes of Lena image using HLBT_DTT coder. It can be verified that HLBT_DTT(2 × 2) saves 2.57%-81.7% and HLBT_DTT(4 × 4) algorithm saves 2.6%-89.6% of bits with respect to DTT_NLS for the considered range of bit plane passes. This give rise to a PSNR gain of ((-0.05) to 2.42)dB in HLBT_DTT(4 × 4) over DCT_NLS and (0.14-4.19)dB in HLBT_DTT(4 × 4) over DTT_NLS on Lena image. However, the PSNR performance of DCT_NLS and DTT_NLS are very close to each other.

Table 5.9 shows the same kind of comparison as that in Table 5.8 for Ruler image. It is observed that the PSNR improvement of HLBT_DTT(4 × 4) is about 0.01-5.42 dB improvement up to a bit rate of $\frac{264530}{(512\times512)} \simeq 1.0$ bpp. The number 264530 is the encoding bit length up to 5th sorting pass in HLBT_DTT(4 × 4). On the other

Table 5.8: Comparison of encoded string length between DCT_NLS, DTT_NLS and HLBT_DTT for Lena image on top seven cumulative bit plane passes

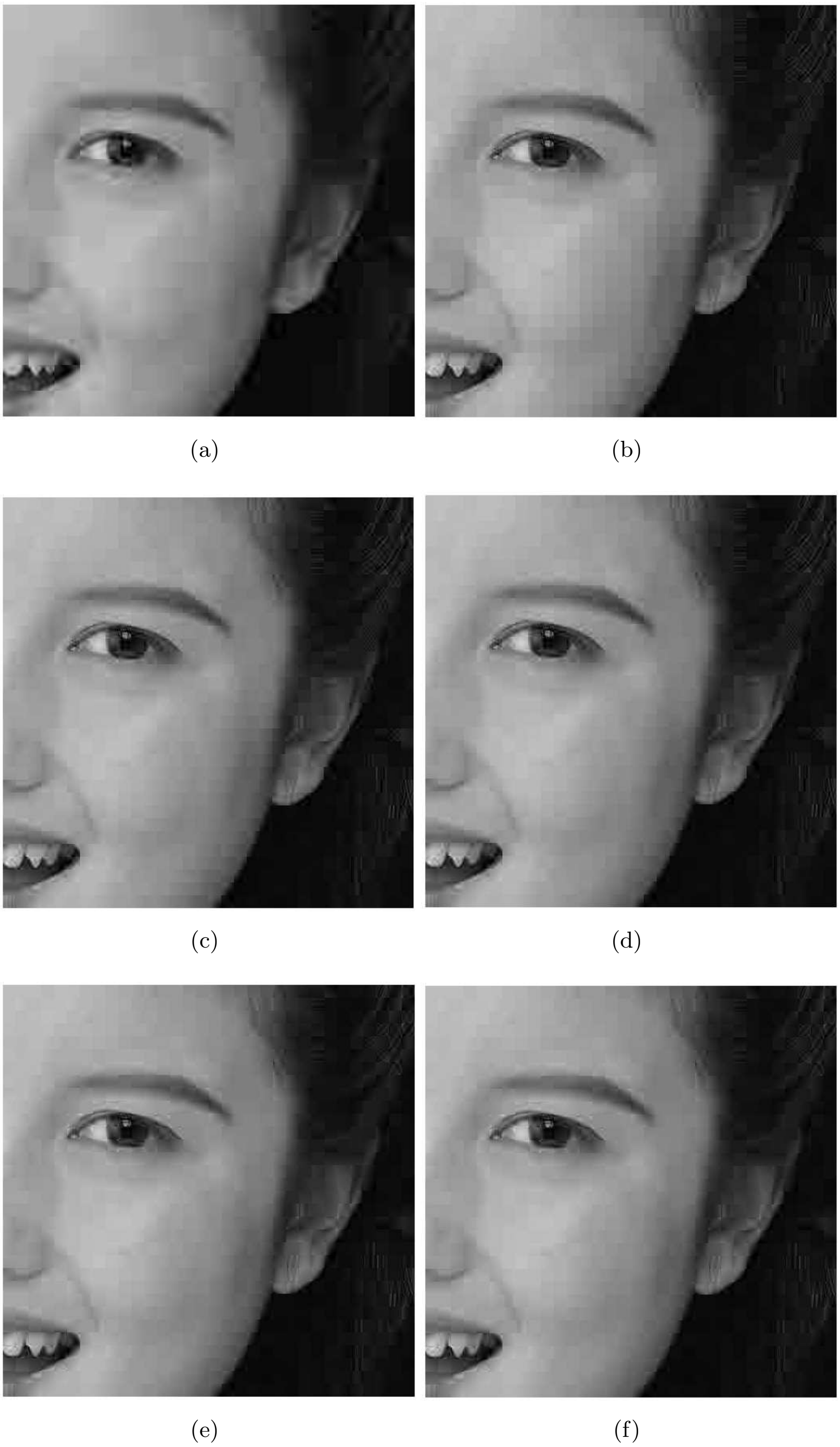
No. of sorting passes	DCT_NLS encoding string-length(no. of significant coeff)	DTT_NLS encoding string-length(no. of significant coeff)	HLBT_DTT(2 × 2) encoding string-length(no. of significant coeff)	HLBT_DTT(4 × 4) encoding string length	PSNR(dB) difference HLBT_DTT(4 × 4) -DCT_NLS	PSNR(dB) improvement HLBT_DTT(4 × 4) vs.DTT_NLS
1	458(6)	454(2)	83(2)	47(2)	2.42	3.01
2	925(25)	926(21)	269(21)	201(21)	2.20	4.19
3	1473(76)	1498(82)	659(82)	570(82)	1.35	1.71
4	2799(243)	2782(260)	1900(260)	1799(260)	0.67	0.98
5	6544(761)	6565(804)	5711(804)	5621(804)	0.48	0.55
6	15082(1963)	14938(1956)	14105(1956)	14018(1956)	0.03	0.22
7	31696(4403)	32060(4506)	31236(4506)	31152(4506)	-0.05	0.14

Table 5.9: Comparison of encoded string length between DCT_NLS, DTT_NLS and HLBT_DTT for Ruler image on top five cumulative bit plane passes

No. of sorting passes	DCT_NLS encoding string-length(no. of significant coeff)	DTT_NLS encoding string-length(no. of significant coeff)	HLBT_DTT(2 × 2) encoding string-length(no. of significant coeff)	HLBT_DTT(4 × 4) encoding string-length(no. of signif. coeff)	PSNR(dB) difference HLBT_DTT(4 × 4) -DCT_NLS	PSNR(dB) improvement HLBT_DTT(4 × 4) vs.DTT_NLS
1	706(20)	1119(39)	920(39)	925(39)	0.010	0.013
2	53268(12280)	41886(9080)	41584(9080)	41597(9080)	3.67	0.001
3	131048(22049)	84524(13906)	84196(13906)	84216(13906)	5.42	0.02
4	213970(29137)	149149(19780)	148960(19780)	148983(19780)	3.05	0.007
5	307756(37353)	264658(47394)	264507(47394)	264530(47394)	1.05	0.018

hand, the PSNR improvement is very small i.e. 0.001-0.02 in HLBT_DTT(4 × 4) with respect to DTT_NLS. It is interesting to note that HLBT_DTT(4 × 4) has slightly more encoded bit length compared to HLBT_DTT(2 × 2). The reason is that in Ruler image the probability for a coefficient being significant is large at earlier passes in contrast to natural images such as Lena, Barbara, etc. Therefore, a block tree having larger root block size needs to be quad split each time until it becomes a tree having a single root. It is evident from the proposed algorithm that the number of symbols to be inserted into the encoded bit string (i.e. number of ones) are equal to the number of times a block tree is being significant. This could be the possible reason for increase in encoded bit length in block trees of larger root blocks. Table 6.10 compares the PSNR performances of DCT_NLS, DCT_LBT of 2 × 2 and 4 × 4 root block sizes with the proposed method i.e. HLBT_DTT(4 × 4) on Lena, Barbara, Mandrill and Ruler images. It is observed that a smooth image such as Lena shows (0.03-0.24) dB of PSNR loss, textured images such as Barbara shows (0.018-0.548) dB and Mandrill shows (0.027-0.201) dB of PSNR loss. However Ruler image shows significant coding gain (i.e. 9.16 dB) which occurs at 2.0 bpp in the proposed coding technique. While comparing HLBT_DTT with different root block sizes, it is observed that HLBT_DTT with higher root block sizes has higher PSNR performance than lower root block sizes at lower rates. However, the coding gain increases slightly with increase in bit rate.

Figure 5.9 (a)-(f) shows the decoded cropped portions of Woman image when coded using DCT_SPIHT, DCT_NLS, DCT_LBT(2 × 2), DCT_LBT(4 × 4), HLBT_DTT (2 ×

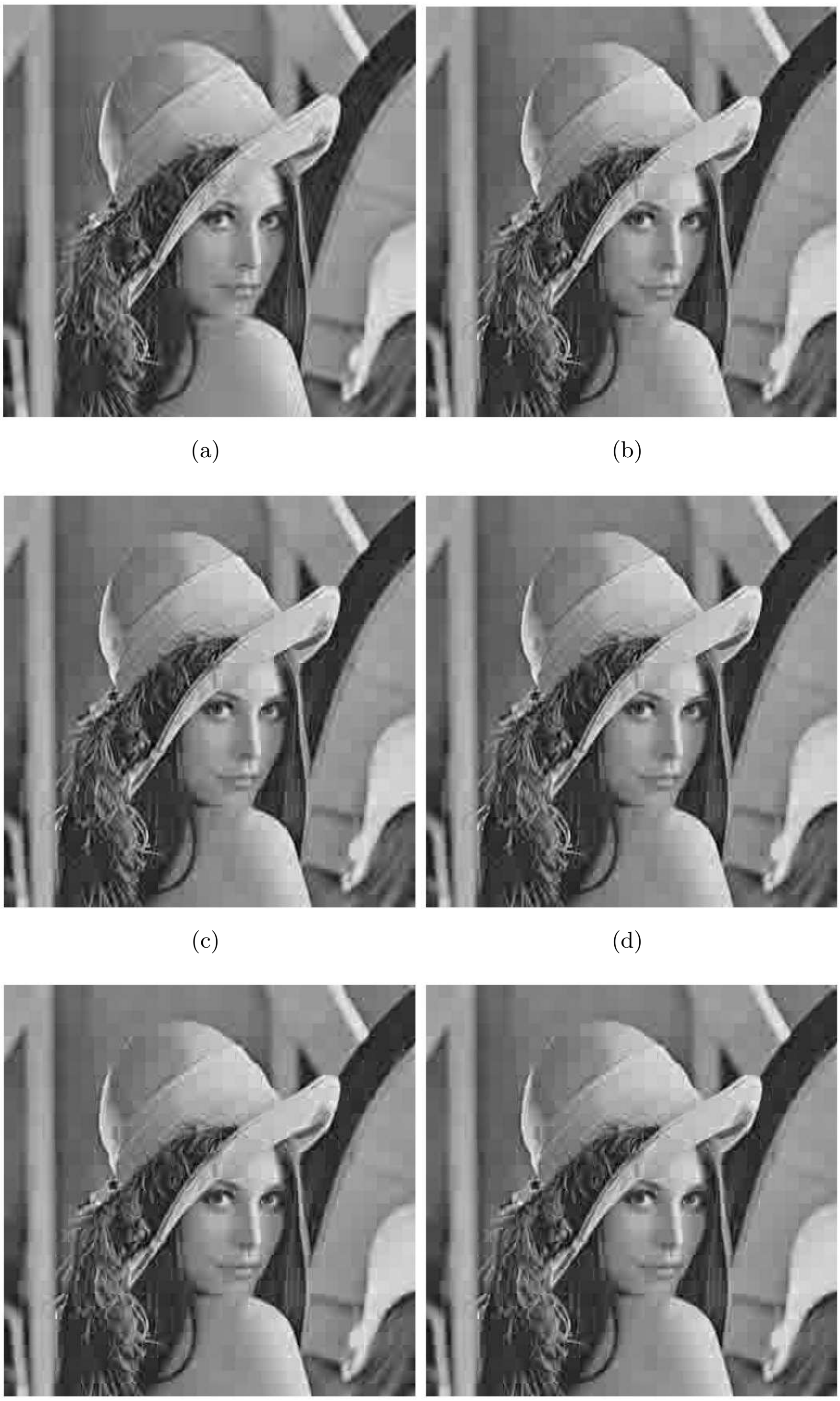


(a) (b)

(c) (d)

(e) (f)

Figure 5.9: Decoded Woman images compressed at 0.05 bpp using (a)DCT_SPIHT (PSNR=30.646 dB, MSSIM=0.8095), (b)DCT_NLS (PSNR=31.325 dB, MSSIM=0.8197), (c)DCT_LBT(2 × 2) (PSNR=31.591 dB, MSSIM=0.8369), (d)DCT_LBT(4 × 4) (PSNR=31.610 dB, MSSIM=0.8379) (e) HLBT_DTT(2 × 2) (PSNR=31.561 dB, MSSIM=0.8360) (f) HLBT_DTT(4 × 4) (PSNR=31.573, MSSIM=0.8365).

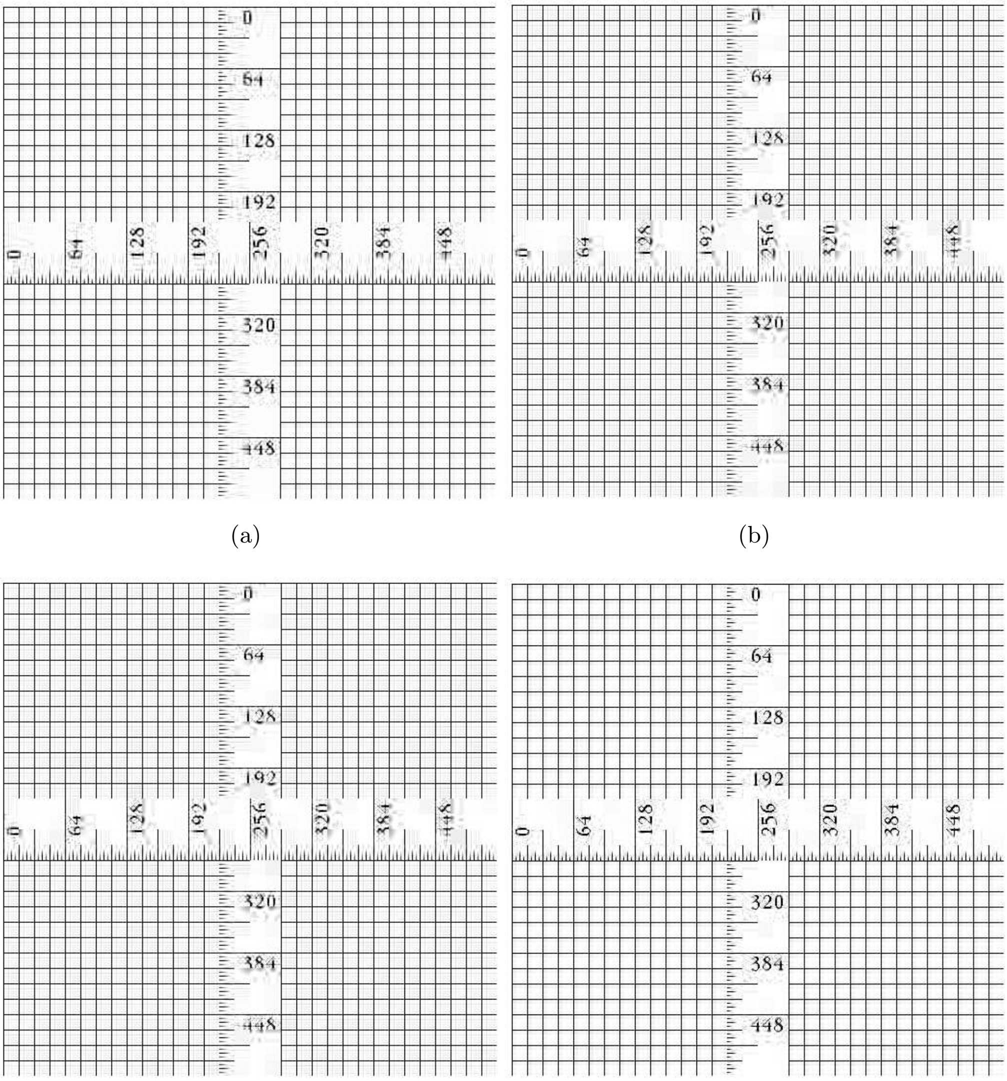


(a) (b)

(c) (d)

(e) (f)

Figure 5.10: Decoded Lena images compressed at 0.1 bpp using (a)DCT_SPIHT (PSNR=28.32 dB, MSSIM=0.8437), (b)DCT_NLS (PSNR=28.44 dB, MSSIM=0.8507), (c)DCT_LBT(2×2) (PSNR=28.57 dB, MSSIM=0.8529), (d)DCT_LBT(4×4) (PSNR=28.58 dB, MSSIM=0.8530) (e)HLBT_DTT(2 × 2) (PSNR=28.45 dB, MSSIM=0.8511) (f)HLBT_DTT(4 × 4) (PSNR=28.46, MSSIM=0.8512).



(a) (b)

(c) (d)

Figure 5.11: Decoded ruler images compressed at 0.5 bpp using (a)DCT_SPIHT (PSNR=21.90 dB, MSSIM=0.9532), (b)DCT_NLS (PSNR=23.36 dB, MSSIM=0.9700), (c)DCT_LBT(4 × 4) (PSNR=23.38 dB, MSSIM=0.9701) (d)HLBT_DTT(4 × 4) (PSNR=26.56, MSSIM=0.9872).

Table 5.10: Comparison of PSNR(dB) values between DTT_NLS, DCT_LBT(2×2), DCT_LBT(4×4) and HLBT_DTT(4×4) algorithms on 512×512 size images.

Images	Algorithm	Bit rate (bpp)							
		0.0156	0.0313	0.0625	0.125	0.25	0.5	1.0	2.0
Lena	DCT_NLS	21.63	23.98	26.52	29.45	32.75	36.09	39.42	43.86
	DCT_LBT(2×2)	22.28	24.26	26.68	29.53	32.80	36.11	39.43	43.87
	DCT_LBT(4×4)	22.31	24.27	26.69	29.54	32.81	36.11	39.43	43.87
	HLBT_DTT(4×4)	22.25	24.24	26.59	29.30	32.57	36.01	39.33	43.65
Barbara	DCT_NLS	20.23	21.38	22.98	25.09	27.81	31.62	36.83	42.97
	DCT_LBT(2×2)	20.71	21.74	23.00	25.12	27.84	31.64	36.85	42.99
	DCT_LBT(4×4)	20.77	21.76	23.01	25.13	27.85	31.65	36.85	42.99
	HLBT_DTT(4×4)	20.71	21.87	22.93	24.95	27.68	31.40	36.66	42.82
Mandrill	DCT_NLS	19.00	19.59	20.17	21.19	22.62	24.80	28.24	33.63
	DCT_LBT(2×2)	19.18	19.68	20.24	21.24	22.66	24.82	28.26	33.66
	DCT_LBT(4×4)	19.20	19.69	20.25	21.24	22.66	24.82	28.26	33.66
	HLBT_DTT(4×4)	19.21	19.72	20.27	21.24	22.52	24.74	28.17	33.56
Ruler	DCT_NLS	10.44	10.93	11.82	13.58	16.98	23.37	30.21	34.09
	DCT_LBT(2×2)	10.48	10.96	11.85	13.60	16.98	23.38	30.23	34.09
	DCT_LBT(4×4)	10.48	10.97	11.86	13.61	16.99	23.38	30.22	34.09
	HLBT_DTT(4×4)	10.56	11.31	13.07	16.33	21.50	26.56	31.35	43.25

2) and HLBT_DTT (4×4) at 0.05 bpp. It is observed that the decoded images using HLBT_DTT coding scheme have nearly similar visual performance over DCT_LBT coding scheme. Similar kind of performance is also noticed for Lena image in Figure 5.10 (a)-(f). For example, the shoulder portion and hat edges of Lena image in (c) and (d) show better visual impression than that of (a) and (b). In Figure 5.10, HLBT_DTT based coding techniques e.g., (e) and (f) show similar performance like DCT_LBT in (c) and (d).

The performance of HLBT_DTT algorithm can be well judged in Ruler image which is shown in Figure 5.11. It is observed that HLBT_DTT decoded ruler image has better PSNR/MSSIM values than the other three coding schemes such as DCT_SPIHT, DCT_NLS and DCT_LBT. For example, the PSNR gain of HLBT_DTT is 3.18 dB higher compared DCT_LBT, 3.20 dB higher as compared to DCT_NLS and 4.66 dB higher as compared to DCT_SPIHT at 0.5 bpp. Similar perceptual enhancement (subjective quality) also verifies the coding efficiency of HLBT_DTT algorithm.

Table 5.10 compares the PSNR performances of DCT_NLS, DCT_LBT(2×2), DCT_LBT(4×4) root block sizes with HLBT_DTT(4×4) on Lena, Barbara, Mandrill and Ruler images. It is observed that smooth image such as Lena shows (0.03-0.24) dB of PSNR loss; textured images such as Barbara and Mandrill shows (0.018-0.548)dB and (0.027-0.201)dB of PSNR loss respectively with respect to DCT_LBT(4×4). However, Ruler image shows significant PSNR gain (i.e. 9.16 dB) which occurs at 2.0 bpp in HLBT_DTT(4×4).

Table 5.11 compares the R-D performances of proposed algorithm with other state-of-the-art DCT coders presented by Hou *et al.*, Song and DCT_LBT. Comparison is also carried out with wavelet based WBTC and JPEG 2000, 2048 × 2560 size images (Bike and Woman) and 512 × 512 size test images (Lena and Barbara). These binary coded version [109] results are taken directly from [44].

It is clear that the proposed algorithm HLBT_DTT shows PSNR loss up to 0.31 dB compared to DCT_LBT algorithm for the considered set of images. The algorithm proposed by Song performs best in texture images than JPEG 2000. For example, it outperforms by 0.40-0.97 dB in Barbara image.
HLBT_DTT exhibits comparable result with Song and Cho in Lena image. While considering smooth image such as Woman, HLBT_DTT shows a PSNR gain of (0.09-1.16) dB below 0.5 bpp and a PSNR loss of 0.48-0.90 dB on 1.0-2.0 bpp. One possible reason is that the block-tree algorithm encodes fewer symbols than images of lower dimensions for some early passes because of large cluster of zero blocks in higher dimension images. This improves the PSNR at lower rates. As more coefficients becomes significant at later passes (lower thresholds), more and more bits are required to represent significance of block trees during partitioning. The bits required to represent block-trees do not have any contribution to PSNR. This could be the reason for overall decrease of PSNR at higher rates. Similar kind of performance improvement has been observed in Bike image. When compared with JPEG 2000, the proposed technique shows an overall PSNR decrement on most of the images. This decrement is compensated by the simplicity of the proposed algorithm compared to JPEG 2000. While comparing with wavelet based WBTC algorithm, HLBT_DTT outperforms for Barbara, Woman and Bike images at lower rates. A slight PSNR reduction is observed in case of Lena image.

Improvement at lower rates is a desirable feature for browsing images over wireless lines where a significant amount of information is required at the earlier stages of transmission. It is worth mentioning that HLBT_DTT coder shows a significant improvement of PSNR values in lower bit rates than most of the DCT based embedded coders except DCT_LBT.

5.6.4 Computational Complexity

The execution time determines the complexity of an algorithm. It is observed from the Table 5.12 that execution times of list based coders, e.g. SPIHT, increases exponentially with respect to coders without lists such as NLS and LBTC. During encoding, SPIHT shows 1.3-7.8 times faster below 0.5 bpp and 2-9 times slower above 0.5 bpp in Lena image. Similar kind of performance is also observed during decoding. In other words, the average encoding time in SPIHT is 29.36 sec and decoding time is 12.29 sec whereas, it is 10.26 sec and 9.18 sec respectively in NLS over the consid-

Table 5.11: Comparison of PSNR(dB) values between Hou *et al.*, JPEG 2000, Song and Cho, WBTC, DCT_LBT and HLBT_DTT algorithms on 2560 × 2056 size images (Bike and Woman) and 512 × 512 size images (Lena and Barbara).

Images	Algorithm	Bit rate (bpp)					
		0.0625	0.125	0.25	0.5	1.0	2.0
Bike	Hou *et al.*	22.18	24.47	27.35	30.73	34.77	39.91
	JPEG2000	23.80	26.36	29.62	33.51	38.10	43.98
	Song and Cho	23.24	25.75	28.60	32.08	36.57	42.57
	WBTC	23.14	25.48	28.76	32.51	37.12	43.13
	DCT_LBT(4 × 4)	23.65	26.32	29.52	32.65	36.86	42.72
	HLBT_DTT(4 × 4)	23.43	26.12	29.33	32.47	36.75	42.59
Woman	Hou *et al.*	24.90	26.69	28.63	31.70	35.98	41.34
	JPEG2000	25.59	27.35	29.99	33.62	38.42	43.99
	Song and Cho	25.60	27.29	29.84	33.38	37.89	43.77
	WBTC	25.67	27.34	30.05	33.73	38.30	44.10
	DCT_LBT(4 × 4)	26.85	28.62	30.87	33.68	37.52	43.01
	HLBT_DTT(4 × 4)	26.76	28.43	30.46	33.47	37.41	42.87
Lena	Hou *et al.*	-	29.42	32.88	36.37	39.68	-
	JPEG2000	27.80	30.79	33.97	37.24	40.34	-
	Song and Cho	27.57	30.28	33.36	36.64	39.93	-
	WBTC	28.52	31.27	34.17	37.28	40.43	45.07
	DCT_LBT(4 × 4)	27.49	30.17	33.41	36.72	40.08	44.82
	HLBT_DTT(4 × 4)	27.35	30.00	33.27	36.51	39.85	44.52
Barbara	Hou *et al.*	-	25.43	28.54	32.29	37.05	-
	JPEG2000	23.17	25.25	28.30	32.17	37.12	-
	Song and Cho	24.06	26.43	29.27	32.82	37.52	-
	WBTC	23.47	24.87	27.67	31.49	36.45	43.01
	DCT_LBT(4 × 4)	23.52	25.84	28.62	32.39	37.41	43.81
	HLBT_DTT(4 × 4)	23.43	25.75	28.48	32.20	37.26	43.52

All the results are with arithmetic Coding   a '-' indicates results are not available.

ered bit rates. The proposed LBTC algorithm is slight more complex than NLS. The reason is additional block partitioning steps where a significant block tree is recursively quad partitioned to find significant coefficients for each pass. A slight increase of encoding time and a slight decrease of decoding time is observed for larger root block sizes. The reason is that larger blocks consumes more time during encoding whereas, reconstructing a pixel from a larger set is a less time consuming operation because of efficient skipping of predictable insignificance sets/blocks. Similar kind of performance is noticed in other standard images.

Table 5.12: Comparison of encoding and decoding time (sec) for Lena image (Wavelet transform and DCT are not included).

Algorithm	Bit rate (bpp)					
	0.0625	0.125	0.25	0.5	1.0	2.0
	Encoding Time					
SPIHT	0.67	1.26	2.76	7.27	25.14	139.07
NLS	5.26	6.30	8.06	9.69	11.82	14.89
LBTC(2×2)	7.02	8.23	10.13	11.83	14.13	17.33
LBTC(4×4)	7.19	8.43	10.40	12.18	14.55	17.63
LBTC(8×8)	7.47	8.50	10.49	12.36	14.71	17.94
	Decoding Time					
SPIHT	0.15	0.45	1.47	5.74	21.55	44.35
NLS	4.97	5.75	7.20	8.19	9.39	11.22
LBTC(2×2)	6.71	7.73	9.25	10.43	11.79	13.67
LBTC(4×4)	6.54	7.46	8.98	10.07	11.44	13.29
LBTC(8×8)	6.30	7.06	8.49	9.56	10.87	12.80

5.7 Conclusions

In this chapter, a listless block tree coder (LBTC) which is a no list variant of WBTC algorithm is proposed. WBTC uses two dimensional block trees whereas, LBTC uses one dimensional block tree structures to exploit inter and intra scale correlation. This, enhance the low bit rate performance of most of the wavelet (i.e., NLS) and block based (i.e., DCT/DTT) embedded coders. The memory requirement of the proposed coder is fixed which is almost 22.3% more than the image alone. The proposed coders (LBTC, DCT_LBT, HLBT_DTT) manifest performance degradation at higher bit rates, typically ≥ 1.0 bpp. The reconstruction quality may not be seriously affected, as at higher rates, the subjective quality of an image remains indistinguishable to slight reduction of PSNR values. It is noted that the proposed coders are little complex than NLS, but much lesser than JPEG 2000 and embedded DCT coder proposed by Song because both use context adaptive arithmetic coding.

Therefore, the proposed coders are suitable for low memory hand held devices (e.g. digital camera, PDAs, mobile phones etc.) in particular and image communication in general, where a significant amount of information is to be transmitted within the available bandwidth.

Chapter 6

A Low Complexity and Efficient Sparse 8×8 Orthogonal Transform Matrix for Image Compression

Preview

This chapter presents an efficient orthogonal sparse 8×8 transform matrix for color image compression particularly at lower bit rate applications. The transform matrix implemented in Xilinx XC2VP30 FPGA device indicates that there is a significant saving in computation and hardware resources compared to DCT, Signed DCT (SDCT), approximate DCT and matrix proposed by Bouguezel *et al.* By using various natural test images, it is demonstrated that the subjective and objective qualities of the proposed matrix are comparable with the above transforms at lower bit rates. Further, it outperforms SDCT by a large margin almost at all bit rates for most of the images.

6.1 Introduction

For complexity constrained encoding situations, where even a fast fixed-complexity DCT algorithm is too complex, one can resort to approximate the computation of DCT at the cost of some degradation in the image quality. These applications can be multimedia, mobile communications, PDAs, digital cameras, etc where a lot of image transmission and processing are required.

Even though a number of algorithms for fast computation of DCT are available in the literature, there has been a lot of interest towards finding out the approximate integer versions of floating point DCT [66]-[71]. Two 8×8 versions of transformation matrices, one for the coarsest and another for the finest approximation levels (represented as $\hat{D}_1$ and $\hat{D}_5$ respectively) of exact DCT has been proposed by Lengwehasatit and Ortega [68]. Using these two matrices, a trade off between speed versus accuracy in various bit ranges can be achieved. The performance shows 73 % complexity reduction with only 0.2 *dB* PSNR degradation. A family of 8×8 biorthogonal transforms called binDCT, which are all approximates of popular 8×8 DCT has been proposed in [69]. A new kind of transform called signed DCT (SDCT) by applying signum function to DCT has been proposed in [70]. However, SDCT and its inverse are not orthogonal and it needs 24 additions for transformation. A 8×8 transform matrix is presented in [71] by appropriately inserting 20 zeros into the elements of $\hat{D}_1$ [68]. A reduction of 25 % in computation is achieved over SDCT and this matrix is orthogonal. Unlike the matrices proposed [72]-[74], the transform order need not be a specific integer or a power of 2.

Wavelet coding algorithms have several advantages over DCT based methods at the cost of computational complexity. These are state-of-the-art image compression algorithms such as EZW [15], SPIHT [16], EBCOT [27] have 0.5-1.5 *dB* PSNR gain over conventional DCT for a wide range of bit rates (0.1-1bpp), absence of blocking artifacts, progressive transmission capabilities and precise rate control. The proposed matrix is applicable to low power devices, where conventional DCT transform or wavelet coding algorithms are not suitable.

To implement a transform kernel using conventional approach requires a number of multipliers. Multipliers are the major source of power hungry elements in a hardware device. Distributed arithmetic (DA) computation is adopted to eliminate multipliers in the computations [75],[76]. This chapter presents a novel 8×8 orthogonal transform matrix. The proposed matrix is sparse and has 24 zeros entries. The matrix has been implemented in a Xilinx XC2VP30 FPGA device. Its resource utilization has been summarised. The application of the matrix to color image compression has been discussed. It has been shown that the proposed matrix provides 7 % reduction in computation over the matrix by Bouguezel *et al.* [71] and 45 % over SDCT [70].

6.2 Signed Discrete Cosine Transform

The two dimensional DCT of order $N \times N$ is defined as:

$$T_{DCT}(u,v) = \alpha_{uv} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} cos\frac{\pi(2i+1)u}{2N} cos\frac{\pi(2j+1)v}{2N}$$

where,

$$\alpha_{uv} = \begin{cases} \frac{1}{\sqrt{N}}, & \text{for u,v} = 0 \\ \frac{2}{\sqrt{N}}, & \text{otherwise.} \end{cases} \tag{6.1}$$

The signed discrete cosine transform (SDCT) is obtained by applying the signum function operator to the elements of DCT. Therefore, it is given as:

$$T_{SDCT}(u,v) = \frac{1}{\sqrt{N}} sign[T_{DCT}(u,v)] \tag{6.2}$$

where, $sign(.)$ is the signum function. It is defined as:

$$sign(.) = \begin{cases} +1, & \text{if x} > 0 \\ 0, & \text{if x} = 0 \\ -1, & \text{if x} < 0. \end{cases} \tag{6.3}$$

Several advantages of SDCT are apparent form Eqnuations 6.1, 6.2 and 6.3. These are:

- All the elements are ±1.
- No multiplication operation or transcendental expressions are required.
- Unlike WHT [72], RSWT [73] and SDFT [74], the transform order need not be a specific integer or a power of 2.
- SDCT maintains the periodicity and spectral structure of the original DCT and maintains good de-correlation and energy compaction characteristics.

It has been verified that only 10 percent spectral components of SDCT contains 80 percent of the total signal power compared to 87 percent signal power in DCT [70]. The 8 × 8 SDCT transform matrix is given by:

$$T_{SDCT} = \frac{1}{\sqrt{8}} \begin{bmatrix} +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 \\ +1 & +1 & +1 & +1 & -1 & -1 & -1 & -1 \\ +1 & +1 & -1 & -1 & -1 & -1 & +1 & +1 \\ +1 & -1 & -1 & -1 & +1 & +1 & +1 & -1 \\ +1 & -1 & -1 & +1 & +1 & -1 & -1 & +1 \\ +1 & -1 & +1 & +1 & -1 & -1 & +1 & -1 \\ +1 & -1 & +1 & -1 & -1 & +1 & -1 & +1 \\ +1 & -1 & +1 & -1 & +1 & -1 & +1 & -1 \end{bmatrix} \tag{6.4}$$

It is worth mentioning that the forward transform of $T_{SDCT}(u,v)$ is almost orthogonal. But, its inverse transform is not orthogonal. Hence, this matrix is only used for some applications like adaptive filtering where forward transform is employed. For applications like image compression, another reverse transform matrix called T_{SDCT}^{r} [66] has been presented where all elements are ±1, ±2, or 0. Like T_{SDCT}, it does not need any multiplication operation for reverse transformation except shift and sign bit changes. Moreover, 25% of the elements are zeros. Unfortunately, this special feature of the matrix is not valid for all orders of N.

6.3 Proposed 8×8 Transform Matrix

The proposed 8×8 transform matrix can be obtained by appropriately inserting some 0s and 0.5s into the SDCT matrix in Eqn. 6.4. The proposed matrix contains 24 zeros in comparison with 20 zeros with the matrix by Bouguezel *et al.* [71]. Multiplication of input pixel with 0.5 is just a shift and addition operation. So, multipliers are not needed during transform stage, which makes the transformation faster. The proposed matrix is shown in Eqn. 6.5. Where D is a diagonal matrix. It is expressed as $D{=}diag(1,\sqrt{2},2\sqrt{\frac{2}{5}},2,1,\sqrt{2},2\sqrt{\frac{2}{5}},2)$.

$$T=\frac{D}{2\sqrt{2}}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1\\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1\\ 1 & 0.5 & -0.5 & -1 & -1 & -0.5 & 0.5 & 1\\ 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0\\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1\\ 1 & -1 & 0 & 0 & 0 & 0 & 1 & -1\\ 0.5 & 0 & 0 & -0.5 & -0.5 & 0 & 0 & 0.5\\ 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 \end{bmatrix} \tag{6.5}$$

The above transform matrix can be represented in Eqn. 6.6 as:

$$T=\hat{D}\times\hat{T} \tag{6.6}$$

where, $\hat{D}=\frac{D}{2\sqrt{2}}$ and

$$\hat{T}=\begin{bmatrix} +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1\\ +1 & +1 & 0 & 0 & 0 & 0 & -1 & -1\\ +1 & +0.5 & -0.5 & -1 & -1 & -0.5 & +0.5 & +1\\ 0 & 0 & -1 & 0 & 0 & +1 & 0 & 0\\ +1 & -1 & -1 & +1 & +1 & -1 & -1 & +1\\ +1 & -1 & 0 & 0 & 0 & 0 & +1 & -1\\ +0.5 & 0 & 0 & -0.5 & -0.5 & 0 & 0 & +0.5\\ 0 & 0 & 0 & -1 & +1 & 0 & 0 & 0 \end{bmatrix} \tag{6.7}$$

It can be seen that the matrix $\hat{T}$ satisfies the property of orthogonality, i.e, $T^{-1} = T^t$ where t denotes the transpose operation. Therefore, the same matrix can be used for image encoding and image decoding. Let, X be an 8×8 block of image data and Y be its corresponding matrix in transformed domain. Then, the forward transform operation is:

$$Y = TXT^t \tag{6.8}$$

Since T is orthogonal, the image can be reconstructed using inverse transform given as:

$$X = T^tYT = \hat{T}^t(\hat{D}^tY\hat{D})\hat{T} \tag{6.9}$$

The proposed matrix in Eqn. 6.5 has been used in R, G and B color planes to achieve compression of color images.

6.4 Distributed Arithmetic based Algorithm for Fast Computation

The distributed arithmetic equation is expressed as:

$$Y = \sum_{k=0}^{K} A_k x_k \tag{6.10}$$

where A_k are the fixed coefficients and x_k are the input data words. If x_k is a 2's complement binary number scaled such that $\mid x_k \mid \leq 1$, then each x_k can be expressed as:

$$x_k = -b_{k0} + \sum_{n=1}^{N-1} b_{kn} 2^{-n} \tag{6.11}$$

where b_{kn} are binary digits, b_{k0} is the sign bit and $b_{k(N-1)}$ is the least significant bit (LSB). Combining Eqn. 6.10 and Eqn. 6.11, Y can be expressed in terms of bits of x_k as:

$$Y = \sum_{k=1}^{K} A_k[-b_{k0} + \sum_{n=1}^{N-1} b_{kn} 2^{-n}] \tag{6.12}$$

Taking transformation of the proposed integer matrix of Eqn. 6.7, the following eight equations can be obtained.

$$
\begin{aligned}
&Y(0) = [x(0)+x(7)] + [x(1)+x(6)] + [x(2)+x(5)] + [x(3)+x(4)],\\
&Y(1) = [x(0)-x(7)] + [x(1)-x(6)],\\
&Y(2) = [x(0)+x(7)] + (0.5)[x(1)+x(6)] + (-0.5)[x(2)+x(5)] + (-1)[x(3)+x(4)],\\
&Y(3) = [x(5)-x(2)],\\
&Y(4) = [x(0)+x(7)] + (-1)[x(1)+x(6)] + (-1)[x(2)+x(5)] + [x(3)+x(4)],\\
&Y(5) = [x(0)-x(7)] + (-1)[x(1)-x(6)],\\
&Y(6) = (0.5)[x(0)+x(7)] + (-0.5)[x(3)+x(4)],\\
&Y(7) = [x(4)-x(3)].
\end{aligned}
\tag{6.13}
$$

The distributed algorithm in Eqn. 6.12 can be used to compute the above eight equations. Taking $Y(2)$ as an example of Eqn. 6.13 to discuss the application of distributed arithmetic.

$$
Y(2) = \begin{bmatrix} -2^1 & 2^0 & 2^{-1} \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} x(0)+x(7) \\ x(1)+x(6) \\ x(2)+x(5) \\ x(3)+x(4) \end{bmatrix} \tag{6.14}
$$

Assigning,

$$
\begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} \times \begin{bmatrix} x(0)+x(7) \\ x(1)+x(6) \\ x(2)+x(5) \\ x(3)+x(4) \end{bmatrix} = \begin{bmatrix} Y^{-1}(2) \\ Y^{0}(2) \\ Y^{1}(2) \end{bmatrix} \tag{6.15}
$$

$$
Y(2) = [\text{-}2^1 \times Y^1(2)]+[2^0 \times Y^0(2)]+[2^{-1} \times Y^{-1}(2)] \tag{6.16}
$$

Therefore, to compute the value of $Y(2)$, values of $Y^1(2)$, $Y^{-1}(2)$ need to be shifted and added with $Y^0(2)$. Meanwhile, $Y(0)$, $Y(1)$, $Y(3)$, $Y(4)$, $Y(5)$, $Y(6)$ and $Y(7)$ values can be computed in a similar manner. The shift operation is implemented by wirings, which has negligible delay and hardware resources.

Table 6.1 is the explanation the algorithm of the proposed transform matrix. For instance, ALU1, ALU2, ALU3, and ALU4 performs addition operation when $Y(0)$ is calculated. The signal flow graph of 1D 8-point transform matrix is shown in Figure 6.1. According to Figure 6.1, many adders and subtracters which can be implemented by a simple ALU can be shared. Therefore, the requirement of hardware resources can be further reduced. The values of $Y^1(0)$ and $Y^{-1}(0)$ are zero. $Y^0(0)$ can be obtained from the signal $R0$.

Table 6.1: Explanation of the Hardware structure of Figure 6.1. (symbols '+' means positive, '-' means negative and 'No' means no operation)

H/W	Y(0)	Y(1)	Y(2)	Y(3)	Y(4)	Y(5)	Y(6)	Y(7)
ALU1	+	-	+	No	+	-	+	No
ALU2	+	-	+	No	+	-	No	No
ALU3	+	No	+	-	+	No	No	No
ALU4	+	No	+	No	+	No	+	-
Y^1	0	0	R2	0	R4	R6	R7	0
Y^0	R0	R1	R3	R5	R0	R1	R7	R7
Y^{-1}	0	0	R4	0	0	0	R8	0

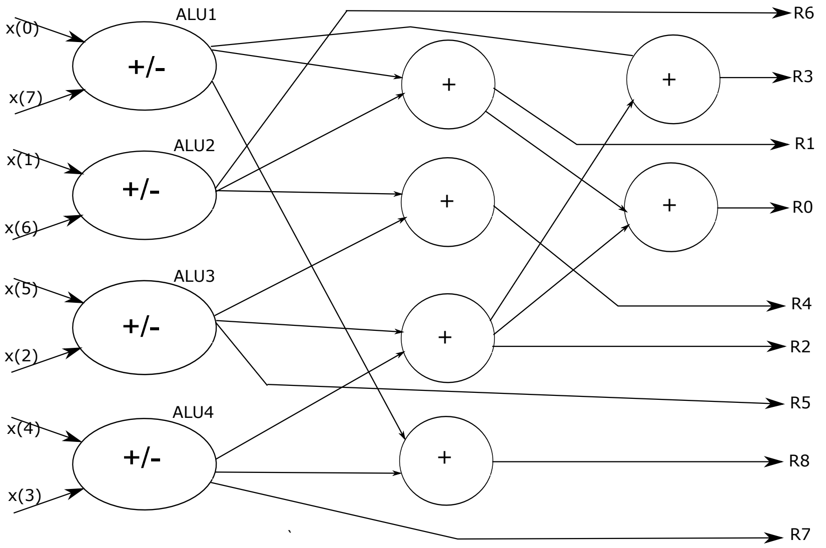


Figure 6.1: Hardware structure of the proposed matrix

6.5 Application to JPEG Color Image Compression

The proposed transform matrix is applied in a standard JPEG baseline encoder as shown in Figure 6.2. Since the quantization operation is applied after transformation using proposed matrix, the diagonal term of the matrix of Eqn. 6.5 can be merged into the quantizer. This is perform so that $\hat{T}$ in Eqn. 6.7 is the only source of computation in the transformation stage.

6.5.1 Chroma Subsampling and Color Space Conversion

It is a well-known fact that there is a high correlation between RGB color plane of an original image. The color conversion is done so as to ensure complete decorrelation among the color planes in order to achieve high compression. YCbCr color space has good decorrelation properties. After the color space conversion, most of the spatial information of the image is contained in the luma (Y) component. The

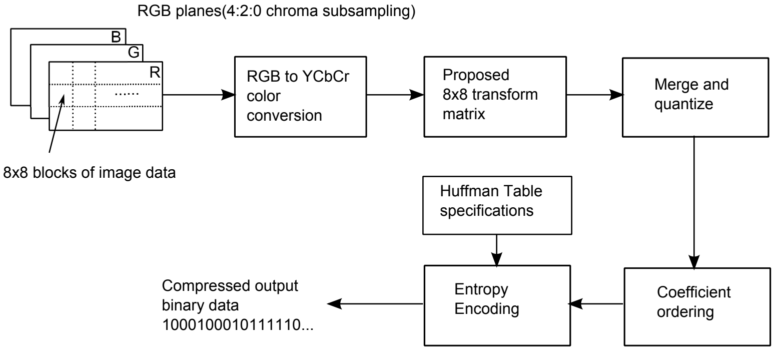

RGB planes(4:2:0 chroma subsampling)
B
G
R
8x8 blocks of image data
RGB to YCbCr color conversion
Proposed 8x8 transform matrix
Merge and quantize
Huffman Table specifications
Entropy Encoding
Coefficient ordering
Compressed output binary data 1000100010111110...

Figure 6.2: Block diagram of proposed transform matrix on the 8×8 blocks of image data.

chroma components (Cb and Cr) contain mostly redundant color information and little information is being lost by subsampling these components both horizontally and/or vertically. In the proposed system both Chroma planes are down sampled by a factor of 2, both horizontally and vertically with respect to Luma plane. This kind of subsampling called 4:2:0 chroma subsampling. Each color component is divided into 8×8 non overlapping blocks and by selecting one or more data blocks from each of the color components, a minimum coded unit (MCU) is constituted. MCU defines the arrangement of the data blocks in interleaved scanning order of the color components. In interleaved order, data blocks from all the color components appear in each MCU. There can be a provision for a non-interleaved scan where each color components are stored and processed separately. In 4:2:0 format, each MCU consists of a 2×2 units of four data blocks form the Y component and one from each of the Cb as well as Cr components.

6.5.2 Merge and Quantize

The fractional term of Eqn. 6.5, i.e.$\frac{D}{2\sqrt{2}}$ is merged with the quantization matrix so as to ensure fast compression and decompression. The new quantization matrix of luminance component is shown in Table 6.2. In a similar manner, quantization matrix for the chrominance components can be obtained. The matrices of both luminance and chrominance components are also being used for de-quantization operation during decoding stage.

6.6 FPGA Synthesis Results of the Proposed Transform Matrix

The proposed transform matrix has been implemented on Xilinx XC2VP30 FPGA device. Table 6.3 and 6.4 shows the hardware utilization summery of the transform

Table 6.2: Proposed Luminance 8×8 Quantization Matrix

6	4	4	6	8	14	18	22
6	6	7	10	13	29	30	20
6	6	7	11	18	25	31	25
10	12	16	21	36	62	57	44
6	8	13	20	24	39	36	27
12	18	28	32	41	52	57	46
22	29	35	39	46	54	54	45
51	65	67	69	79	71	73	70

matrix of Equ. 6.7. It is clearly evident that by using the matrix of Equ. 6.7, a lot of FPGA resources can be saved. For instance, the hardware utilization (No. of Slices and 4 input LUTs) of 1D 8×8 transform matrix $\hat{T}$ of Equ. 6.7 is 3.9 % lesser than 1D 8×8 transform matrix T of Equ. 6.5. The basis of the proposed transform is integer powers of 2. Therefore, the hardware utilization indicated in Table 6.3 and 6.4 without diagonal term $\frac{D}{2\sqrt{(2)}}$ are far less than using diagonal term $\frac{D}{2\sqrt{(2)}}$.

Table 6.5 shows the adder cost saving of the proposed transform matrix over DCT, NEDA [77], Chungan et al.[76], Approx. DCT [68], SDCT and Matrix by Bouguezel *et al.*. It is clearly evident that the proposed transform matrix shows 97 % saving of ALU and adders compared to direct DCT. Further, the adder bit-width is 333, which is lowest among the other transforms. The adder bit-width is calculated by taking into account the total number of signal lines required to generate outputs $Y(0), Y(1), Y(2), ...Y(7)$ from inputs $x(0), x(1), ...x(7)$.

Table 6.3: H/W Utilization of 1-D transform of proposed matrix in XC2VP30.

		without using $\frac{D}{2\sqrt{2}}$		with using $\frac{D}{2\sqrt{2}}$	
Resources	Available	Utilise	% Utilisation	Utilize	% Utilisation
No. of slices	13696	164	1.1	613	4
Flip Flops	27392	0	0	0	0
4 input LUTs	27392	299	1.1	1099	4
Bonded IOBs	556	168	30	152	27

Table 6.4: H/W Utilization of 2-D transform of proposed matrix in XC2VP30.

		without using $\frac{D}{2\sqrt{2}}$		with using $\frac{D}{2\sqrt{2}}$	
Resources	Available	Utilise	% Utilisation	Utilize	% Utilisation
No. of slices	13696	295	2	1582	11
Flip Flops	27392	190	0.7	664	2
4 input LUTs	27392	496	1.8	2749	10
Bonded IOBs	556	147	26	115	20

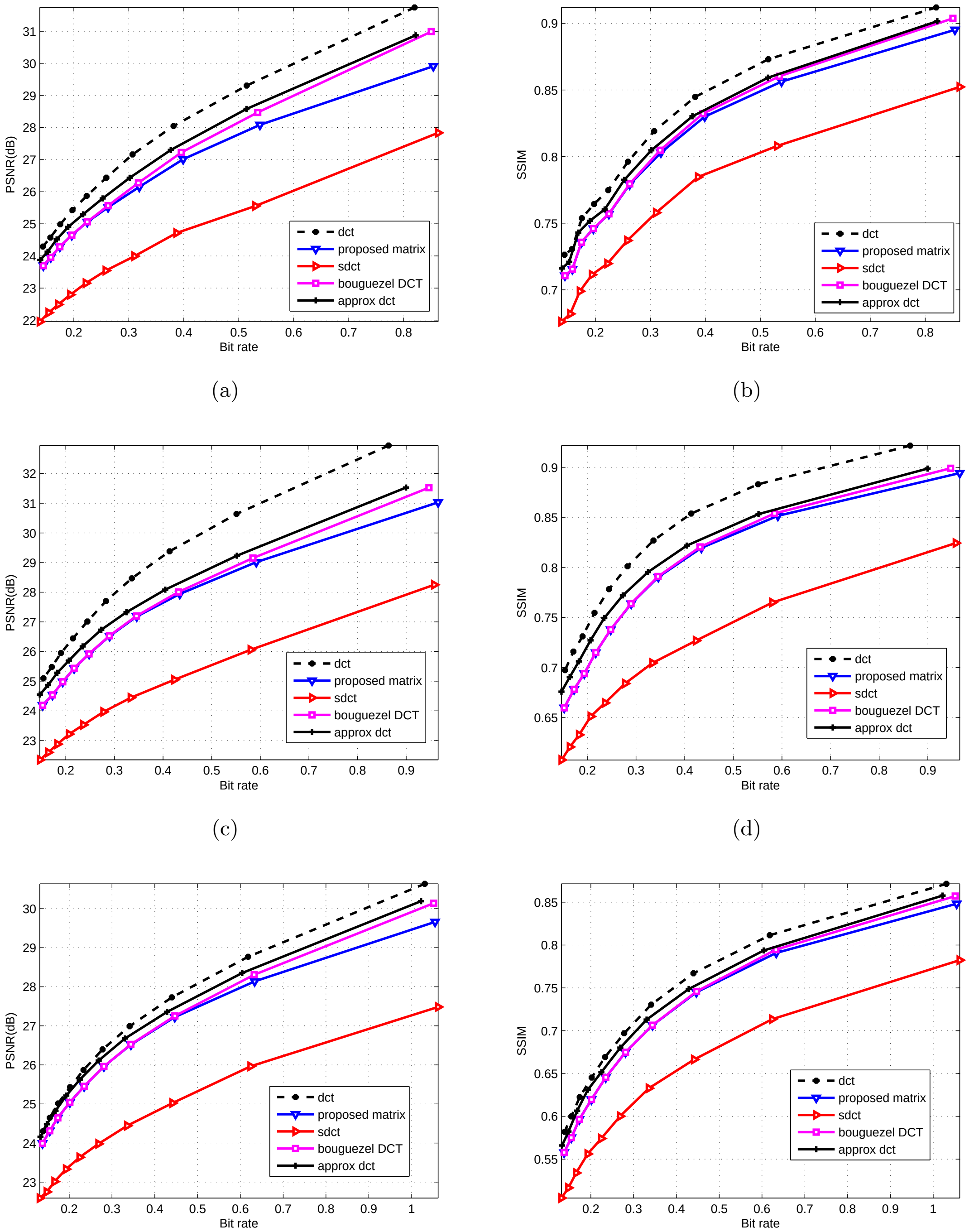
Table 6.5: Comparison of Adder Cost Savings

Scheme	*Adder matrix*	*Adder bit-width*	*% saving*
Direct DCT	308	2496	-
NEDA	35	1800	88
Chungan *et al.*	9 ALUs+6 adders	850	95
Approx. DCT	4 ALUs+16 adders	533	94
Bouguezel *et al.*	4 ALUs+8 adders	426	95
SDCT	4 ALUs+7 adders	554	96
Proposed	4 ALUs+6 adders	333	97

6.7 Matlab Simulation Results and Discussion

The superiority of the proposed technique is demonstrated through computer simulation running on Microsoft Window XP, Intel Core2 Duo CPU, 3 GHz Platform. PSNR and MSSIM are used for comparison. Experiments are conducted on 3 monochrome images, i.e., Cameraman, Lena, Goldhill and 2 color images, i.e., pepper and f16. All the images are of 512×512.

While comparing between the proposed transform matrix and Approx. DCT, it is observed from Figures 6.3 (a),(c) and (e) that there is a PSNR reduction of approx. 0.2-1.0 dB for Cameraman image, approx. 0.15-0.8 dB for Lena image and approx. 0.2-0.8 dB for Goldhill image. The MSSIM performance of the proposed matrix closely follows that of Approx. DCT and Matrix by Bouguezel *et al.* for all images. Figure 6.4(b)-6.4(f) shows the reconstructed images of Cameraman at a scale factor of 5 among all the five transforms. It is observed that the MSSIM performance of Approx. DCT is only 0.34% higher than the proposed transform matrix, whereas DCT outperforms by approx. 5% than the proposed matrix in most of the images. However, the proposed transform matrix shows improvement in PSNR by a wide margin (about 2.0 dB) and MSSIM performance by 5-10% than SDCT. The PSNR vs. bit rate and MSSIM vs. bit rate plots of Pepper image are shown in Figures 6.5 (a) and (b)respectively. Similarly, PSNR vs. bit rate and MSSIM vs. bit rate plots of f16 image is shown in Figure 6.5 (c) and (d) respectively. These figures illustrates a comparison between proposed 8×8 transform matrix, DCT, Transform Matrix by Bouguezel *et al.*, approx. DCT, and SDCT. It can be seen from these figures that PSNR of proposed transform is almost 1.5-2.5 dB higher than SDCT for the considered set of bit rates. It follows closely with the transform matrix by Bouguezel *et al.*. When compared with Approx. DCT, the proposed transform shows a maximum of 0.01-0.4 MSSIM reduction between 0.1 to 0.8 bit rates on Pepper image. Although DCT outperforms all the transforms both in terms of PSNR and



(a) (b)

(c) (d)

(e) (f)

Figure 6.3: Rate-distortion plots of (a) Cameraman, (c) Lena, (d) Goldhill images. MSSIM vs. bit rate plots of (b) Cameraman, (d) Lena, (f) Goldhill images.

MSSIM, it is computationally expensive than other transforms. In Figure 6.5(c) of f16 image, there is a sudden decrease of PSNR value at bit rates between 0.3-0.45 bpp. Below 0.3 and above 0.45 bpp, it is around (1-1.5) dB lower than approx. DCT. It is interesting to note that same trend does not appear in its MSSIM performance which is shown in Figure 6.5(d).

Figure 6.6 shows the reconstructed images of f16 using DCT, Matrix proposed by Bouguezel *et al.*, approx. DCT, SDCT and the proposed transform matrix at a scale factor of 5. The scale factor is an integer variable term which is multiplied with the quantization matrices during encoding/decoding in order to control bit rates. It is clear that the reconstructed images of SDCT algorithm in Figure 6.6(e) has MSSIM values of 0.7971, which is lowest among all algorithms. Reconstructed images using proposed transform matrix T of Eqn. 6.7 has MSSIM value 0.8283. This is almost same as of Figure 6.6(d). MSSIM values of Approx. DCT values are little higher ($\sim$ 1%) than the proposed transform matrix. This indicates that the quality of the decoded images are visually indistinguishable.

6.8 Conclusion

An orthogonal sparse transform matrix for image compression application has been proposed in this chapter. A fast algorithm for computation using DA is developed and its FPGA implementations are performed. The proposed 8×8 transform matrix needs 4 ALUs+6 addition operations compared to 4 ALUs+7 adders by SDCT, 4 ALUs+8 adders by Bouguezel *et al.*, 4 ALUs+16 adders by Approx. DCT. The basis of the proposed sparse matrix is depends on integer powers of 2. By merging the diagonal matrix D with quantization process, the forward and reverse transformations are made faster than other transforms. FPGA synthesis result shows that the proposed matrix needs less number of hardware resources. Therefore, it is suitable for low power and complexity constrained hardware platforms.

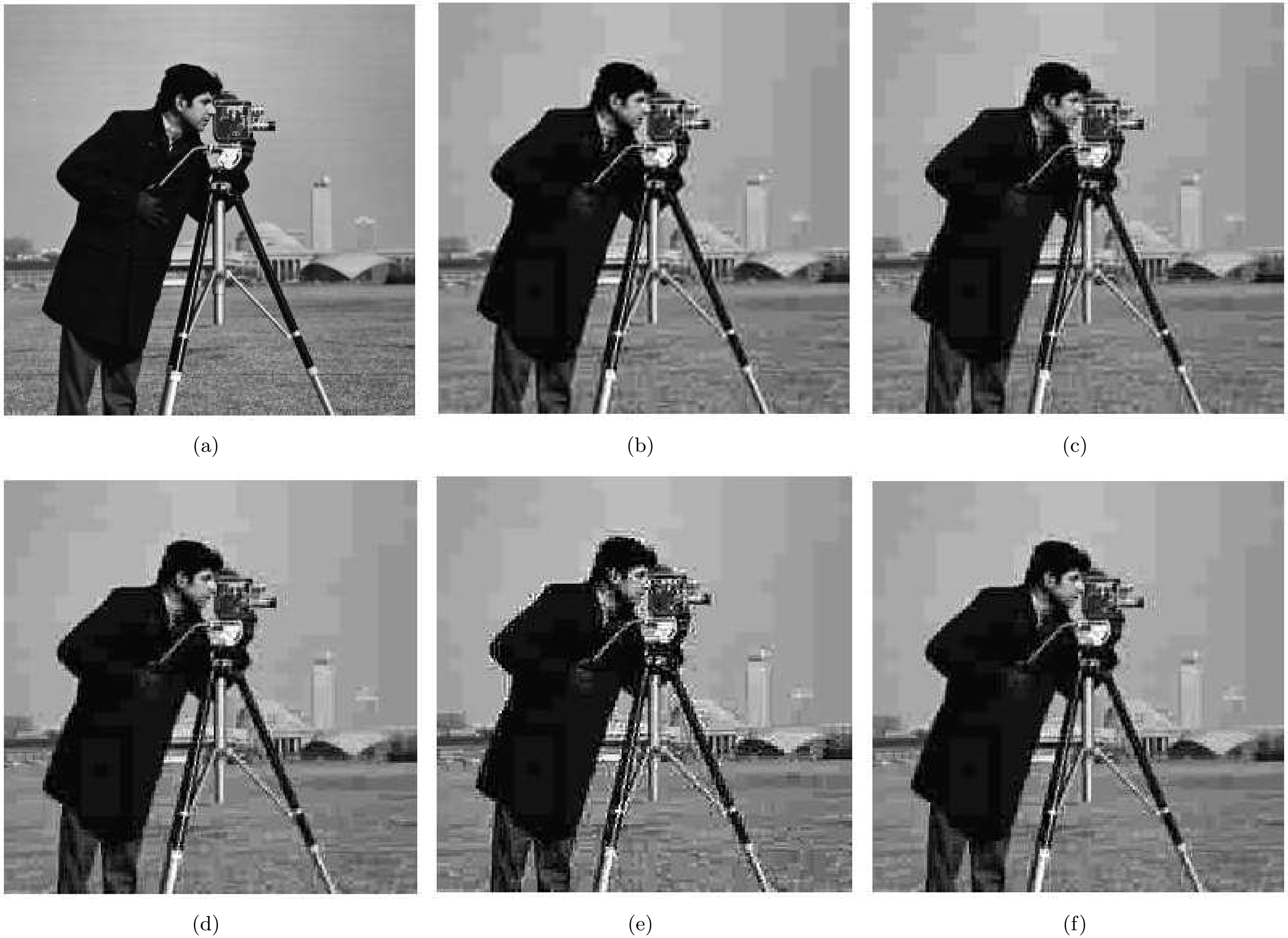
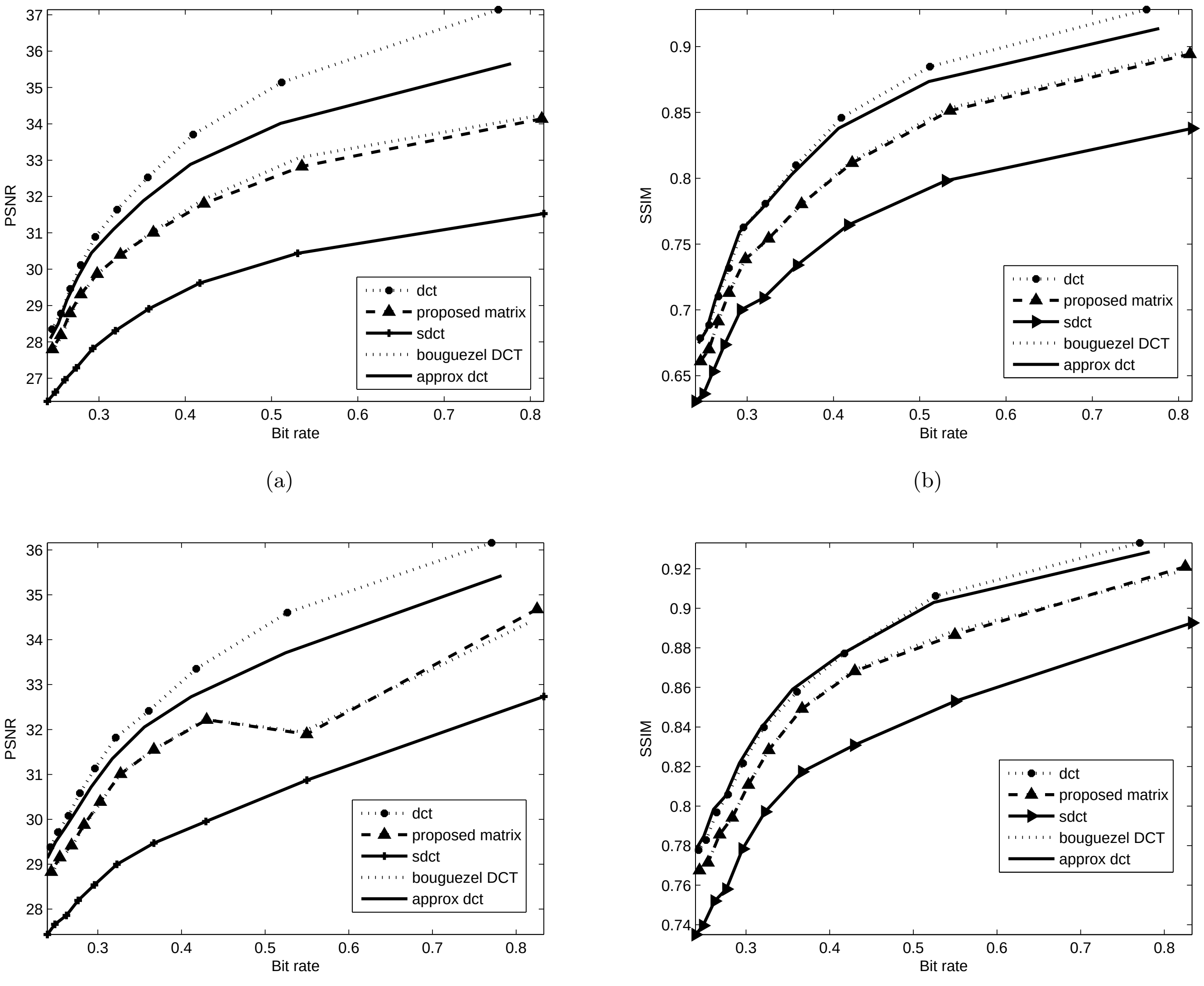


Figure 6.4: (a) Original Cameraman image reconstructed at a scale factor of 5 using (b)DCT [MSSIM=0.7961, CR=30.86] (c) Approx. DCT [MSSIM=0.7825, CR=31.68] (d) Bouguezel *et al.* [MSSIM=0.7795, CR=30.48] (e) SDCT [MSSIM=0.7370, CR=30.98] (f) Proposed matrix [MSSIM=0.7791, CR=30.48]



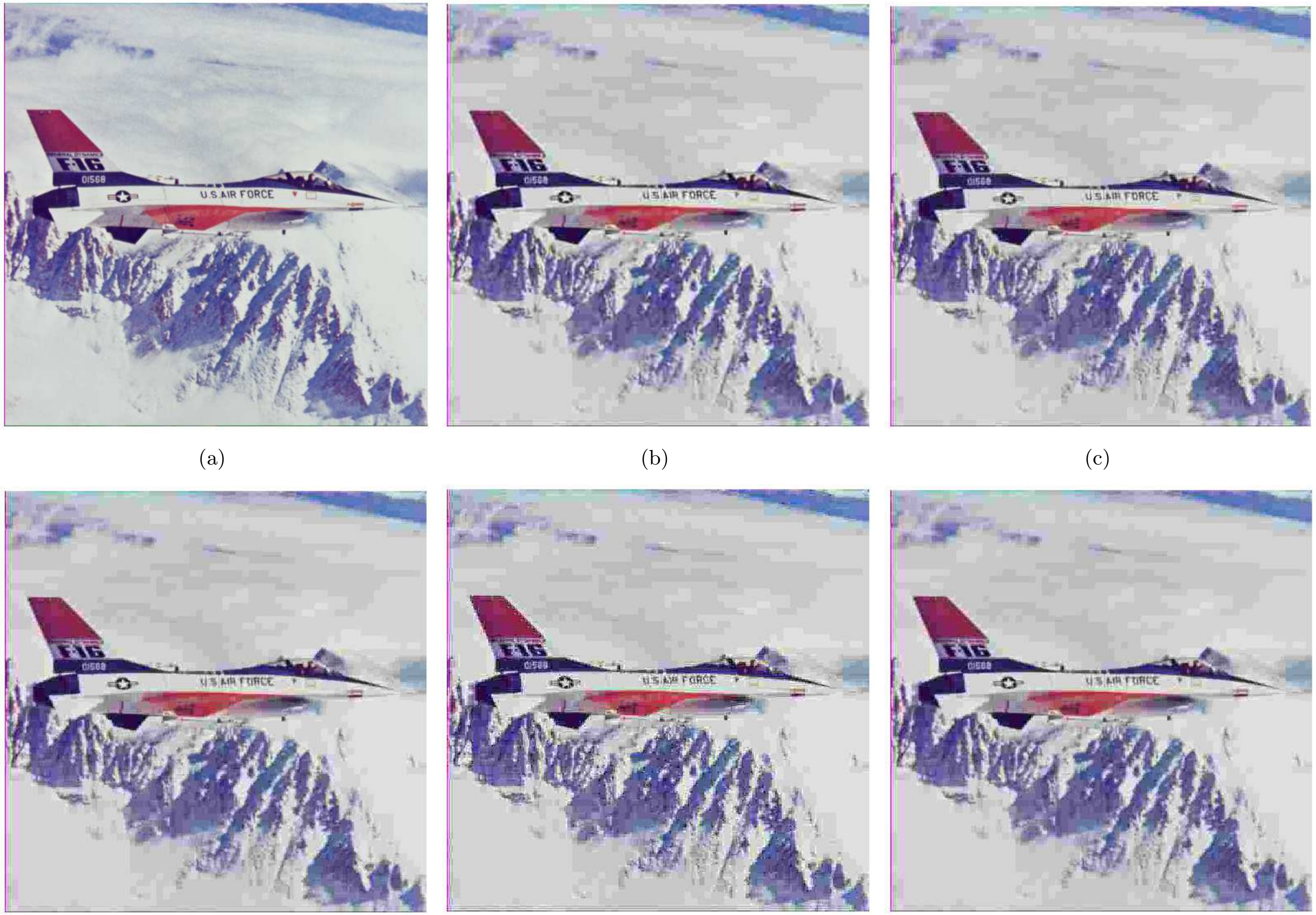
(a)

(b)

(c)

(d)

Figure 6.5: Rate-distortion and MSSIM vs. bit rate plots of (a) and (b) Pepper, (c) and (d) f16 images respectively.



(a) (b) (c)

(d) (e) (f)

Figure 6.6: (a) Original f16 images reconstructed at a scale factor of 5 using (b)DCT [MSSIM=0.8398, CR=74.67] (c) Approx. DCT [MSSIM=0.8391, CR=75.63] (d) Bouguezel *et al.* [MSSIM=0.8284, CR=73.37] (e) SDCT [MSSIM=0.7971, CR=74.36] (f) Proposed matrix [MSSIM=0.8283, CR=73.35]

Chapter 7

Conclusions

In this chapter, the overall conclusions are presented and contributions are summarized. Further research directions in the same or related topics are outlined. The algorithms in this thesis have been developed with a goal to improve the rate-distortion performance particularly at lower bit rates, while reducing the memory requirements. A good trade-off between the above parameters has been achieved.

7.1 Summary of Work Done

A thorough experimental analysis of DTT is presented on JPEG and embedded codecs. A distributed arithmetic based fast zigzag pruning DTT algorithm has been developed, which can be integrated on a conventional JPEG baseline codec using minor modifications. The image reconstruction qualities are also compared with similar number of pruned DCT coefficients. The comparisons show satisfactory PSNR results with respect to DCT pruned coefficients. In the embedded encoder, DTT is reapplied to the coarsest subband coefficients in order to improve the rate-distortion performance at lower rates. Different perceptual weights are applied to different subband coefficients in the DTT_SPIHT embedded coder to improve the MSSIM performance.

A Hierarchical listless DTT (HLDTT) algorithm has been developed which is based on the recursive block partitioning procedure similar to SPECK, SBHP and LSK. The difference of these algorithms with respect to HLDTT is that it further exploits the fundamental principle of coefficient decaying spectrum of transformed images to encode less number of symbols to many hierarchical subbands. This improves the low bit rate performance compared to DCT based embedded coders. The dynamic memory requirement is also reduced by a significant amount for earlier five to six bit plane passes compared to LSK and SPIHT in most of the monochrome images. The proposed coder can be best suited for browsing better quality images in a narrow bandwidth channel, downloading or reconstructing images in a system with limited memory buffers. It also has been shown that the region of interest feature can be incorporated in a decoded image using HLDTT. This can be useful for medical image compression where the desired part can be encoded with higher fidelity compared to the rest of the image.

The image quality at the earlier stages of image communication can be improved without sacrificing much on the bandwidth using wavelet transform. Further, the proposed LEBP algorithm shows improved progressive transmission and scalability efficiency over SPIHT, and on par with LSK. Random access decodability with progressive-in-resolution feature has also been incorporated which is not present in LSK. This is particularly useful for interactive multimedia applications, manipulation of certain areas of an image such as cropping, flipping, rotation, translation, scaling, feature extraction, etc. Another novel algorithm has been developed for color images (i.e., Color listless embedded block partitioning (CLEBP)), which extends the concept of LEBP for compression of color images. The algorithm is highly dependant upon the image sizes. For example, it exhibit excellent performance in case of equal dimension images. For unequal dimension images, appropriate padding is used before compression. From simulation, it is verified that symmetrical padding across the border gives rise to better PSNR performance than other kind of paddings. While comparing rate-distortion performance, CLEBP shows a significant improvement of rate-distortion performance with respect to state-of-the-art CSPIHT and other algorithms on wide range of bit rates.

Most of the above algorithms explicitly perform breadth first search. Another low complexity, yet efficient algorithm i.e., Listless block tree coding (LBTC) has been proposed, which performs explicitly depth first search to exploit both inter and intra subband correlation. It exploit clusters of zero-trees without using list arrays to improve the low bit rates performance in contrast to WBTC. LBTC exhibits similar rate distortion performance as that of WBTC, but with 88-89% memory saving compared to WBTC and SPIHT. Two novel listless algorithms are proposed e.g., DCT_LBTC and HLBT_DTT, which combine block based transforms such as DCT/DTT with LBTC. By compromising the trade off betwen complexity and image quality, any one of the algorithms can be selected. While the comparison is made between DCT_LBTC and HLBT_DTT, the former outperforms the later in most of the bit rates for natural images. For unnatural images such as computer graphics or adjcent video frames having higher intensity gradation (inter frame compression), HLBT_DTT is more suitable than DCT_LBT.

Finally, a low complexity orthogonal transform matrix is developed which is particularly useful for applications like digital camera and camcorders. In such applications, the images always need to have a fixed pre-defined image quality. The extra effort required for quality scalability is wasted. It is possible to encode specific number of bit-planes at a time instead of bit-plane by bit-plane in all the proposed algorithms without compromising image quality. A sparse orthogonal transform matrix which can be best suited for the considered applications has been proposed. The proposed matrix shows 97% saving in adder cost compared to direct DCT implementation. Therefore, the proposed matrix exihibit a significant saving in hardware complexity with a marginal loss of image quality at lower rates.

7.2 Contributions

The contributions in this thesis are summarized as follows:

- A fast zigzag pruned multiplier less DTT algorithm has been proposed. The proposed algorithm can be integrated into a standard JPEG image/video codec to speed up the processing. The 3×3 zigzag pruned DTT coefficients provide lower complexity and better PSNR than 3×3 block pruned DTT coefficients.

- A novel Hierarchical listless DTT (HLDTT) algorithm has been proposed, which further exploits the fundamental principle of coefficient decaying spectrum using breadth first embedded scanning techniques. Reapplying DTT to the DTT transformed coefficients in the coarsest subband, a further improved low bit rate performance in HLDTT is achieved compared to DCT based embedded coders. Since, DTT polynomials have certain properties that matches with Human visual systems (HVS), an improved MSSIM performance of the HLDTT decoded images has also been obtained over all the bit rates.

- The idea of HLDTT is extended to wavelet. Two reduced memory wavelet based compression algorithms, i.e., LEBP and CLEBP have been introduced. LEBP generates a feature rich bit stream, which has some additional desirable attributes like random access decodability with progressive-in-resolution scalability. CLEBP improves a significant low bit rate performance compared to CSPIHT and other embedded coders dealing with color images. The PSNR of CLEBP has been improved by a reasonable amount in higher bit rates.

- A listless block-tree partitioning (LBTC) algorithm has been introduced. The proposed algorithm uses similar idea used for WBTC, i.e., coding a cluster of zero-trees with few symbols, but the implementation uses flag maps (markers) in contrast to list data structures in WBTC. This substantially reduces memory requirements almost by 88-89% and speeds up encoding as well as decoding times by a wide margin.

- An orthogonal sparse transform matrix has been proposed. The proposed matrix reduces hardware complexity by 97% compared to DCT with a marginal loss of image quality at lower bit rates. This is especially applicable for complexity constrained hardware platforms, e.g., digital camera, camcorders etc.

7.3 Future Work

The research work in this thesis can be further extended in the following ways:

- By suitably selecting a proper quantization matrix and adaptive scan pattern, the performance of DTT on a JPEG baseline coder can be improved for a wide variety of images.
- Many fast algorithms for 8×8 DCT have been reported in the literature. Therefore, it is required to develop fast 8×8 algorithms for DTT for real time image/video applications in order to compete with DCT.
- Using adaptive HVS and modified SPIHT, the performance of DTT_SPIHT can be improved by a large margin. Further, sophisticated post processing can be used to enhance the subjective quality of the images.
- The performance of HLDTT can be improved further by the use of context modeling at a cost of complexity. The feasibility of hardware implementation can be investigated to work with HD images/video.
- The concept of LEBP/CLEBP can be extended for 3D (volumetric) images and video coding applications. Error resilient capability and security aspect features can be incorporated to meet the high end multimedia applications.
- LBTC algorithm can be extended for color image/video coding by exploiting the inter subband correlations among different color planes. A rich set of features can be incorporated into the algorithm to compete with JPEG2000 standard in certain applications.
- An experimental analysis of non-embedded LEBP, CLEBP and LBTC algorithms can be made on JPEG2000 Verification Model(VM) in order to better meet the portable multimedia applications (e.g. Digital camera, Camcorders, etc). An attempt towards efficient VLSI implementation can also be made.

Bibliography

[1] R. C. Gonzalez and R. E. Woods, *Digital Image Processing, 3rd Ed.* Pearson Prentice Hall, 2008.

[2] S. Saha, "Image compression - from DCT to wavelets : A review." *ACM Crossroads*, vol. 6, no. 3, 2000. [Online]. Available: http://dblp.uni-trier.de/db/journals/crossroads/crossroads6.html#Saha00

[3] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Trans. on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.

[4] W. B. Pennebakr and J. L. Mitchell, *JPEG Still Image Compression Standard.* Chapman Hall, New York, 1993.

[5] N. Ahmed, T. Natarajan, and K. R. Rao, "Discrete cosine transfom," *IEEE Trans. Computers*, vol. 23, no. 1, pp. 90–93, Jan. 1974.

[6] T. Acharya and P.-S. Tsai, *JPEG2000 standard for image compression: concepts, algorithms and VLSI architectures.* John Wiley and Sons, 2005.

[7] *Information Technology–Digital Compression and Coding of Continuous-Tone Still Images*, ISO/IEC 10918 (JPEG) Std.

[8] L. Lohscheller, "A subjectively adapted image communication system," *IEEE Trans. on Communications*, vol. 32, no. 12, pp. 1316–1322, Dec. 1984.

[9] S. G. Mallat, "A theory for multiresolution signal decomposition: The wavelet representation," *IEEE Trans. Pattern Anal. and Mach. Intelligence*, vol. 11, no. 7, pp. 674–693, Jul. 1989. [Online]. Available: http://dx.doi.org/10.1109/34.192463

[10] I. Daubechies and W. Sweldens, "Factoring wavelet transforms into lifting steps," *J. Fourier Anal. Appl*, vol. 4, no. 3, pp. 247–269, 1998.

[11] I. Daubechies, "The wavelet transfom, time-frequency localization and signal analysis," *IEEE Trans. on Information Theory*, vol. 36, no. 5, pp. 961–1005, Sept. 1990.

[12] W. Sweldens, "The lifting scheme: A custom-design construction of biorthogonal wavelets," *Applied and Computational Harmonics Analysis*, vol. 3, no. 15, pp. 186–200, 1996.

[13] R. A. DeVore, B. Jawerth, and B. J. Lucier, “Image compression through wavelet transfom coding,” *IEEE Trans. on Image Processing*, vol. 38, no. 2, pp. 719–746, Mar. 1992.

[14] A. S. Lewis and G. Knowles, “Image compression using 2D wavelet transfom,” *IEEE Trans. on Image Processing*, vol. 1, no. 2, pp. 244–250, April 1992.

[15] J. M. Shapiro, “Embedded image coding using zerotrees of wavelet coefficients,” *IEEE Trans. on Signal Processing*, vol. 41, no. 12, pp. 3445–3462, Dec 1993.

[16] A. Said and W. A. Pearlman, “A new fast and efficient image codec based on set partitioning in hiererchical trees,” *IEEE Trans. on Cir. and Syst. for Vid. Technology*, vol. 6, no. 3, pp. 243–250, Jun. 1996.

[17] ——, “An image multiresolution representation for lossless and lossy compression,” *IEEE Trans. on Image Processing*, vol. 5, no. 9, pp. 1303–1310, Sept. 1996.

[18] A. N. Skodras, C. A. Christopoulos, and T. Ebrahimi, “JPEG 2000: The upcoming still image compression standard,” in *Proc. of the 11th Portuguese Conf. on Pattern Recognition*, Porto, Portugal, May 11-12 2000, pp. 359–366.

[19] D. Santa-Cruz and T. Ebrahimi, “An analytical study of JPEG2000 functionalities,” in *Proc. of Int. Conf. in Image Proc.*, vol. 2, Vancouver, Canada, Sept. 10-13 2000, pp. 49–52.

[20] *Information Technology–JPEG2000 Image Coding System– Part 1: Core Coding System*, ISO/IEC 15444-1 Std., 2000.

[21] *Final Committee Draft, ”Information Technology–JPEG2000 Image Coding System–Part 2: Extension*, ISO/IEC 15444-2 Std., 2000.

[22] *Information Technology–JPEG2000 Image Coding System– Part 3: Motion JPEG2000*, ISO/IEC 15444-3 Std., 2002.

[23] *Information Technology–JPEG2000 Image Coding System–Part 4: Conformance Testing*, ISO/IEC 15444-4 Std., 2002.

[24] *Information Technology–JPEG2000 Image Coding System–Part 5: Reference Software*, ISO/IEC 15444-5 Std., 2003.

[25] *Final Committee Draft, ”Information Technology–JPEG2000 Image Coding System–Part 6: Compound Image File Format*, ISO/IEC 15444-6 Std., 2001.

[26] W. A. Pearlman, A. Islam, N. Nagaraj, and A. Said, “Efficient, low-complexity image coding with a set-partitioning embedded block coder,” *IEEE Trans. on Cir. and Syst. for Vid. Technology*, vol. 14, no. 11, pp. 1219–1235, Nov. 2004.

[27] D. Taubman, “High performance scalable image compression with EBCOT,” *IEEE Trans. on Image Processing*, vol. 9, no. 7, pp. 1158–1170, July. 2000.

[28] H. T. Hsiang and J. W. Woods, “Embedded image coding using zeroblock of subband/wavelet coefficients and context modeling,” in *Proc. IEEE Data Compression Conference*, Mar. 2001, pp. 83–92.

[29] C. Chrysafis, A. Said, A. Drukarev, A. Islam, and W. A. Pearlman, "SBHP - a low complexity wavelet coder," in *IEEE Int. Conf. Acoust., Speech and Sig. Proc. (ICASSP2000)*, 2000, pp. 2035–2038.

[30] H. Danyali and A. Mertins, "Fully spatial and SNR scalable SPIHT- based image coding for fransmission over heterogeous networks," *Journal of Telecommunications and Information Technology*, vol. 2, pp. 92–98, 2003.

[31] G. Xie and H. Shen, "Highly scalable, low-complexity image coding using zeroblocks of wavelet coefficients," *IEEE Trans. Cir. and Sys. for Video Technology*, vol. 15, no. 6, pp. 762–770, Jun. 2005. [Online]. Available: http://dx.doi.org/10.1109/TCSVT.2005.848311

[32] Y. Cho and W. A. Pearlman, "Quantifying the coding performance of zerotree of wavelet coefficients: Degree k-zerotree," *IEEE Trans. on Signal Processing*, vol. 55, no. 6, pp. 2525–2531, Jun 2007.

[33] A. A. Moinuddin, E. Khan, and M. Ghanbari, "Efficient algorithm for very low bit rate embedded image coding," *IET image processing*, vol. 2, no. 2, pp. 59–71, Apr 2008.

[34] A. Moinuddin, E. Khan, and M. Ghanbari, "Low complexity, efficient and embedded color image coding technique," *IEEE Trans. on Consum. Electronics*, vol. 54, no. 2, pp. 787–794, May 2008. [Online]. Available: http://dx.doi.org/10.1109/TCE.2008.4560161

[35] F. Wheeler and W. A. Pearlman, "SPIHT image compression without lists," in *IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 4, 05 - 09 June 2000, pp. 2047–2050.

[36] M. Latte, N. Ayachit, and D. Deshpande, "Reduced memory listless SPECK image compression," *Digital Signal Processing, Elsevier*, vol. 16, no. 6, pp. 817–824, Nov. 2006.

[37] I. E. Richardson, *H.264 and MPEG-4 Video Compression.* John Wiley and Sons, 2003.

[38] D. Le Gall, "MPEG: a video compression standard for multimedia applications," *Commun. ACM*, vol. 34, no. 4, pp. 46–58, Apr. 1991.

[39] Z. Xiong, O. Guleryuz, and M. Orchad, "A DCT based embedded image coder," *IEEE Signal Processing Lett.*, vol. 3, no. 11, pp. 289–290, Nov 1996.

[40] G. M. Davis and S. Chawla, "Image coding using optimised significance tree quantization," in *IEEE Data Compression Conference, Snowbird*, UT, USA, Mar 1997, pp. 387–396.

[41] D. M. Monoro and G. J. Dickson, "Zerotree coding of DCT coefficients," in *Proc. IEEE Int. conf. Image processing*, vol. 2, 1997, pp. 625–628.

[42] L. Junqiang and X. Zhuang, "Embedded image compression using DCT based subband decomposition and SLCCA data organisation," in *IEEE Workshop Multimedia Signal Processing, St. Thomsons, US Virgin Island*, 2002, pp. 81–84.

[43] X. Hou, G. Liu, and Y. Zou, "Embedded quadtree-based image compression in DCT domain." in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing*, vol. 3, Hong Kong, Apr 2003, pp. 277–280.

[44] H. S. Song and N. L. Cho, "DCT-based embedded image compression with a new coefficient sorting method," *IEEE Signal Processing Lett.*, vol. 16, no. 5, pp. 410–413, May 2009.

[45] R. Mukundan, "Image analysis by tchebichef moments," *IEEE Trans. on Image processing*, vol. 10, no. 9, pp. 1357–1364, 2001.

[46] ——, "Some computational aspects of discrete orthonormal moments," *IEEE Trans. on Image Processing*, vol. 13, no. 8, pp. 1055–1059, Aug. 2004. [Online]. Available: http://dx.doi.org/10.1109/TIP.2004.828430

[47] ——, "Radial Tchebichef invariants for pattern recognition," in *IEEE TENCON*, Melbourne, Qld., Nov. 21-24 2005, pp. 1–6.

[48] R. Mukundan and O. Hunt, "A comparison of discrete orthogonal basis functions for image compression," in *Image and Vision Computing, New Zealand, IVCNZ*, 2004, pp. 53–58.

[49] W. S. Lang, N. Z. Abu, and H. Rahmalan, "Fast 4x4 Tchebichef moment image compression," in *Int. Conf. on Soft Computing and Pattren Recognition*, 2009, pp. 295–300.

[50] N. A. Abu, N. Suryana, and R. Mukundan, "Perfect image reconstruction using discrete orthogonal moments," in *Proc. of the 4th IASTED Int. Conf. on Visualization, Imaging and Image Processing*, Marbela, Spain, 6-8 Sep 2004, pp. 903–907.

[51] K. Nakagaki and R. Mukundan, "A fast 4x4 forward discrete Tchebichef transform algorithm," *IEEE Signal Processing Lett.*, vol. 14, no. 10, pp. 684–687, Oct. 2007.

[52] S. Ishwar, P. Meher, and M. Swamy, "Discrete tchebichef transform-a fast 4x4 algorithm and its application in image/video compression," in *Int. Symp. on Circuits and Systems*, Seattle, WA, May 18-21 2008, pp. 260–263.

[53] S. Abdelwahab, "Image compression using fast and efficient discrete Tchebichef transform algorithm," in *Proc. of the 6th International Conference on Informatics and System*, Cairo, Egypt, March 27-29 2008, pp. 49–55.

[54] Z. Wang, "Pruning the fast discrete cosine transforms," *IEEE Trans. on Communications*, vol. 39, no. 5, pp. 640–643, May 1991.

[55] R. Stasinski, "On pruning the discrete cosine and sine transforms," in *IEEE MELECON*, vol. 1, May 12-15 2004, pp. 269–271.

[56] M. El-Sharkawy, "A fast recursive pruned DCT for image compression applications," in *Proc. of the 37th Midwest Symposium on Circuits and Systems, 1994.*, vol. 2. Lafayette, LA: IEEE, Aug 3-5 1994, pp. 887– 891.

[57] A. Skodras, "Fast discrete cosine transform pruning," *IEEE Trans. on Signal Processing*, vol. 42, no. 7, pp. 1833–1837, Jul. 1994. [Online]. Available: http://dx.doi.org/10.1109/78.298293

[58] Y.-M. Huang, J. ling Wu, and C.-L. Chang, "A generalized output pruning algorithm for matrix-vector multiplication and its application to compute pruning discrete cosine transform," *IEEE Trans. Signal Processing*, vol. 48, no. 2, pp. 561–563, 2000.

[59] S. Peng, “Complexity scalable video coding via IDCT data pruning,” in *Int. Conf. on Consumer Electronics*, Jun19-21 2001, pp. 74–75.

[60] N. P. Walmsley, A. N. Skodras, and K. M. Curtis, “A fast picture compression technique,” *IEEE Trans. on Consum. Electron.*, vol. 40, no. 1, pp. 11–19, Feb. 1994. [Online]. Available: http://dx.doi.org/10.1109/30.273656

[61] C. A. Christopoulos and A. N. Skodras, “Pruning the 2-D fast cosine transform,” in *Proc. of VII European Signal Processing Conference (EUSIPCO)*, Edinburgh, Scotland, UK, Sept. 13-16 1994, pp. 596–599.

[62] S. Antonio and N. Antonio, “Fast 8x8 DCT pruning algorithm,” in *IEEE Int. Conf. on Image Processing*, vol. 2, 2005, pp. 317–320.

[63] R. Mukundan, “Transform coding using discrete Tchebichef polynomials,” in *Sixth IASTED Int. Conf. of Visualization Imaging and Image Processing.* Computer science and software engineering, University of Canterbury, 2006.

[64] A. Balvias, “Visual analysis in unspecialized receptive fields as an orthogonal series expansion,” *Neurophysiology, Springer*, vol. 6, no. 2, pp. 168–173, 1974.

[65] C. Bokariya and J. Dong, “Performance study of discrete Tchebichef transform based MPEG video codec,” in *IPCV’08.* Las Vegas: IEEE, 2008, pp. 64–68.

[66] W. K. Cham, “Development of integer cosine transforms by the principle of dyadic symmetry,” *IEE Proc. Comm. Speech and Vision*, vol. 136, no. 4, pp. 276–282, Aug 1989.

[67] V. Dimitrov, K. Wahid, and G. Jullien, “Multiplication-free 8x8 2D DCT architecture using algebraic integer encoding,” *IET Electronic Lett.*, vol. 40, no. 20, pp. 1310–1311, Sep. 30 2004.

[68] K. Lengwehasatit and A. Ortega, “Scalable variable complexity approximate forward DCT,” *IEEE Trans. Cir. and Sys. for Video Technology*, vol. 14, no. 11, pp. 1236–1248, Nov. 2004.

[69] T. D. Tran, “The BinDCT: Fast multiplierless approximation of the DCT,” *IEEE Signal Processing Lett.*, vol. 7, no. 6, pp. 141–144, 1999.

[70] T. I. Haweel, “A new square wave transform based on the DCT,” *Signal Processing*, vol. 81, no. 11, pp. 2309–2319, Nov. 2001.

[71] S. Bouguezel, M. O. Ahmad, and M. N. S. Swamy, “Low-complexity 8x8 transform for image compression,” *IET Electronic Lett.*, vol. 44, no. 21, pp. 1949–1950, Oct. 2008.

[72] K. Beauchamp, *Application of Walsh and related functions.* New York: Academic Press, 1984.

[73] J. Pender and D. Covey, “New square wave transform for digital signal processing,” *IEEE Trans. Signal Processing*, vol. 40, no. 8, pp. 2095–2097, Aug. 1992.

[74] T. I. Haweel and A. M. Alhasan, *A simplified square wave transform for signal processing*, ser. Contemporary Math. American Math. Society, Jan. 3-9 1994, vol. 190.

[75] A. Amira, "An FPGA based transform for discrete Hartly transform," in *Int. Conf. on Visual Information Engineering*, July 7-9 2003, pp. 137–140.

[76] P. Chungan, C. Xixin, Y. Dunshan, and Z. Xing, "A 250 MHz optimised distributed architecture of 2D 8x8 DCT," in *Int. Conf. on ASIC*, ser. 7, no. 189-192, Guilin, China, Oct. 22-25 2007.

[77] A. M. Shams, A. Chidanandan, W. Pan, and M. A. Bayoumi, "NEDA: a low-power high-performance DCT architecture," *IEEE Trans. on Signal Processing*, vol. 54, no. 6, pp. 955–964, 2006.

[78] H. Pan, W.-C. Siu, and N.-F. Law, "A fast and low memory image coding algorithm based on lifting wavelet transform and modified SPIHT," *Signal Processing: Image Commun, Elsevier*, vol. 23, no. 3, pp. 146–161, Mar. 2008.

[79] S. Servetto, K. Ramchandran, and M. T. Orchand, "Image coding based on morphological representation of wavelet data," *IEEE Trans. on Image processing*, vol. 8, no. 9, pp. 1161–1174, Sep. 1999.

[80] B. B. Chai, J. Vass, and X. Zhuang, "Significance-linked connected component analysis for wavelet image coding," *IEEE Trans. on Image processing*, vol. 8, no. 6, pp. 774–784, Jun. 1999.

[81] R. K. Senapati, U. C. Pati, and K. K. Mahapatra, "A fast zigzag-pruned 4x4 DTT algorithm for image compression," *WSEAS Trans. on Signal Processing*, vol. 7, no. 1, pp. 34–43, Jan. 2011.

[82] M. G. Ramos, S. S. Hemami, and M. A. Tamburro, "Psychovisually-based multiresolution image segmentation," in *Proc. of the 1997 Int. Conf. on Image Processing (ICIP '97)*, vol. 3, Santa Barbara, CA, Oct. 26-29 1997, pp. 66–69.

[83] X. Wu and A. Gersho, "Rate-constrained picture adaptive quantization for JPEG baseline coders." in *Proc. IEEE ICASSP*, vol. 5, Apr 1993, pp. 389–392.

[84] P. Duhamel and C. Guillemot, "Polynomial transform computation of 2D DCT," in *Proc. ICASSP*, 1990, pp. 1515–1518.

[85] E. Feig and S. Winograd, "Fast algorithms for the discrete cosine transform," *IEEE Trans. on Signal Processing*, vol. 40, no. 9, pp. 2174–2193, 1992.

[86] M. Vetterli, "Fast 2-D discrete cosine transform," in *Proc. IEEE ICASSP*, vol. 10, Apr. 1985, pp. 1538–1541.

[87] H. I. Saleh, "A fast block-pruned 4x4 DTT algorithm for image compression," *Int. J. of Computer Theory and Engineering*, vol. 1, no. 3, pp. 1793–8201, Aug. 2009.

[88] L. Kotoulas and I. Andreadis, "Fast computation of Chebyshev moments," *IEEE Trans. Cir. and Sys. for Video Technolgy*, vol. 16, no. 7, pp. 884–888, Jul. 2006.

[89] L.-B. Zhang and M.-Q. Zhou, "Embedded reversible medical image compression using integer wavelet transform," in *Proceedings of the 2006 Int. Conf. on Intelligent Computing: Part II*, ser. ICIC'06. Berlin, Heidelberg: Springer-Verlag, 2006, pp. 620–624. [Online]. Available: http://dl.acm.org/citation.cfm?id=1882540.1882621

[90] M. Antonini, M. M. Barlaud, P. Mathieu, and I. Daubechies, "Image coding using wavelet transform," *IEEE Trans. on Image processing*, vol. 1, no. 2, pp. 205–220, April 1992.

[91] A. Calderbank, I. Daubechies, W. Sweldens, and B.-L. Yeo, "Lossless image compression using integer to integer wavelet transforms," in *Proceedings of the 1997 International Conference on Image Processing (ICIP '97)*, ser. ICIP '97, vol. 1, Washington, DC, USA, 1997, pp. 596–599.

[92] D. Santa-Cruz, R. Grosbois, and T. Ebrahimi, "JPEG 2000 performance evaluation and assessment," *Signal Processing: Image Communication*, vol. 17, no. 1, pp. 113–130, 2002.

[93] H. Everett, "Generalized lagrange multiplier method for solving problems of optimization allocation of resources," *Operational Research*, vol. 11, no. 3, pp. 399–417, 1963.

[94] N. Cho and S. Lee, "A fast 4x4 dct algorithm for the recursive 2-d dct," *IEEE Trans. Signal Processing*, vol. 40, no. 9, pp. 2166–2173, Sep. 1992. [Online]. Available: http://dx.doi.org/10.1109/78.157217

[95] H. S. Malvar, A. Hallapuro, M. Karczewicz, and L. Kerofsky, "Low-complexity transform and quantization in H.264/AVC," *IEEE Trans. Cir. and Sys. for Video Technology*, vol. 13, no. 7, pp. 598–603, Jul. 2003. [Online]. Available: http://dx.doi.org/10.1109/TCSVT.2003.814964

[96] N. Cho and S. Lee, "Fast algorithm and implementation of 2D discrete cosine transform," *IEEE Trans. Signal Processing*, vol. 38, no. 3, pp. 297–305, Mar. 1991.

[97] H. Shu, H. Zhang, B. Chen, P. Haigron, and L. Luo, "Fast computation of Tchebichef moments for binary and grayscale images," *IEEE Trans. on Image Processing*, vol. 19, no. 12, pp. 3171–3180, Dec. 2010. [Online]. Available: http://dx.doi.org/10.1109/TIP.2010.2052276

[98] [Online]. Available: http://www.jpeg.org/jpeg2000/

[99] B.-J. Kim, Z. Xiong, and W. A. Pearlman, "Low bit-rate scalable video coding with 3-D set partitioning in hierarchical trees (3-D SPIHT)," *IEEE Trans. Cir. and Sys. for Video Technology*, vol. 10, no. 8, pp. 1374–1387, Dec. 2000.

[100] Y.-L. Jeang, H.-Y. Wang, and C.-C. Wong, "A scalable wavelet image coder based on zeroblock and array," in *Proc. of the 2008 3rd Int. Conf. on Innovative Computing Information and Control*, ser. ICICIC '08. Washington, DC, USA: IEEE Computer Society, 2008, pp. 184–187.

[101] Y. Cho, "Resolution scalable and random access decodable image coding with low time complexity," Ph.D. dissertation, Rensselaer Polytechnic Institute, 2005.

[102] W. Sweldens, "The lifting scheme: A new philosophy in biorthogonal wavelet constructions," in *Wavelet Applications in Signal and Image Processing III*, 1995, pp. 68–79.

[103] Y. Cho, S. Cho, and W. A. Pearlman, "Fast and constant time random access decoding with $log_2 n$ block seek time." in *Storage and Retrieval Methods and Applications for Multimedia*, ser. SPIE Proceedings, R. Lienhart, N. Babaguchi, and E. Y. Chang, Eds., vol. 5682. SPIE, 2005, pp. 10–17.

[104] G. Liu, X. Zeng, F. Tian, K. Chaibou, and Z. Zheng, "A novel direction-adaptive wavelet based image compression," *Int. J. of Electronic and Communications, Elsevier*, vol. 64, no. 6, pp. 531–539, Jun. 2010.

[105] M. Lanuzza, S. Perri, P. Corsonello, and G. Cocorullo, "An efficient wavelet image ender for FPGA-based design," in *Proc. of IEEE Workshop on SIPS*, 2005, pp. 652–656.

[106] J. Lian, K. Wang, and J. Yang, "Listless zerotree image comparison algorithm," in *Proc. of 8th Int. Conf. on Signal Processing*, vol. 2, Beijing, 2006, pp. 1–4.

[107] Y. Li, J. Song, C. Wu, K. Liu, J. Lie, and K. Wang, *FPGA Design of Listless SPIHT for On-board Image Compression.* Satellite Data Compression, Springer-Verlag, 2011, ch. 4, pp. 67–85.

[108] F. Douak, R. Benzid, and N. Benoudjit, "Color image compression algorithm based on DCT transform combined to an adaptive block scanning," *Int. J. of Electronic and Communications, Elsevier*, vol. 65, no. 1, pp. 16–26, Jan. 2011.

[109] I. B. Witten, R. L. Neal, and J. G. Cleary, "Arithmetic coding for data compression," *Communications of the ACM*, vol. 30, no. 6, pp. 520–540, Jun. 1987.

Dissemination of Work

International Journals

1. **Ranjan K Senapati**, Umesh C. Pati, Kamala K. Mahapatra: "Listless block-tree set partitioning algorithm for very low bit rate embedded image compression" *Int. Journal of Electronics and Communication, Elsevier, vol. 66, issue 12, pp. 985-995, Dec. 2012.*
2. **Ranjan K Senapati**, Umesh C. Pati, and Kamala Kanta Mahapatra: "Low bit rate image compression using Hierarchical listless block-tree DTT algorithm" *Int. Journal of Image and Graphics, World scientific, vol. 13, no.1, pp. 1-22, Apr. 2013.*
3. **Ranjan K Senapati**, Umesh C. Pati, Kamala Kanta Mahapatra: "A reduced memory, low complexity embedded image coding algorithm using Hierarchical listless DTT" *IET Image Processing.* (In Press)
4. **Ranjan K Senapati**, Umesh C. Pati, Kamala K. Mahapatra: "An efficient sparse 8 × 8 orthogonal transform matrix for color image compression" *Int. Journal of Signal and Imaging System Engineering (IJSISE), Inderscience publishers, vol. 6, no.1, pp. 16-23, Jan. 2013.*
5. **Ranjan K Senapati**, Umesh C. Pati, Kamala K. Mahapatra: "A fast zigzag prune 4 × 4 DTT algorithm for image Compression" *World Scientific Engineering Academy and Society (WSEAS) Transactions on Signal Processing, vol. 7, issue 1, pp. 34-43, Jan. 2011.*
6. **Ranjan K Senapati**, Umesh C. Pati, and Kamala Kanta Mahapatra: "Reduced memory listless scalable embedded set partitioning algorithms for image compression" *Int. Journal of Visual Representation and Image Communications, Elsevier.* (Under Review)

International Conferences

1. **Ranjan K Senapati**, Umesh C. Pati, and Kamala K. Mahapatra: "Image compression using discrete Tchebichef transform algorithm" In *Proc. of IEEE Int. Conf. on Communication Network and Computing (CNC), Calicut, Kerala, India, pp. 104-108, Oct. 2010.*
2. **Ranjan K Senapati**, Umesh C. Pati, and Kamala K. Mahapatra: "A novel zigzag prune discrete Tchebichef moment based image compression algorithm" In *Proc. of IEEE Int. Conference on Computational Intelligence and Communication Network (CICN), Bhopal, India, pp. 73-78, Nov. 2010.*

3. **Ranjan K Senapati**, Umesh C. Pati, and Kamala K. Mahapatra: "A low complexity orthogonal 8 × 8 transform matrix for fast image compression" In *Proc. of IEEE Annual India Int. Conf. (INDICON), Kolkata, India, pp. 1-4, Dec 2010.*

4. **Ranjan K Senapati**, Umesh C. Pati, and Kamala K. Mahapatra: "An efficient sparse 8 × 8 orthogonal transform matrix for color image compression" In *Proc. of Int. Conference on Electronics Systems (ICES), NIT Rourkela, India, 7-9th Jan. 2011.*

5. **Ranjan K Senapati**, Umesh C. Pati, and Kamala K. Mahapatra: "A novel hybrid HVS based embedded image coding algorithm using DTT and SPIHT" In *Proc. of IEEE Int. Conf. on Devices and Communication (ICDeCom), Mesra, Ranchi, pp.1-5, Feb. 2011.*

6. **Ranjan K Senapati**, Umesh C. Pati, and Kamala K. Mahapatra: "A low complexity embedded image coding algorithm using hierarchical listless DTT" In *Proc. of* 8^{th} *IEEE Int. Conf. on Information, Communications and Sgnal Processing (ICICS), NTU, Singapore, pp. 1-5, Dec. 2011.*

BIO-DATA

Ranjan Kumar Senapati

Date of Birth: 11^{th} April, 1977

Correspondence:
PhD Scholar, Department of Electronics and Communication Engineering,
National Institute of Technology Rourkela, India – 769 008.

Ph: +91 94397 85897 (M)

e-mail: rksphd@gmail.com

Qualification

- Ph.D. (Continuing)
  National Institute of Technology Rourkela, Odisha, India
- M.Tech. (Electronics and Instrumentation Engineering)
  National Institute of Technology Warangal, Andhra Pradesh, India [First division]
- AMIETE
  The Institution of Electronics & Telecommunication Engineers, New Delhi, India [First class]
- +2 (Science)
  Council of Higher Secondary Education, Odisha, India [Second division]
- 10th
  Board of Secondary Education, Odisha, India [First division]

Professional Experience

- Asso. Prof. in the Dept. of ECE (Aug 2008-till today)
  GMR Institute of Technology, Srikakulum (Dist.), 532127, Andhra Pradesh, India.
- Asst. Prof. in the Dept. of ECE (July 2006-July 2008)
  GMR Institute of Technology, Srikakulum (Dist.), 532127, Andhra Pradesh, India.
- Asst. Prof. in the Dept. of ECE (July 2004-June 2006)
  Vasavi College of Engineering, Hyderabad, 500031, Andhra Pradesh, India.

- Asst. Prof. in the Dept. of ECE (May 2003-July 2004)
  Mahatma Gandhi Institute of Technology, Hyderabad, 500075, Andhra Pradesh, India.

Publications

- 5 Journal Articles
- 6 Conference Articles

Appendix A

To analyze the computational complexity of SPIHT, LSK and ILSK, the following analysis is presented.

Case 1: The complexity required by SPIHT:

Let L be the number of decomposition levels. Assuming the number of coefficients hold by LIP and LSP is equal to N_c. At the lowest bit plane N_c equals to number of pixels in the original image. Total number of coefficients in a single tree, including type A and type B descendants will be:

$$\left\{\left(4^0 \times \sum_{n=1}^{L} 4^n\right) + \left(4^1 \times \sum_{n=1}^{L-1} 4^n\right) + \left(4^2 \times \sum_{n=1}^{L-2} 4^n\right) + ... + \left(4^{L-1} \times \sum_{n=1}^{L-5} 4^n\right)\right\} \tag{7.1}$$

The first term of Eqn. 7.1 is the total number of coefficients of the tree having root at the coarsest subband. The second term is the total number of trees whose roots are the offspring (type A descendants) of the coefficient located at coarsest level. The third term is the total number of trees whose roots are the type B descendants of the coefficient located at coarsest level. The other terms can be defined in a similar way. Therefore total number of comparisons with the lowest threshold will be

$$N_c + \left[\left\{\frac{3}{4} \times \left(\frac{1}{4}\right)^L \times N_c\right\} \times \left\{4^0 \times \sum_{n=1}^{L} 4^n + 4^1 \times \sum_{n=1}^{L-1} 4^n + 4^2 \times \sum_{n=1}^{L-2} 4^n + ... + 4^{L-1} \times \sum_{n=1}^{L-5} 4^n\right\}\right] \tag{7.2}$$

where $\left\{\frac{3}{4} \times \left(\frac{1}{4}\right)^L \times N_c\right\}$ is the number of tree roots. $\left(\frac{1}{4}\right)^L \times N_c$ is the total number of coefficients in the coarsest subband.

Eqn. 7.1 can be approximated to

$$L \times \sum_{n=1}^{L} 4^n \tag{7.3}$$

Therefore, Eqn. 7.2 becomes:

$$N_c + \left[\left\{\frac{3}{4} \times \left(\frac{1}{4}\right)^L \times N_c\right\} \times \left\{L \times \sum_{n=1}^{L} 4^n\right\}\right] \tag{7.4}$$

Eqn. 7.4 is the total number of comparisons (worst case) required by SPIHT with the lowest threshold.

Case 2. The complexity required by ILSK:

ILSK uses 3 state table markers.

$MV[k]$ state table can address any coefficient in an image. This gives rise to total number of comparisons equal to N_c. $MF[k]$ state table compares the coefficients in a wavelet level. Therefore, total number of comparisons in the lowest threshold will be:

$$3 \times \left[\left(\frac{1}{4}\right)^{L} \times N_c\right] + 3 \times \left[\left(\frac{1}{4}\right)^{L-1} \times N_c\right] + \ldots\ldots\ldots\ldots\ldots \left[\left(\frac{3}{4}\right) \times N_c\right] \tag{7.5}$$

The multiplication term 3 in Eqn. 7.5 is due the fact that each level consists of three subbands. In Eqn. 7.5, $\left(\frac{1}{4}\right)^{L} \times N_c$ is the number of coefficients at subband level L. Similarly, $\left(\frac{1}{4}\right)^{L-1} \times N_c$ is the total number coefficients at subband level $L-1$.

$M[k]$ state table compares coefficients across several levels. Total number of comparisons by $M[k]$ state table markers are:

$$\begin{aligned}&3 \times \left[\left(\tfrac{1}{4}\right)^{L} \times N_c + \left(\tfrac{1}{4}\right)^{L-1} \times N_c + \ldots\ldots\ldots\ldots \left(\tfrac{1}{4}\right) \times N_c\right] + \\ &3 \times \left[\left(\tfrac{1}{4}\right)^{L-1} \times N_c + \left(\tfrac{1}{4}\right)^{L-2} \times N_c + \ldots\ldots \left(\tfrac{1}{4}\right) \times N_c\right] + \ldots\ldots\ldots 3 \times \left[\left(\tfrac{1}{4}\right)^{2} \times N_c\right]\end{aligned} \tag{7.6}$$

Therefore, total number of comparisons with lowest threshold by ILSK will be sum of the comparisons due to $MV[k]$, $MF[k]$ and $M[k]$ state table markers.

$$\begin{aligned}&N_c + 3 \times \left[\left(\tfrac{1}{4}\right)^{L} \times N_c + \left(\tfrac{1}{4}\right)^{L-1} \times N_c + \ldots\ldots\ldots\ldots \left(\tfrac{1}{4}\right) \times N_c\right] + \\ &+3 \times \left[\left(\tfrac{1}{4}\right)^{L} \times N_c\right] + 3 \times \left[\left(\tfrac{1}{4}\right)^{L-1} \times N_c\right] + \ldots\ldots\ldots\ldots\ldots \left[\left(\tfrac{3}{4}\right) \times N_c\right] + \\ &3 \times \left[\left(\tfrac{1}{4}\right)^{L-1} \times N_c + \left(\tfrac{1}{4}\right)^{L-2} \times N_c + \ldots\ldots \left(\tfrac{1}{4}\right) \times N_c\right] + \ldots\ldots\ldots 3 \times \left[\left(\tfrac{1}{4}\right)^{2} \times N_c\right]\end{aligned} \tag{7.7}$$

Eqn. 7.7 can be approximated to

$$N_c + 3 \times 3N_c \left[\left(\frac{1}{4}\right)^{L} + \left(\frac{1}{4}\right)^{L-1} + \ldots\ldots \frac{1}{4}\right] \approx N_c + 3 \times 3N_c \times \frac{1}{2} \approx 5.5N_c \tag{7.8}$$

Case 3: Comparison required by LSK:

It is same as the comparisons required by $MV[k]$ and $MF[k]$ markers in ILSK. The comparison due to $MF[k]$ state table markers are included because, LSK needs to test the significance of each subband before partition the subbands to quad blocks by $MV[k]$ state table markers. Therefore, total number of comparisons in LSK $\approx 4.5N_c$.

Comparing Eqn. 7.4 and Eqn. 7.8, the number of comparisons required by SPIHT is larger than ILSK. For example, if $L = 6$ and $N_c = 512 \times 512$, The total number of computation by SPIHT is 1.3 times ILSK. It may be noted that the overhead incurred because of movement of coefficients between the list arrays and the overhead to distinguish between type A and type B descendants by the SPIHT during bit plane passes is not included into the Eqn. 7.4. This is a memory management task. This is clearly evident from Table 3.9 in Chapter-3 and Table 4.10 in chapter-4 that it accounts a more time consuming operation. This give rise to a near exponential increase in time complexity, i.e. approximated to $O(N_c log N_c)$ operations using empirical methods. However, this mechanism is completely absent in ILSK or LSK. But while comparing Eqn. 7.4 and Eqn. 7.8, the computational complexity in SPIHT, ILSK and LSK is approximated to $O(N_c)$ operations.