1

MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS
STANDARD REFERENCE MATERIAL 1010a
(ANSI and ISO TEST CHART No. 2)

**National Library
of Canada**

**Bibliothèque nationale
du Canada**

Canadian Theses Service

Service des thèses canadiennes

**Ottawa, Canada
K1A 0N4**

NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents.

MULTI-STAGE HYBRID CODING OF DIGITAL IMAGES

YIYAN WU, M. ENG.

**A thesis submitted to the Faculty of
Graduate Studies and Research in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy**

**Faculty of Engineering
Department of Systems and Computer Engineering
Carleton University
Ottawa, Ontario**

**April 23, 1990
©1990 Yiyang Wu**

National Library
of Canada

Bibliothèque nationale
du Canada

Canadian Theses Service Service des thèses canadiennes

Ottawa, Canada
K1A 0N4

The author has granted an irrevocable non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of his/her thesis by any means and in any form or format, making this thesis available to interested persons.

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

L'auteur a accordé une licence irrévocable et non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de sa thèse de quelque manière et sous quelque forme que ce soit pour mettre des exemplaires de cette thèse à la disposition des personnes intéressées.

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

ISBN 0-315-58989-2

Carleton University
Ottawa, Canada K1S 5B6

Thesis contains coloured
photographs &/or explanatory tables
which when microfilmed may lose their
significance. The hardcopy of the
thesis is available upon request from
Carleton University Library.

The undersigned recommend to the Faculty of Graduate Studies and Research acceptance of the thesis, "Multi-Stage Hybrid Coding of Digital Images" submitted by Yiyan Wu, M. Eng., in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

.....

Thesis Supervisor

.....

External Examiner

.....

Chairman, Department of Systems
and Computer Engineering
Carleton University

April 23, 1990

Abstract

There are three criteria that may be used to evaluate the performance of an image compression algorithm: computational complexity, compression ratio, and fidelity. Low computational complexity implies short coding delay and low hardware expense. High compression ratio means greater channel efficiency and reduced transmission time. High fidelity indicates low distortion, or coding errors, and high reliability. Unfortunately, all three cannot be achieved uniformly.

In this thesis, different coding techniques and fidelity measurements are first reviewed. Then, three hybrid coding techniques, BTC-VQ-DCT Hybrid Coding, Minimax Error Multi-Level Block Truncation Coding and Single Bit-Map Block Truncation Coding, which achieve compromises between computational complexity, compression ratio, and fidelity are presented. Each coding scheme contains more than one data compression stage, which removes the redundancy more completely than that of single stage coding. Different coding techniques are implemented in each step.

BTC-VQ-DCT coding combines the advantages of the simple computation and edge preservation of BTC, the high fidelity and high compression ratio of adaptive DCT, and the high compression ratio and good subjective performance of VQ. The overall computational complexity of BTC-VQ-DCT coding is much less than either DCT and VQ, while the fidelity performance is competitive. The compression ratio is about 10:1.

Multi-Level Block Truncation Coding can preserve the spatial details in composite imagery while providing compression ratios of up to 6:1. An adaptive quantizer level

allocation algorithm is introduced that minimizes the maximum quantization error in each block, and substantially reduces the computational complexity in allocation of the optimal quantization levels. Vector quantization and predictive coding are used to further compress the coded data. The coding algorithm demonstrates very high fidelity, with no perceived visual degradation, and very low computational complexity.

Single Bit-Map Block Truncation Coding is a colour image data compression algorithm. BTC is first implemented to encode a weighted trichromatic image. The bit-map generated is then used to compress each of the R, G and B images separately, removing the inter-colour correlations. The weighted trichromatic image is generated by properly weighting the R, G and B images so that the MSE between the original and reconstructed R, G and B images is minimized. Another approach using a bit-wise majority rule on the R, G and B BTC bit-maps is also discussed. The algorithm preserves the spatial details in the image content and has low computational complexity. An 8:1 compression can be achieved. The truncated data is further compressed using adaptive DCT. Differential coding and error feedback techniques are also implemented to reduce the bit rate and improve the MSE performance. The total compression ratio is about 12:1. By exploiting human visual system characteristics, a 14:1 compression of NTSC colour TV signals can be achieved.

Possible areas of further research are also discussed.

Acknowledgements

I extend my deepest gratitude and appreciation to my thesis supervisor, *Dr. David C. Coll*, for his patient guidance, supervision and friendship throughout this research work.

I would like to record my sincere thanks to *Dr. Robert Morris*, *Dr. David Falconer*, *Dr. Howard Schwartz* and *Dr. Morris Goldberg* for their valuable guidance and encouragement.

I also want to take this opportunity to give my special thanks to my colleagues and friends, *Mr. Ian MacLaren*, *Mr. Steve Istvan Hernadi*, *Mr. Janli Liu*, *Dr. Limin Wang* and *Dr. Xiaobing Li* for valuable discussions during the research period.

Finally, I wish to dedicate this thesis to my wife, *Ms. Ming Zhong*, and my parents, *Mr. Bin-Lin Wu* and *Ms. Shang-Lu Jiang* for their patience and encouragement throughout this work.

Contents

1	INTRODUCTION	1
2	REVIEW OF IMAGE CODING	6
2.1	Lossless Compression	6
2.1.1	Huffman Coding	7
2.1.2	Run-Length Coding	7
2.1.3	Block Coding	7
2.1.4	Contour Coding	8
2.1.5	Rectangular Coding	8
2.1.6	Quadtree Representation	8
2.1.7	Predictive Differential Quantization	8
2.2	Lossy Compression	9
2.2.1	Spatial Coding	9
2.2.2	Transform Coding	17
2.2.3	Hybrid Transform/Predictive Coding	32
2.2.4	Vector Quantization	32
2.3	Summary	37

3	IMAGE FIDELITY MEASUREMENTS	39
3.1	Different Measurement Methods	39
3.2	Subjective Assessment	39
3.3	Objective Measurement	41
3.4	Human Visual System and Its Application in Image Data Compression	43
4	IMAGE STATISTICS	50
4.1	Pixel Correlation	52
4.2	Entropy	58
4.3	Spectrum	64
5	BTC-VQ-DCT HYBRID CODING OF DIGITAL IMAGES	66
5.1	Motivation of BTC-VQ-DCT Coding	67
5.2	BTC-VQ-DCT Coding	68
5.2.1	Phase-I: BTC Encoding of the Input Image	69
5.2.2	Phase-II: VQ Encoding of the Bit-Map	70
5.2.3	Phase-III: Adaptive DCT Encoding of the High- and Low-Mean Sub-Images	73
5.3	System Performances	80
5.3.1	Computational Complexity of BTC-VQ-DCT vs. Direct VQ and DCT	80
5.3.2	Encoding and Decoding Delay of VQ, DCT and BTC-VQ-DCT	80
5.3.3	Error Performance of VQ, DCT and BTC-VQ-DCT	82
5.4	Bit Rate, NMSE and Computational Complexity	83
5.5	Summary	86

6	MULTI-LEVEL BTC USING A MINIMAX ERROR CRITERION FOR HIGH FIDELITY COMPRESSION OF DIGITAL IMAGES	90
6.1	Introduction	91
6.2	Minimax Error Multi-Level BTC and A Fast Quantizer Level Allocation Scheme	93
6.2.1	Minimax Error Multi-Level BTC	93
6.2.2	Fast Quantizer Level Allocation Scheme	99
6.2.3	Performance of Minimax Error Multi-Level BTC	102
6.3	Quantization Levels Data Compression	108
6.3.1	Intra-Block Prediction	108
6.3.2	Inter-Block Prediction	112
6.3.3	Higher Order Intra-Block Prediction	116
6.3.4	Coding of Prediction Errors	119
6.4	Lossless Quaternary Plane Data Compression	128
6.4.1	Correlation of Quaternary Plane Elements	128
6.4.2	Lossless Coding of 4-dimensional Vector Pattern	136
6.4.3	Compression Overhead Using Pattern Symmetry Property	141
6.4.4	Block Adaptation and Parallelism	143
6.4.5	Quaternary Plane Encoding Diagram.	144
6.4.6	The Possibility of Using Larger Pattern Size	145
6.4.7	Transmission Error Property	147
6.5	Rate Distortion Curves of Minimax Error Multi-Level Block Truncation Coding with Lossless Post-Quantization Compression	148
6.6	Quaternary Plane Data Compression Using Vector Quantization	152
6.7	Summary	155

7	SINGLE BIT-MAP BTC OF COLOUR IMAGES	156
7.1	Motivation for Single Bit-Map BTC	157
7.2	Block Truncation Coding	158
7.3	BTC of Colour Images	160
7.3.1	BTC OF Colour Images Using Single Bit-Map	160
7.3.2	Minimum MSE Approach	161
7.3.3	Bit-Map Majority Rule Approach	165
7.3.4	Post Quantization Data Compression Using Adaptive DCT . .	167
7.4	NTSC Colour TV Signal Compression	170
7.5	Conclusion	174
8	SUMMARY AND FUTURE RESEARCH WORK	178
8.1	A Research Summary	178
8.2	Suggestions fo Futher Research Work	180

List of Figures

2.1	Image compression methods.	11
2.2	Mean square error performance of image transforms as a function of block size.	18
2.3	Eigenvectors of (8×8) Toeplitz matrix ($\rho = 0.9$) and basis vectors of the DCT [55].	22
2.4	DFT and DCT spectrum [56].	23
2.5	Fast discrete cosine transform flowchart [66].	25
2.6	Comparison of computational steps for conventional FDCT, FFT and FDCT [66]: (a) Additions. (b) Multiplications.	26
2.7	8×8 DCT coefficients	28
2.8	Block diagram of scene adaptive DCT coding system [70].	31
2.9	Zigzag scan of DCT coefficients [70].	32
3.1	Image compression system.	41
3.2	Contrast sensitivity measurement [33].	44
3.3	Spatial impulse response of cone retina cells.	45
3.4	Block diagram of the human visual system [107].	46
3.5	Spatial frequency response of HVS [58].	47
4.1	Correlation coefficients of "girl's" image.	53

4.2	Correlation coefficients of CSI (with/without overlays).	54
4.3	A 2×2 grid of pixels.	56
4.4	Spectrum of girl's image.	65
4.5	Spectrum of composite satellite image.	65
5.1	A pixel block data before and after BTC.	70
5.2	Adaptive DCT coding of high- and low-mean sub-images.	75
5.3	BTC-VQ-DCT coding system diagram.	81
5.4	Rate distortion curves of VQ, DCT and BTC-VQ-DCT.	84
6.1	Probability distribution functions.	94
6.2	A block histogram with $x_{min} = 5$, $x_{max} = 8$ and $N = 3$	96
6.3	Block histogram with "zero" intervals.	97
6.4	In the case that non-zero components can be partitioned into N equal-length intervals.	98
6.5	An example of the threshold-dilation algorithm to "fast search" output levels which minimize $ e_{max} $ over a pixel block.	100
6.6	Diagram of "zero-masking" scheme.	103
6.7	Zero-masking scheme implemented on a block with overlays.	104
6.8	MSE performance comparison of different quantizers (4-level).	105
6.9	MSE performance comparison using CSI with/without overlay.	107
6.10	Triangular distribution.	109
6.11	2nd order 2-dimensional predictor.	115
6.12	Symmetric triangular distribution.	117
6.13	Flowchart of the optimal bit allocation algorithm.	122
6.14	Mathematic model of a 4-level quantizer.	130

6.15 (a) Hard limiting model (b) Poisson process and telegraph signal. .	131
6.16 Correlation functions of raw and truncated data.	135
6.17 4-dimensional vector-byte file histogram.	136
6.18 Variable block entropy code for 4-element pattern encoding.	140
6.19 Quaternary plane encoding diagram.	146
6.20 Rate distortion curves of BTC with various quantization levels and block sizes.	149
6.21 Rate distortion curves of BTC-VQ-DPCM coding and VQ.	153
7.1 System diagram of MSE optimized single bit-map BTC of colour images.	166
7.2 Adaptive DCT coding of subsampled colour plane (red plane).	168
7.3 System diagram of SBMBTC-DCT coding of NTSC signals.	172

List of Tables

2.1	Comparison of image compression methods.	10
3.1	Five-point adjectival scales for quality and impairment, and associated number scores [104].	40
4.1	Values of correlation coefficients of “girl’s” image and CSI (with/without overlays).	55
4.2	Predictor coefficients and prediction gain of different predictors [13]. .	57
4.3	Predictor coefficients and prediction gain for test images.	58
4.4	Entropies and conditional entropies.	62
4.5	Differential entropies.	63
5.1	MSE performance of computer simulation results for the Girl image, codebook size = 256.	79
5.2	Computation complexities of VQ, DCT and BTC-VQ-DCT.	82
5.3	Simulation results of BTC-VQ-DCT.	83
5.4	Bit rate, NMSE and computational complexity.	86
6.1	(a) Original data; (b) 4-level 8×8 BTC; (c) 2-level 4×4 BTC. . . .	106
6.2	Values of Pixel Correlation Coefficients of Different Images.	111
6.3	Multi-dimension variable block code bit allocation map.	123

6.4	Quantization level prediction error entropies and bit rates.	124
6.5	Bit rate change due to statistical mis-match.	125
6.6	Code distribution for CSI (visual band).	126
6.7	High probability patterns (CSI).	138
6.8	High probability patterns (Girl).	139
6.9	Overhead code before and after compression.	142
6.10	Block entropy and bit rates.	144
6.11	MSE performances of BTC with various quantization levels and block sizes.	150
6.12	MSE performances of BTC-VQ-DPCM coding and VQ.	154
7.1	MSE Performance of BTC of R, G and B planes.	162
7.2	MSE performances of SBMBTC-DCT coding	169
7.3	MSE performances of SBMBTC-DCT coding of NTSC colour signal.	171
7.4	Bit rate of SBMBTC-DCT coding of NTSC colour signal.	173

List of Pictures

Picture I	Girl with hat.	51
Picture II	Composite satellite image.	51
Picture III	(a) Original image; (b) Bit-map; (c) High- and low-mean images, and their difference image.	87
Picture IV	(a) Reconstructed image; (b) Coded bit-map; (c) Coded high- and low-mean images, and their difference image. . .	88
Picture V	(a) Composite satellite image; (b) Reconstructed image. .	89
Picture VI	Reconstructed image (4,8).	151
Picture VII	Reconstructed image (4,8).	151
Picture VIII	Original image.	175
Picture IX	Reconstructed image (12:1 compression).	176
Picture X	NTSC reconstructed image (14:1 compression).	177

List of Symbols

- BTC** — Block Truncation Coding.
- CSI** — Composite Satellite Imagery.
- DCT** — Discrete Cosine Transform.
- DPCM** — Differential Pulse Code Modulation.
- DSP** — Digital Signal Processing.
- HDTV** — High Definition Television.
- ISDN** — Integrated Services Digital Network.
- LUT** — Look Up Table.
- MBTC** — Multi-Level Block Truncation Coding.
- MSE** — Mean Square Error.
- NMSE** — Normalized Mean Square Error.
- PMSE** — Peak Mean Square Error.
- QBTC** — Quarternary Block Truncation Coding.
- R(D)** — Rate Distortion Function.
- SAR** — Synthetic Aperture Radar.
- SBMBTC** — Single Bit-Map Block Truncation Coding.
- SNR** — Signal to Noise Ratio.
- VQ** — Vector Quantization.
- VQA** — Codebook Adaptive Vector Quantization.
- VQU** — Universal Codebook Vector Quantization.

Chapter 1

INTRODUCTION

Digital image communications is a newly emerging field which has been, or will be, widely used in facimile, television, teleconferencing, teleconsultation, videotex, remote surveillance, high definition TV, interactive visual searching of picture databases. etc. However raster imagery, such as satellite remote sensing imagery, synthetic aperture radar imagery, medical X-ray imagery, HDTV imagery, and so on, usually has a huge data set. Presently, data communication networks simply do not have enough bandwidth to carry image data traffic. For example, today's state-of-the-art digital network – the Integrated Services Digital Network (ISDN) – which integrates voice and data services into one digital network, has a basic channel rate of 64 kbits/sec. To transmit a satellite image of size 1024 by 1024 pixels with 8 bits/pixel over a basic ISDN channel would require 131 seconds, which is too long for an on-line user. Thus, image data compression is a key requirement in image communication systems to increase the network or system throughput, and to save storage space.

There are three criteria that may be used to evaluate the performance of an image compression algorithm: *Computational Complexity*, *Compression Ratio*, and *Fidelity*. Lower computational complexity implies shorter coding delay and lower hardware expense. Higher compression ratio means greater channel efficiency and

reduced transmission time. High fidelity indicates low distortion, or coding errors, and high reliability. Unfortunately, all three can not be achieved simultaneously. From information theory, channel efficiency can be achieved by increasing the complexity and coding delay; whereas adoption of a less stringent fidelity requirement will reduce the amount of information to be transmitted.

Image data compression can be dated back to the 1950's. Since then, many successful coding algorithms, such as predictive coding, transform coding, vector quantization, entropy coding, etc., have been developed. Presently, computational complexity is the major bottle-neck in the implementation of image data compression. Today's computer techniques simply cannot support, or afford, the real-time application of some good coding algorithms, such as vector quantization and transform coding, through which high compression ratio and high fidelity can be achieved.

In this thesis, three new hybrid image coding techniques are developed: *BTC-VQ-DCT Hybrid Coding* [1], *Minimax Error Multi-Level Block Truncation Coding* [2,6] and a colour image data compression algorithm called *Single Bit-Map Block Truncation Coding* [5]. These new techniques achieve compromises between computational complexity, compression ratio, and fidelity. Each coding scheme contains more than one data compression stage. Different coding techniques are implemented in each step removing the redundancy more completely than that of conventional single stage coding.

BTC-VQ-DCT Coding combines the advantages of simple computation and edge preservation of *Block Truncation Coding* (BTC), the high fidelity and high compression ratio of *Adaptive Discrete Cosine Transform coding* (ADCT), and the high compression ratio and good subjective performance of *Vector Quantization* (VQ). The input image is first sub-divided into sub-blocks. Each sub-block undergoes BTC en-

coding, which truncates the input image into a bit-map and two reduced-sized grey-level images which are composed of “high-average” and “low-average” pixels from each sub-block. After that, the bit-map is decomposed into a set of vectors and, then, vector quantization is implemented. Since the bit-map vectors have a much smaller dynamic range than that of the original image, a *LUT-based vector “encoder”* may be designed to quickly vector quantize the bit-map. A *moment preserving scheme* is also used in the VQ. Adaptive DCT coding is implemented to code the high-average and low-average sub-images. A *residual error feedback* technique is used to increase the fidelity and reduce the bit rate. The overall computational complexity of BTC-VQ-DCT coding is much less than either DCT and VQ, while the fidelity performance is competitive. Good performance is demonstrated at a compression ratio of about 10:1.

Minimax Error Multi-Level Block Truncation Coding is a compression scheme that can preserve the spatial detail in composite imagery. The input image data is first adaptively quantized into a quarternary-, or ternary-level, plane and corresponding quantization level representation values. An adaptive quantizer-level allocation algorithm is introduced that minimizes the maximum quantization error in each block, while substantially reducing the computational complexity involved in allocation of the optimal quantization levels. VQ and predictive coding are implemented on the multi-level plane and quantization level representation values to further remove the residual inter- and intra-block correlations. The coding algorithm demonstrates very high fidelity, where no visual degradation is perceived, and very low computational complexity. The compression ratio is up to 6:1.

Single Bit-Map Block Truncation Coding is a colour image data compression algorithm. BTC is first implemented to encode a weighted trichromatic image. The

bit-map generated is then used to compress each of the R, G and B images separately, removing the inter-colour correlations. The weighted trichromatic image is generated by properly weighting the R, G and B images so that the MSE between the original and reconstructed R, G and B images is minimized. Another approach using a bit-wise majority rule on the R, G and B BTC bit-maps is also discussed. The algorithm preserves the spatial details in the image content and has low computational complexity. An 8:1 compression can be achieved. The truncated data is further compressed using adaptive DCT. Differential coding and error feedback techniques are also implemented to reduce the bit rate and improve the MSE performance. The total compression ratio is about 12:1. By exploiting human visual system characteristics, a 14:1 compression of NTSC colour TV signals can be achieved.

The rest of this thesis is organized as follows. Chapter 2 reviews image data compression algorithms. Chapter 3 discusses image fidelity measurements, including subjective and objective methods. Chapter 4 presents image data statistics including pixel correlations, entropies, and spectrum for different test images. Chapter 5 introduces the *BTC-VQ-DCT Hybrid Image Coding* algorithm. Chapter 6 presents the *Minimax Error Multi-level Block Truncation Coding*. Chapter 7 introduces the *Single Bit-Map Block Truncation Coding* which is a colour image data compression algorithm. Chapter 8 briefly summarizes the major results and provides suggestions for further research works.

The research described in thesis embodies the following novel and original ideas:
For BTC-VQ-DCT Hybrid Coding:

- DCT coding of high- and low-mean sub-images.
- Use of a moment preserving scheme to resolve the conflict of identical Hamming distances in binary VQ design.
- LUT-based binary VQ design.

For Minimax Error Multi-Level Block Truncation Coding:

- **Use of multi-level quantization in BTC.**
- **Use of a minimax error criterion in BTC.**
- **Invention of the threshold dilation scheme for allocation of the quantization levels for the minimax error quantizer.**
- **DPCM coding of quantization levels and lossless vector coding of the prediction errors.**
- **Lossless vector coding of the multi-level map.**
- **Implementation of VQ on the multi-level map.**

For Single Bit-Map Block Truncation Coding:

- **Use of one bit-map to quantize the R, G and B components to remove the inter-component correlations.**
- **Generation of the W-plane based on a minimum MSE guideline and using its BTC bit-map to quantize the colour components.**
- **Use of majority rule on BTC bit-maps to generate the quantization bit-map.**
- **DCT coding of quantization levels.**
- **Use of Y-component bit-map to quantize I and Q components for coding the NTSC colour TV signal.**

Chapter 2

REVIEW OF IMAGE CODING

Digital image compression based on redundancy removing techniques can be classified into two general categories: *lossless compression* and *lossy compression*. *Lossless compression* removes, or reduces, the pixel correlation in such a way that it is possible to recover exactly the original data. *Lossy compression* converts the input data into a smaller set from which a “close approximation” to the original image can be reconstructed [123].

2.1 Lossless Compression

Lossless image coding is, usually, used to compress *graphics*, which are those images that contain large areas of constant intensity and long uniform level segments (runs); for example, weather charts, documents, hand drawings, bar charts, pie charts, etc. They can be bi-level or multi-level. The pixel correlations of graphics are quite high. Many lossless coding algorithms have been developed to compress the graphics [43-45], such as, *Huffman coding*, *run-length coding*, *block coding*, *contour coding (chain coding)*, *rectangular coding*, *quadtrees representation*, and *predictive differential quantization (PDQ)*.

2.1.1 Huffman Coding

From Shannon's first theorem, or the lossless coding theorem, the average bit rate R for encoding a source can be as small as, but no less than, the source entropy H [8, 124]. Huffman [9] introduced a practical procedure to implement the Shannon coding theorem for encoding a statistically independent source. Huffman coding results in a variable length code in which the number of bits for a source sample, i.e., pixel, varies inversely with the corresponding source probability.

The design of a Huffman code depends upon the source statistics. If source statistics change, a new code is required to ensure the minimum bit rate. In adaptive Huffman coding [10, 11, 12], the code set is modified based upon frequency information derived from the most recent samples, e.g., Lempel-Ziv-Welch algorithm [37, 123, 127].

2.1.2 Run-Length Coding

Run-length coding [14-20] is one of the most extensively studied and simplest methods of one-dimensional compression. Each raster line is broken into a series of segments of constant level, i.e., runs. The codes used to represent the runs are determined by the distribution of the runs. Fewer bits are assigned to shorter runs than to longer ones. Some examples are the CCITT code for document coding [17] and the DIFAX code for weather chart coding [18].

2.1.3 Block Coding

Block coding [21] partitions the graphics into disjoint rectangular blocks. The blocks with uniform level are coded using fewer bits. Block coding can be one-dimensional or two-dimensional, and the size of the block can be adaptive to the local contents.

2.1.4 Contour Coding

In *contour coding*, or chain coding [22, 23], one initial boundary coordinate is stored and the boundary around a region is then followed back to the first pixel, and this is repeated for each separate region.

2.1.5 Rectangular Coding

In *rectangular coding* [24, 25], graphics are decomposed into a set of rectangles, and coded. Four parameters are used to represent the rectangular: the X, Y coordinates of one vertex and the width and height of the rectangular.

2.1.6 Quadtree Representation

Quadtree representation [26-28] is usually used for graphics storage, i.e. in an image data base. It is a tree structure in which successively deeper levels represent successively finer subdivisions of the graphics area with each level having four subdivisions.

2.1.7 Predictive Differential Quantization

Predictive Differential Quantization (PDQ) [29, 30] makes use of the correlation between scans, and can be considered as an extension of run-length coding to two dimensions.

All the techniques mentioned above are, originally, designed to code bi-level images. But they can be extended to code multi-level images. For example, they can be implemented to code the individual bit planes of an multi-level image [31].

2.2 Lossy Compression

Lossy coding is usually implemented to compress digitized continuous-tone images. It can be further sorted into three classes: *spatial coding*, *transform coding*, and *hybrid coding*. Coding in the spatial domain involves the direct manipulation of the sampled image data to remove inherent redundancies. Transform coding involves an energy preserving mapping of the original image data such that maximum information is preserved in a minimum number of samples or coefficients. Hybrid coding employs a combination of both techniques, usually transform coding followed by predictive coding of coefficients. *Vector quantization* is a recently developed technique for data compression, which, based on Shannon's rate distortion theory, directly removes redundancies between groups of pixels, i.e., vectors. Vector quantization can be implemented in either spatial or transform domains.

Figure 2.1 lists some of the techniques [4] and Table 2.1 summarizes reported compression performance [13, 14, 32-38]. This Section reviews *lossy* compression techniques with emphasis upon *block truncation coding*, *adaptive cosine transform coding* and *vector quantization*, which will be implemented in later chapters.

2.2.1 Spatial Coding

Spatial coding is usually simple to implement both in terms of memory requirements and number of operations. It is quite sensitive to changes in the data statistics and channel error effects can accumulate, severely degrading image quality. The fidelity of spatial coding can be very high, while the compression ratio is moderate.

Differential Pulse Code Modulation

Differential Pulse Code Modulation (DPCM) [39], or *predictive coding*, is based on the

Method	Typical Average Rates bits/pixel	Comments
<u><i>Spatial Coding</i></u>		
Block Truncation Coding	1–2	Very simple implementation.
Laplacian Pyramid Coding	1.5–2	Simple computation and suited to progression transmission.
Intraframe DPCM	2–3	Simple implementation.
Intraframe Adaptive DPCM	1–2	Same as above.
Interframe DPCM†	1–1.5	Same as above.
Interframe Adaptive DPCM†	0.5–1	Same as above.
Vector Quantization	0.4–1.5	Most complex
<u><i>Transform coding</i></u>		
Intraframe	1–1.5	Highly complex.
Intraframe Adaptive	0.5–1	Same as above.
Interframe†	0.5–1	Same as above.
Interframe Adaptive†	0.1–0.5	Most complex.
<u><i>Hybrid Coding</i></u>		
Intraframe	1–2	Complex.
Intraframe Adaptive	0.5–1.5	Same as above.
Interframe†	0.5–1	Same as above.
Interframe Adaptive†	0.3–0.5	Same as above.

Table 2.1: Comparison of image compression methods.

† Interframe coding requires more than one image in a series.

Figure 2.1: Image compression methods.

fact that the gray level values of adjacent pixels in most images are highly correlated. That is, if a pixel at $(x-1, y)$ has a certain gray level then the adjacent pixel at (x, y) is likely to have a similar gray level. Whereas the gray level may range over 256 values (8 bits/pixel), most differences between adjacent pixel values in natural scene images lie within a range of about 20 gray level values [14].

The philosophy underlying *predictive coding* is to remove mutual redundancy between successive image pixels by quantizing only the difference between the actual gray level value of a pixel and its optimally predicted value. The predicted value is computed from previously coded pixels.

The key element in a DPCM system is the predictor. The number of previous pixels used by the predictor determines its order. First order prediction is by definition one-dimensional, but higher order predictors may be two-dimensional. A one-dimensional predictor relies only on previous pixels in the same scan line, while two-dimensional predictors may use pixels from previous adjacent rows. The higher

the predictor order is, the better the prediction will be. On the other hand, higher order prediction means more computation and memory access.

The prediction gain characteristics show early saturation of performance with increasing predictor order; the best gain is found with third order methods [40]. High order two-dimensional predictors are used not for better SNR, or MSE [41], but for better subjective performance [42] and for better tolerance to transmission error effects [43].

For a source image with a separable autocorrelation function, to be discussed in Chapter 4, a two-dimensional predictor has about 10 dB higher prediction gain than a first order predictor. For an isotropic source, two-dimensional predictors have a gain of no more than 3 dB [13] over one-dimensional predictors.

DPCM coding is quite sensitive to changes in input data statistics [3]. If the input data is not stationary, adaptive techniques are used to trace changes in the local statistics. Adaptive techniques generally fall into one of the two classes: an adaptive predictor with a fixed quantizer, or an adaptive quantizer with a fixed predictor [44]. In the adaptive approaches, the parameters of the predictors, or the quantizers, are continuously modified to keep up with the changing statistics of the input data.

The advantages of DPCM are high fidelity, computational simplicity, low memory access and robustness to transmission errors. The major drawback is that the compression ratio is not very high.

Block Truncation Coding

Block Truncation Coding (BTC) [45-49] uses a two-level moment-preserving quantizer that adapts to local properties of the image. It has the advantages of preserving single pixel resolution and edges and having low computational complexity. The original

algorithm preserves the block mean and the block standard deviation [45, 46]. Other choices of the moments result either in less MSE or less computational complexity [48, 49].

In BTC, the image is, first, partitioned into non-overlapping $m \times m$ pixel blocks, in which there will likely be fairly high pixel-to-pixel correlation [45-49].

In Delp and Mitchell's algorithm [45], each pixel block is coded, individually, into a two level signal. The levels for each block are chosen such that the first two moments are preserved. Let $l = m^2$ and let $x(i)$, where $i = 1, 2, \dots, l$, be the pixel values in an original block. Then, the first and second sample moments and sample variance are,

$$\bar{X} = \frac{1}{l} \sum_{i=1}^l x(i), \quad (2.1)$$

$$\overline{X^2} = \frac{1}{l} \sum_{i=1}^l x^2(i), \quad (2.2)$$

$$\overline{\sigma^2} = \overline{X^2} - \bar{X}^2 \quad (2.3)$$

respectively. A threshold, X_{th} , and two output levels, a and b such that

$$\text{if } x(i) \geq X_{th}, \quad \text{output} = b,$$

$$\text{if } x(i) < X_{th}, \quad \text{output} = a,$$

$$\text{for } i = 1, 2, \dots, l. \quad (2.4)$$

Set $X_{th} = \bar{X}$ and let q indicate the number of $x(i)$'s greater than $X_{th} (= \bar{X})$, then, to preserve $\bar{X}$ and $\overline{X^2}$,

$$l\bar{X} = (l - q)a + qb, \quad (2.5)$$

and

$$l\overline{X^2} = (l - q)a^2 + qb^2, \quad (2.6)$$

must hold. Solving for a and b :

$$a = \bar{X} - \sigma \sqrt{\frac{q}{l-q}} \quad (2.7)$$

$$b = \bar{X} + \sigma \sqrt{\frac{l-q}{q}}. \quad (2.8)$$

Each block is, then, described by the values of $\bar{X}$, σ and a $m \times m$ bit plane, which has the same dimensions as the original block, consisting of 1's and 0's indicating whether pixels are above or below X_{th} . These spatial resolution data are, then, transmitted together with the mean and variance of the original block. For a 4×4 block size, assigning 8 bits each to $\bar{X}$ and σ results in a bit rate of 2 bits/pel. The receiver reconstructs the image block by calculating a and b from Equation 2.7 and 2.8 and assigning these values to pixels in accordance with the code in the bit plane.

The Absolute Moment BTC described in [48] quantizes the pixels in each block into high-mean and low-mean, which will preserve the Mean Absolute Moment or minimize the Mean Absolute Error in each block.

$$\bar{X} = \frac{1}{l} \sum_{i=1}^l x(i), \quad (2.9)$$

is used as the threshold for the 1-bit quantizer, where $x(i)$ is the value of i th pixel in the block and l is the number of pixels in a block, i.e., $m \times m$. If

$$x(i) \leq \bar{X} \text{ or } x(i) > \bar{X}, \quad (2.10)$$

$\bar{X}_L$ or $\bar{X}_H$, respectively, is assigned to the quantizer output levels, where

$$\bar{X}_L = \frac{1}{l-q} \sum_{x(i) \leq \bar{X}} x(i), \quad (2.11)$$

$$\bar{X}_H = \frac{1}{q} \sum_{x(i) > \bar{X}} x(i), \quad (2.12)$$

and q is the number of pixels having value higher than $\bar{X}$. Hence, the outputs of the BTC for each block include two numbers which specify the high-mean, $\bar{X}_H$, and low-mean, $\bar{X}_L$; and a bit-map which specifies the state, $\bar{X}_H$ or $\bar{X}_L$, of each output pixel. This algorithm has the potential of low computational complexity with no multiplications involved and real-time implementation has been reported[50].

Halverson et al. [49] generalized BTC to include a family of moment preserving quantizers with the potential of improving performance. It has been proved that the selection of the quantizer threshold and levels can be generalized for much larger classes of pairs and triples of sample moments, respectively. Performance improvements were demonstrated from the standpoint of peak signal to noise ratio. However, the application of the BTC algorithm for the higher moments case is often hampered by overflow problems and increased computational complexity.

A major advantage of block truncation coding is that the coding algorithm is very simple and decoding is trivial. In addition, it preserves abrupt edges with no blurring. However, the characteristics of any particular block may be different from those surrounding it so that coding parameters will be different from block to block. This can lead to discernible block type error. Hence, a two-level quantizer is simply not able to code a highly busy image with a satisfactory fidelity [51]. To achieve high fidelity, as described in Chapter 5 we implement a multi-level quantizer and a min-max quantization error criterion [2] is developed to allocate the quantizer levels which will obtain a close replica of the original image and which is robust to the pixel statistical distribution within each pixel block.

Laplacian Pyramid Coding

In *Laplacian Pyramid coding* [52-54], the image is first low-pass filtered using a weighted convolving function of a roughly Gaussian profile to produce a so-called “Gaussian” image. This Gaussian image is then subtracted from the original to produce the “Laplacian” image, a difference image with little pixel-to-pixel correlation. This Laplacian image has lower variance and entropy since most pixel gray levels are tightly and symmetrically distributed about the zero level [52]. Compression is achieved owing to the lower entropy in the Laplacian image, and by coarsely quantizing it. To reconstruct the original image both the Gaussian and Laplacian images must be added together, which appears to permit no compression. However, the Gaussian may be represented at a low sample density, since it was a low pass version of the original image. Therefore, in practice the Gaussian image is subsampled and both the subsampled Gaussian and the Laplacian image must be transmitted. At reconstruction, the Gaussian image is expanded again to its original area by interpolation before addition to the Laplacian image. The above steps are then applied repeatedly to each successive low pass image to form a pyramid of Laplacian image planes, each one smaller than the previous. The pyramid production stops when the final Gaussian shrinks to the size of the convolving filter (typically 5×5 pixels). The Laplacian can be viewed as a series of bandpassed images. The characteristics of the *Human Visual System (HVS)* can be used to reduce the bit rate in those planes; lower planes represent higher spatial frequencies which the HVS cannot perceive, and thus can be coded with fewer levels. The disadvantages of this coding algorithm are computational complexity, large memory requirement, and transmission error sensitivity.

2.2.2 Transform Coding

Transform coding achieves relatively large compression ratios compared to spatial coding. Generally, any distortion due to quantization and channel errors gets distributed over the image block during the inverse transformation [32]. From an overall visual perspective, this is less objectionable than the errors in spatial coding which are distributed locally [58]. Moreover, transform coding is not sensitive to fluctuations in data statistics.

An optimal orthogonal transform should completely decorrelate the transform coefficients and, therefore, minimize the geometric mean of the variance of the transform coefficients [13]. It can be shown [13] that minimizing the geometric mean of the variance of the transform coefficients corresponds to minimizing the reconstruction error variance in transform coding.

An orthogonal transform has following properties [13]:

1. The average energy of the transformed image is equal to the average energy of the input image, i.e.,

$$\frac{1}{N \times N} \sum_{i,j}^{N-1} \sigma_{ij}^2 = \frac{1}{N \times N} \sum_{u,v}^{N-1} \sigma_{uv}^2 = \sigma^2,$$

where (i, j) and (u, v) represent the spatial and transform domain, respectively.

2. The average reconstruction error variance is equal to the average quantization error variance in the transform domain, i.e.,

$$\sigma_r^2 = \frac{1}{N \times N} \sum_{i,j}^{N-1} \sigma_{r,ij}^2 = \frac{1}{N \times N} \sum_{u,v}^{N-1} \sigma_{q,uv}^2 = \sigma_q^2,$$

where $\sigma_{r,ij}^2$ is the reconstruction error variance in spatial domain, and $\sigma_{q,uv}^2$ is the quantization error variance in transform domain.

Figure 2.2: Mean square error performance of image transforms as a function of block size.

It can be seen from Figure 2.1 that there are many kinds of transform schemes. Figure 2.2 contains a plot of the mean square error performance as a function of block size for various transformations [61,64].

The Karhunen-Loeve Transform (KLT), which is based upon the statistical properties of an image [35], is known to be optimal with respect to the variance distribution [59] and rate distortion function [60]. The problem of implementing the KLT is the computation complexity, which requires $2N^3$ multiplications and additions, where N is the transform block size. There is no general algorithm that enables its fast

computation [55]. This problem gets simpler when using suboptimal deterministic transforms [13] such as the discrete Fourier transform (DFT), discrete cosine transform (DCT), Walsh-Hadamard transform (WHT), Harr transform (HT), and the slant transform (ST). It has been shown [62] that the DCT performs very closely to the KLT when the first-order correlation coefficient lies in the interval (0.5,1), even when the block size is small (Figure 2.2). The DCT can be computed via a fast algorithm [55, 61, 63]. These properties have made the DCT a useful substitute for the KLT in image coding [32, 62] since many images have high inter-pixel correlation.

One Dimensional DCT and Inverse DCT

The one dimensional DCT of a discrete function $f(i), i = 0, 1, \dots, N - 1$, is defined as [55]

$$\begin{aligned} F(0) &= \frac{\sqrt{2}}{N} \sum_{i=0}^{N-1} f(i) \\ F(u) &= \frac{2}{N} \sum_{i=0}^{N-1} f(i) \cos \left[\frac{(2i+1)u\pi}{2N} \right]; \\ u &= 0, 1, \dots, N-1. \end{aligned} \quad (2.13)$$

where $F(u)$ is the u th DCT coefficient. It can be seen from Equation 2.13 that the DCT coefficients $F(u)$ are real numbers for real $f(i)$ and they correspond, respectively, to the N frequencies $\omega_i = 2\pi k/2N$, $k = 0, 1, \dots, (N-1)$, which are equally spaced around the upper half of the unit circle [56]. It is also worthwhile noting that the set of basis vectors $\{1/\sqrt{2}, \cos((2i+1)u\pi)/(2\pi)\}$ is actually a class of discrete Chebyshev polynomials. This can be seen by recalling that Chebyshev polynomials can be defined as [57]

$$\hat{T}_0(\xi_p) = \frac{1}{\sqrt{2}}$$

$$\hat{T}_u(\xi_p) = \cos(u \cos^{-1} \xi_p), \quad p, u = 1, 2, \dots, N. \quad (2.14)$$

where $\hat{T}_u(\xi_p)$ is the u th Chebyshev polynomial.

Now, in Equation 2.14, ξ_p is chosen to be the p th zero of $\hat{T}_N(\xi_p)$, which is given by [57]

$$\xi_p = \cos \frac{(2p-1)\pi}{2N}, \quad p = 1, 2, \dots, N. \quad (2.15)$$

Substituting Equation 2.15 in Equation 2.14, one obtains the set of Chebyshev polynomials

$$\begin{aligned} \hat{T}_0(p) &= \frac{1}{\sqrt{2}} \\ \hat{T}_u(p) &= \cos \frac{(2p-1)u\pi}{2N}, \quad p, u = 1, 2, \dots, N. \end{aligned} \quad (2.16)$$

From Equation 2.16 it follows that the $\hat{T}_u(p)$ can equivalently be defined as

$$\begin{aligned} T_0(i) &= \frac{1}{\sqrt{2}} \\ T_i(u) &= \cos \left[\frac{(2i-1)u\pi}{2N} \right], \quad u = 1, 2, \dots, N-1, \\ &\quad i = 0, 1, \dots, N-1. \end{aligned} \quad (2.17)$$

Comparing Equation 2.17 with Equation 2.13 we conclude that the basis member $\cos((2i+1)k\pi)/(2N)$ is the u th Chebyshev polynomial $T_u(\xi)$ evaluated at the m th zero of $T_M(\xi)$.

Again, the inverse DCT (IDCT) is defined as

$$\begin{aligned} f(i) &= \frac{1}{\sqrt{2}} F(0) + \sum_{u=0}^{N-1} F(u) \cos \left[\frac{(2i+1)u\pi}{2N} \right]; \\ &\quad i = 0, 1, \dots, N-1. \end{aligned} \quad (2.18)$$

We note that applying the orthogonal property [57]

$$\sum_{i=0}^{N-1} T_u(i) T_v(i) = \begin{cases} N/2, & u = v = 0 \\ N/2, & u = v \neq 0 \\ 0, & u \neq v \end{cases} \quad (2.19)$$

to Equation 2.18 yields the DCT in Equation 2.13.

DCT vs. KL

The motivation for using the DCT is that it can be demonstrated [55] that its basis set provides a good approximation to the eigenvectors of the class of Topelitz matrices defined as

$$\psi = \begin{bmatrix} 1 & \rho & \rho^2 & \dots & \rho^{M-1} \\ \rho & 1 & \rho & \dots & \rho^{M-2} \\ \vdots & & & \ddots & \vdots \\ \rho^{M-1} & \rho^{M-2} & \rho^{M-3} & \dots & 1 \end{bmatrix}, \quad 0 < \rho < 1. \quad (2.20)$$

For the purposes of illustration, the eigenvectors of ψ for $M = 8$ and $\rho = 0.9$ are plotted, in Figure 2.3, against

$$\left\{ \frac{1}{\sqrt{2}}, \cos \left[\frac{(2i+1)u\pi}{16} \right], u = 1, 2, \dots, 7, i = 0, 1, \dots, 7 \right\} \quad (2.21)$$

which constitute the basis set for the DCT. The close resemblance (aside from the 180° phase shift) between the eigenvectors and the set defined in Equation 2.21 is apparent.

DCT vs. DFT

Considering a DFT coding system, let $\{f(i); i = 0, 1, \dots, N-1\}$ be one block of data of length N as depicted in Figure 2.4(a) and $\{\mathcal{F}(u); u = 0, 1, \dots, N-1\}$ be its Fourier transform. If the total number of bits used to encode $\mathcal{F}(u)$ is very low, the quantization noise is a combination of both multiplicative and additive effects: i.e. [56],

$$\hat{\mathcal{F}}(u) = G_f(u)\mathcal{F}(u) + E_f(u) \quad (2.22)$$

Figure 2.3: Eigenvectors of (8×8) Toeplitz matrix ($\rho = 0.9$) and basis vectors of the DCT [55].

where $G_f(u)$ and $E_f(u)$ represent the noise components; $\hat{\mathcal{F}}(u)$ is the quantized Fourier transform. The decoding procedure then leads to the result

$$\hat{f}(i) = g_f(i) \otimes f(i) + e_f(i) \quad (2.23)$$

where $g_f(i)$ is the inverse transform of $G_f(u)$ and $\otimes$ denotes the *circular* convolution of $f(i)$ with $g_f(i)$. The circular convolution results in an exchange of energy between the left and right boundaries of $f(i)$ (i.e., aliasing in time), as illustrated by the arrows in Figure 2.4(b). These end effects can lead to very undesirable block type errors on the reconstructed image.

A well known solution to the above aliasing problem is to pad $f(i)$ with zeros (equal to the number of samples of $g_v(i)$ minus 1) and use a larger transform. Unfortunately, for transform coding, $g_f(i)$ and $e_f(i)$ are not well-defined time limited quantities, and increasing the transform size to be larger than the block size will

Figure 2.4: DFT and DCT spectrum [56].

reduce the average number of bits/sample across the block which further compounds the problem.

The DCT reduces the above end-effect problems between the left and right boundaries while still keeping a minimum spectrum redundancy (i.e., N transform coefficients for N data points). As pointed out by Chen and Fralick [65], the DCT is closely related to the $2N$ -point DFT of a sequence $y(i)$, which is formed from the N -point sequence $f(i)$ by defining

$$y(i) = \begin{cases} \frac{1}{2} f(i) & i = 0, 1, \dots, N-1 \\ \frac{1}{2} f(2N-1-i) & i = N, N+1, \dots, 2N-1. \end{cases} \quad (2.24)$$

The sequence $y(i)$ is depicted by the shaded region in Figure 2.4(c). The $2N$ point DFT of $y(i)$ leads to

$$\begin{aligned} Y(u) &= \sum_{i=0}^{2N-1} y(i) e^{-j2\pi ui/2N} \\ &= e^{j(\pi u/2N)} \sum_{i=0}^{N-1} f(i) \cos [(2i+1)\pi u/2N]. \end{aligned} \quad (2.25)$$

By comparison of Equation 2.13 with 2.25, it can then be seen that the DCT of $f(i)$ can be obtained from $Y(u)$ according to the relation

$$F(u) = c(u)e^{-j\pi u/2N} Y(u) \quad u = 0, 1, \dots, N-1. \quad (2.26)$$

where

$$c(u) = \begin{cases} \frac{1}{\sqrt{2}} & u = 0 \\ 1 & u = 1, 2, \dots, N-1. \end{cases}$$

It follows that all the N DCT coefficients can be computed via the FFT using a $2N$ -point FFT algorithm [55, 56].

The above interpretation not only leads to a fast computation algorithm, but also provides insight into the properties of the DCT. Because of the association of the DCT with the $2N$ -points of the symmetric sequence $y(i)$ (symmetric about a “half sample”), it can be seen that quantizing $F(u)$ is, in effect, equivalent to quantizing $Y(u)$. Therefore, as in the DFT analysis, we can write

$$\hat{Y}(u) = G_y(u)Y(u) + E_y(u) \quad (2.27)$$

and

$$\hat{y}(i) = g_y(i) \otimes y(i) + e_y(i) \quad (2.28)$$

Figure 2.4(d) depicts the end effects between the left and right boundaries of $\hat{y}(i)$ due to the circular convolution of $y(i)$ with the multiplicative component of quantization noise $g_f(i)$. In terms of sequence $f(i)$, however, it is seen that these interactions are now localized, and there is no longer an exchange of energy between the left and right boundaries of $f(i)$. Thus, the DCT produces less noticeable boundary effects than DFT in transform coding applications. This is an additional advantage of the DCT besides the fact that it is “close” in performance to the KL transform.

Figure 2.5: Fast discrete cosine transform flowchart [66]

$$N = 4, 8, 16, C_i = \cos i, S_i = \sin i.$$

Fast Computation Algorithm of DCT

Examination of Equation 2.13 reveals that direct computation of a one dimensional DCT requires N^2 multiplications and $N \times (N - 1)$ additions. This is very inefficient. A method of implementing the DCT through a double sided $2N$ -point FFT algorithm was introduced in the last section, but it involves complex arithmetic throughout the computation.

Fortunately, there is a recently developed Fast Discrete Cosine Transform (FDCT) algorithm, which involves only real operations and the number of computation steps is less than $1/6$ of those of the conventional 2-sided FFT algorithm [63,65,66].

Figure 2.5 is the signal-flow graph of the FDCT for $N = 4, 8$, and 16 . Note that the input samples are in natural order from top to bottom. For every N , the output transform coefficients are in bit-reversed order.

The signal-flow graph in Figure 2.5 represents the forward transform but the coef-

Figure 2.6: Comparison of computational steps for conventional FDCT, FFT and FDCT [66]: (a) Additions. (b) Multiplications.

ficients have not been normalized. To obtain the normalized N -point DCT samples, the appropriate terminal points should each be multiplied by $2/N$.

The FDCT signal-flow graphs are also bidirectional; i.e., the inverse transform may be computed by introducing the sequence $F(u)$ at the output and recovering the sequence $f(i)$ at the input. This follows the fact that every butterfly pair in the signal-flow graph is a unitary matrix. The normalization factor for the inverse transform is $(N/2)$.

The conventional fast approach to computing the DCT utilizing a 2-sided FFT

takes $2N \log 2N$ complex additions and $N(\log 2N + 1)$ complex multiplications (which are equivalent to $6N \log 2N + 2N$ additions and $4N(\log 2N + 1)$ real multiplications). From the signal-flow graphs, the FDCT method needs only $(3N/2)(\log_2 N - 1) + 2$ real additions and $N \log_2 N - 3N/2 + 4$ real multiplications. For the purposes of comparison, Figure 2.6 [66] plots the number of computation steps versus the transform sizes. It can easily be seen that the new FDCT takes less than 1/6 as many steps as that of the 2-side FFT approach. Also plotted in the figure are the number of steps for the FFT. Note the FDCT costs only 1/3 of the computations of the FFT!

Two Dimensional DCT

The one dimensional DCT discussed in the previous section can easily be extended to two dimensions, as

$$F(u, v) = \frac{4 c(u) c(v)}{N^2} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \cos \frac{(2i+1)u\pi}{2N} \cdot \cos \frac{(2j+1)v\pi}{2N} \quad (2.29)$$

where $f(i, j)$ is a digitized image. It can be seen that the two dimensional DCT kernel is separable so that it can be obtained by sequential row and column one-dimensional transforms.

Figure 2.7 shows some 8×8 image arrays and their DCT coefficients.

Statistical Properties of Two Dimensional DCT

The image statistical information plays an important role in developing an efficient coder. Here we present some of the statistical properties which are useful in devising the adaptive coder.

$$\begin{array}{c} \mathbf{f}(i,j) \\ \left| \begin{array}{cccccccc} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{array} \right| \Leftrightarrow \left| \begin{array}{cccccccc} 2.00 & 0.00 & 9.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \end{array} \right| \\ \text{(a)} \end{array}$$

A. Mean and Variance

Suppose that the image pixel $f(i, j)$ is a sample of a random process with mean m . Then $E\{f(i, j)\}$ can be replaced by m and the expected value of the DCT samples can be expressed as

$$E\{F(u, v)\} = \frac{4m c(u) c(v)}{N^2} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \cos \frac{(2i+1)u\pi}{2N} \cdot \cos \frac{(2j+1)v\pi}{2N} \quad (2.30)$$

or

$$E\{F(0, 0)\} = 2m$$

and

$$E\{F(u, v)\} = 0; \quad (u, v) \neq 0. \quad (2.31)$$

It can be seen, from Equation 2.31, that $F(0, 0)$ is proportional to the mean m which represents the image brightness or *dc* component, while $F(u, v)$ $(u, v) \neq 0$ represent the *ac* components.

With the mean denoted as above the variance of the transform samples can be computed from

$$\sigma^2(0, 0) = E\{[F(0, 0)]^2\} - 4m^2$$

and

$$\sigma^2(u, v) = E\{[F(u, v)]^2\}; \quad (u, v) \neq 0. \quad (2.32)$$

B. Statistical Description of DCT Coefficients

It would be desirable to know the probability density of DCT samples for the image transform. Unfortunately, this result is not easily obtained because the original image probability density is not usually well defined, and, also, the DCT operation is mathematically complex. However, each DCT sample is formed from the cosine-weighted sum of all the pixels in the original image. One can evoke qualitative

arguments, based upon the central limit theorem of statistics, that the probability density of DCT samples tends to be Gaussian [61,64]; i.e.,

$$p_{(0,0)}(x) = \frac{1}{\sqrt{2\pi} \sigma(0,0)} \exp \left\{ -\frac{(x - 2m)^2}{2\sigma^2(0,0)} \right\}$$

and

$$p_{(u,v)}(x) = \frac{1}{\sqrt{2\pi} \sigma(u,v)} \exp \left\{ -\frac{x^2}{2\sigma^2(u,v)} \right\}; \quad (2.33)$$

$(u,v) \neq 0.$

However, the probability fitting tests on the empirical data [68-70] indicates that *Laplacian* distribution seems the best model for non-dc coefficients; i.e.,

$$p_{(u,v)}(x) = \frac{1}{\sqrt{2} \sigma(u,v)} \exp \left\{ -\frac{\sqrt{2}|x|}{\sigma(u,v)} \right\}; \quad (2.34)$$

$(u,v) \neq 0.$

Adaptive DCT Coding

In a DCT coding system, the input image is first partitioned into non-overlapping $N \times N$ blocks, usually $N = 16$. Then, two-dimensional DCT are conducted on each block. Because of the nonstationary nature of the image data, the information content of the transformed samples changes from one block to another and this affects the coding performance. The performance of the coding systems could be improved substantially by adapting them to changes in image statistics [32, 44].

There are many DCT coding techniques [4, 64, 70, 72]. Among them, Chen and Pratt's [70, 71] seems the most successful one. All of the adaptive algorithms are based on the fact that images are composed of objects which are in turn a combination of edges and areas of varying degrees of detail. Within each block, the degree of picture detail changes drastically. An efficient encoding of these blocks requires more bits for

Figure 2.8: Block diagram of scene adaptive DCT coding system [70].

areas of high detail and fewer bits for areas of low detail. This results in a variable average bit rate from block to block, but gives a better utilization of the total bits over the entire ensemble of image blocks. A buffer and some buffer control schemes must be used to transmit the encoder outputs over a fixed rate channel.

Figure 2.8 is the block diagram of Chen and Pratt's scene adaptive DCT coding system [70, 71]. The input image undergoes a two dimensional DCT of 16×16 pixel blocks and the resulting transform coefficients within each block are stored in a zigzag scan sequence of Figure 2.9. Then, the transform coefficients are thresholded by an initial threshold value. Any coefficient that is less than the threshold is truncated to zero. The non-zero coefficients are scaled according to a feedback parameter from the output buffer. The scaled coefficients are quantized and Huffman coded, where their magnitudes and addresses in zigzag scan are coded by a magnitude LUT and a runlength LUT respectively. The results are fed into a channel buffer. The buffer status and input rate are monitored to generate the coefficient scaling factor and threshold value, which will prevent the buffer over/under flow. Good MSE and subjective performances are reported at 0.4 bits/pel [70].

Figure 2.9: Zigzag scan of DCT coefficients [70].

2.2.3 Hybrid Transform/Predictive Coding

Hybrid coding refers to techniques which combine transform coding and predictive coding techniques [73]. Typically, a two-dimensional image is unitarily transformed in one of its dimensions to obtain a sequence of one dimensional, independent sequences. Each of these sequences is then coded independently by a one dimensional predictive technique such as Differential Pulse Code Modulation (DPCM). This technique combines the advantage of the hardware simplicity of DPCM and the robust performance of transform coding [32,40]. Its performance lies between transform and predictive coding.

2.2.4 Vector Quantization

Vector quantization (VQ) is a recently developed technique for data compression [4-77]. Shannon's *rate distortion theory* [78-82] indicates that better performance can be achieved by coding vectors instead of scalars. Generally, VQ has the following major steps:

1. Vector Formation;

2. Training Sequence Formation;
3. Codebook Generation;
4. Quantization;
5. Decoding.

Vector Formation

The first step in image vector quantization is decomposition of the image into a set of vectors, which is a sub-space of an M-dimensional Euclidean space. The vectors can be defined in a number of ways; for example, as the intensity values of a spatially contiguous group of pixels [83], as orthogonal transformed components [84-86], as prediction errors [87], or as mean/residual errors [88,89].

Training Sequence Formation

There are two methods for training sequence and codebook generation: universal and adaptive.

In *universal codebook VQ* (VQU), the vectors for the training sequence are drawn from many different kinds of images [1,76,83,91,92]. The resulting universal codebook can then be fixed at both the encoder and decoder, and, therefore, there is no overhead required for transmitting or storing the codebook. The problems with universal VQ are that images outside of the training sequence may not always be represented well and, to ensure a good fit for all the images, a codebook with many codewords are needed which increases the bit rate. One proposed modification is *classified vector quantization* [100], where the vectors in a training sequence are divided into a finite set of classes based upon some features, such as edge and shade types. For each class,

a separate codebook is generated. In quantization, the class for each input vector is first determined and the vector is then encoded with only the codebook specifically designed for the class, thus, significantly reducing the searching complexity. Aravind and Gersho [101] exploit the memory of the input image to determine the state of the input vector, where the state of each input vector is predicted from the previous reproduction vectors; thus, there is no side information required for indicating the choice of codebook, which reduces the bit rate.

In the *adaptive codebook VQ* (VQA), the training sequence and the codebook are adapted to each input image [86, 90]. For each input image, a subset of vectors formed from the image is taken as the training sequence and a codebook is generated. A further refinement is to break up the input image into subimages, and design a local codebook for each subimage [102]. The local codebooks better match the local statistics, thus, smaller codebook can be used. However, an overhead is required for transmitting the codebook for each input image.

Codebook Generation

The guideline in designing a codebook is to obtain a quantizer consisting of N_c reproduction vectors, such that it minimizes the mean distortion. Optimality is achieved if there is no other quantizer that can achieve the minimum mean distortion. Lloyd [93] proposed an iteration nonvariational technique for the design of scalar quantizers. Recently, Linde *et al.* [94] extended Lloyd's basic approach to the general case of designing vector quantizers, which is commonly called the *LBG algorithm*.

Let the mean distortion be:

$$D(x, q(x)) = \frac{1}{N} \sum_{i=0}^{N-1} d^2(x_i, \hat{x}_i). \quad (2.35)$$

For an unknown distribution training sequence, the LBG algorithm contains the following steps [84],

1. *Initialization:* Assuming an initial set of N_c reproduction alphabet (codebook) $\hat{A}_0$, a training sequence $(x_j; j = 1, \dots, n-1)$, a distortion threshold $\epsilon \geq 0$, and the number of iterations $m = 0$.
2. *Partition:* Given $\hat{A}_m = (v_i; i = 1, \dots, N_c)$, search for the minimum distortion partition $P(\hat{A}_m) = (S_i; i = 1, \dots, N_c)$ of the training sequence: $x_j \in S_i$ if $d(x_j, v_i) \leq d(x_j, v_l)$, for all l . Calculate the average distortion

$$D_m = D[(\hat{A}_m, P(\hat{A}_m))] = \frac{1}{n} \sum_{j=0}^{n-1} \min_{v \in \hat{A}_m} d^2(x_j, v). \quad (2.36)$$

3. *Codebook Updating:* Find the optimal reproduction alphabet (codebook)

$$\hat{x}(P(\hat{A}_m)) = (\hat{x}(S_i); i = 1, \dots, N_c)$$

for $P(\hat{A}_m)$, where

$$\hat{x}(S_i) = \frac{1}{\|S_i\|} \sum_{j: x_j \in S_i}^m x_j. \quad (2.37)$$

4. *Termination:* if $(D_{m-1} - D_m)/D_m \leq \epsilon$, stop the iteration with $\hat{A}_m$ as the final codebook; otherwise, set $\hat{A}_{m+1} = \hat{x}(S_i)$, increment m to $m+1$, and go to step 2.

One important feature of the above codebook generation algorithm is that the codebook can be designed without requiring explicit knowledge of the source statistics. Furthermore, the algorithm has been shown to converge to a local optimum [94].

However, the solution is not unique [76], but depends upon the initial codebook. There are two basic approaches to obtain the initial codebook:

1. The first approach [94] uses a splitting technique where the centroid for the training sequence is calculated and split into two close vectors. The centroids or the reproduction vectors for the two partitions are then calculated. Each resulting vector is then split into two vectors and the above procedure is repeated until an N_C entry initial codebook is created. Splitting is performed by adding a fix perturbation vector ϵ to each vector v_i producing two vectors $v_i + \epsilon$, $v_i - \epsilon$.
2. The second approach [95] is to employ a product code technique, where a scalar quantizer is used K times successively to yield a K -dimensional vector quantizer.

There are a number of other techniques for codebook design. Kohonen [96, 97] introduces the self-organization feature maps method where a neural network clustering technique is used; the weights (synaptic strength) between neurons are adaptively sensitized from the prevailing input patterns and the asymptotic values of the weights define the codebook entries. Simulated annealing (Boltzmann Machine) [98, 99] is a stochastic optimization technique that can be used to find the optimal codebook patterns with respect to the training sequence; it will avoid local minima by introducing noise into the system.

Quantization

In vector quantization, the input vector $V = \{V_i\}$ drawn from an M -dimensional Euclidean space R^k is mapped into a finite subset $\hat{V}$ of R^k . Thus,

$$V : R^k \rightarrow \hat{V}, \quad (2.38)$$

where $\hat{V} = (\hat{x}_i; i = 1, 2, \dots, N_c)$ is the set of reproduction vectors and N_c the number of vectors in $\hat{V}$.

The mapping must use a minimum distortion or nearest neighbour selection rule,

$$q(V) = \hat{V}_i \quad \text{iff } d(V, \hat{V}_i) < d(V, \hat{V}_j), \quad \text{for all } j \quad (2.39)$$

where $q(\cdot)$ is the quantization operator. In other words, the quantizer chooses the code vector that results in the minimum distortion with respect to V .

Each reproduction vector is represented by a *label*. After quantization, a label is assigned to each input vector and is transmitted over the channel.

Decoding

The decoding process in VQ is simply reading a label and replacing it with the corresponding reproduction vector, or codeword. The whole procedure can be realized by a simple table lookup using the label as an entry to a table containing the reproduction vectors of the codebook.

It can be seen that the key issues in VQ are selecting the training sequence and codebook generation. An optimal codebook should completely reflect the statistics of the input vectors. A codebook generated by the LBG algorithm is optimized for the particular training sequence, chosen as the representative of the input image source.

VQ is a relatively new coding technique which achieves significant performance advantage over scalar quantization and decoding procedure can be realized by a simple LUT operation. However, the codebook design and quantization procedures involve immense computation.

2.3 Summary

In this chapter, different coding algorithms were reviewed, with emphasis on *block truncation coding*, *adaptive discrete cosine transform coding* and *vector quantization*.

Block Truncation Coding has very few computations, edge preserving ability and single pixel resolution; but only a medium compression ratio. *Vector Quantization* provides high compression ratio; fast, LUT-based, decoding and competitive subjective performance; but the encoding procedure is quite time consuming. *Adaptive Discrete Cosine Transform Coding* has high compression ratio, especially at low bit rates, and good mean square error performance, but both encoding and decoding processes are computationally complex. All three of the coding algorithms are robust in the presence of transmission errors, which implies high reliability on real channels.

Chapter 3

IMAGE FIDELITY MEASUREMENTS

3.1 Different Measurement Methods

Most data compression techniques will inevitably introduce some fidelity degradation. How to measure or scale these distortions and evaluate the quality of the processed image is a problem which has never been very well solved [102,103].

There are two classes of image fidelity measurements: *Subjective Assessment* and *Objective Measurement*.

3.2 Subjective Assessment

Subjective assessments of image quality or intelligibility are made by a panel of viewers, either untrained or experienced, depending on whether users of the coding system in question are expected to be untrained or experienced. In either case, it is important that the number of subjects be large enough so that the variability in the results are adequately small [13].

The subjective testing procedure differs depending on whether the issue is the quality or the intelligibility of the image. A high quality output implies very good

Number Scores	Quality Scale	Impairment Scale
5	Excellent	Imperceptible
4	Good	Perceptible but not Annoying
3	Fair	Slightly Annoying
2	Poor	Annoying
1	Unsatisfactory(Bad)	Very Annoying(Objectionable)

Table 3.1: Five-point adjectival scales for quality and impairment, and associated number scores [104].

intelligibility, but the converse is not true.

The subjects' basic task is easiest when the decision to be made is binary: as a two-point scale (e.g., acceptable, not acceptable). This is called the *binary decision tasks* method.

Another quality test, the *Mean Opinion Score (MOS)* method, involves the recruitment of an ensemble of subjects, each of whom classifies a stimulus (coder output) on an N-point quality scale; for example, on a 5-point adjectival scale for image quality or impairment. These scales are shown in the second and third columns of Table 3.1 [104]. An alternative, or often a supplement, to adjectival description consists in associating numbers with categories, as in the first column of Table 3.1. Judgements can thus be made on a scale of 1 to 5. The final result from these tests is a pooled average judgement called the *Mean Opinion Score* for the ensemble of viewers.

When the coding bit rate becomes very low, the intelligibility of the reconstructed image can be a serious issue. In these situations, the intent will not be to measure quality which will be quite low anyway, but rather to measure features that preserve

Figure 3.1: Image compression system.

information-bearing contrast, perhaps by the recognition of particular features as either present or absent.

3.3 Objective Measurement

Figure 3.1 shows a data compression, or coding system, where $f(i, j)$ is a $N \times N$ source image and $\hat{f}(i, j)$ is the reconstructed image. The reconstruction error $e(i, j)$ of this system is defined as,

$$e(i, j) = f(i, j) - \hat{f}(i, j),$$

$$i, j = 0, 1, \dots, N - 1. \quad (3.1)$$

For a well designed system, $e(i, j)$ is a zero mean random process. The variance of $e(i, j)$ is simply equal to its mean square value and is called the *mean square error (MSE)*, as

$$MSE = \frac{1}{N^2} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} [f(i, j) - \hat{f}(i, j)]^2. \quad (3.2)$$

MSE is an objective performance measure. However, it can only be used for comparing the results from those systems which have the same dynamic range and when processing the same image, since MSE is an absolute quantity rather than a relative quantity. To avoid this problem, *normalized mean square error (NMSE)* is introduced.

Unfortunately, there are two commonly used normalization schemes. One has to be careful when reading papers and text books in the image processing area to make sure which normalization scheme is being used by the authors while comparing their results.

The first normalization scheme is the *normalized mean square error (NMSE)*, defined as [13]

$$NMSE = \frac{\sum_{i=0}^{N-1} \sum_{j=0}^{N-1} [f(i,j) - \hat{f}(i,j)]^2}{\sum_{i=0}^{N-1} \sum_{j=0}^{N-1} [f(i,j)]^2} \times 100 \% \quad (3.3)$$

which is the total error energy divided by total signal energy.

Another commonly used normalization scheme is the *peak mean square error (PMSE)* [104], defined as

$$PMSE = \frac{\sum_{i=0}^{N-1} \sum_{j=0}^{N-1} [f(i,j) - \hat{f}(i,j)]^2}{N^2 \times [2^n - 1]^2} \times 100 \% \quad (3.4)$$

where the n is the number of bits per pixel and 2^n is the system dynamic range, which in our case is 256.

Mean square error is sometimes expressed in decibels (dB) formed as an equivalent *signal-to-noise ratio (SNR)*

$$NSNR = -10 \log_{10}(NMSE) \quad \text{dB} \quad (3.5)$$

$$PSNR = -10 \log_{10}(PMSE) \quad \text{dB} \quad (3.6)$$

The disadvantage of the *MSE* is that it simply is a measurement of error power which matches neither the *Human Visual System (HVS)* nor the properties of the image display facility, most commonly a CRT.

3.4 Human Visual System and Its Application in Image Data Compression

Presently, the most common image/graphic display device is the *Cathode-Ray Tube (CRT)*. The intensity dynamic range of a CRT is quite limited, about 40 to 60 levels, thus, a digitized image with 256 gray levels simply can not be displayed on a CRT without grey level distortion or dynamic range compaction.

The response of the human eye to changes in the intensity of illumination is known to be nonlinear [33]. Consider a patch of light of intensity $I + \Delta I$ surrounded by a background of intensity I , as shown in Figure 3.2(a), where the "just noticeable difference" ΔI is determined as a function of I . Over a wide range of intensity, it is found that the ratio $\Delta I/I$, call the *Weber fraction*, is nearly constant at a value of about 0.02 [105]; This relation is known as *Weber's law*. However, this result does not hold at very low or very high intensities, as illustrated by Figure 3.2(b) [106]. It can be understood, see Figure 3.2, that 0.02 is the smallest threshold of just noticeable difference. For an 8-bit digital image, there are 256 different intensity levels. Assuming $I = 100$, the just noticeable difference is 2 levels. If $I = 200$, it is 4 levels! This property, somehow, matches the NMSE measurement that high intensity images have lower NMSE values, Eq-4.3. In this thesis NMSE is used as the objective measurement.

From the respect of display devices and the *Human Visual System*, small values of intensity distortion are neither noticeable nor important for the implementation. One could intuitively expect that if a suitable model of the *HVS* could be successfully combined with the compression process, an improvement in compression performance would result. This expectation follows from the fact that, in most compression applications, a human is the final observer of the imagery operated upon.

Figure 3.2: Contrast sensitivity measurement [33].

Another property of the *Human Visual System* is the spatial frequency response. The HVS is a part of the nervous system [107]. The eye is the sensor of visual signals. It focuses them to form the image on the retina. Roughly speaking, the eye can be viewed as a camera. The amount of light allowed to enter is regulated by the pupil as a diaphragm. The lens focuses the light on the photoreceptive cells of the retina. The lens of the eye is, however, not perfect even for persons with no weakness of vision. This imperfection is the source of spherical aberration which appears as a blur in the focal plane. Such a blur can be modeled as a two-dimensional low-pass filter.

The retina is the neurosensorial layer of the eye. The photoreceptors in the retina transform the incoming light into electrical signals that are transmitted to the visual cortex through the optic nerve. The spatial response of the retinal cells is shown in Figure-3.3 [107]. This response corresponds to a high pass filter in the frequency domain. However, the high-pass behavior does not extend infinitely. Beyond a radial spatial frequency of about 10 cycles per degree of solid angle, an integration takes place due to the previously described low-pass filters and to the limited number of cells per unit retina length. If two excitations are closer than 1 minute of arc to each

Figure 3.3: Spatial impulse response of the retina cells.

other, their projections on the retina fall on the same cell and, therefore, are not distinguishable. As a summary, at this level of the visual system, a linear model with a band-pass frequency response can be assumed.

Simultaneous contrast effect and edge detection are mainly due to the existence of the ganglion cells in the retina. Their effect is equivalent to a two dimensional high-pass filter.

Finally, the visual cortex is a folded layer of neurons which processes the orientation of the excitation.

All the properties described in this section can be summarized in a block diagram, see Figure 3.4 [107], where different parts related to the lens, retina and cortex are indicated. The first block is a spatial, isotropic, low-pass filter. It represents the spherical aberration of the lens, the effect of the pupil, and the frequency limitation caused by the finite number of photoreceptors. It is followed by the nonlinear characteristic of the photoreceptors. At the level of the retina, this nonlinearity is followed by an isotropic high-pass filter corresponding to the effect of the ganglion cells. Finally, there is a directional filter bank that represents the processing performed by the cortex cells.

Figure 3.4: Block diagram of the human visual system [107].

Figure 3.5: Spatial frequency response of HVS [58].

An appropriate, rotationally symmetric *HVS* mathematical model was introduced recently by Nill [58], as

$$H(r) = (0.2 + 0.45r)e^{-0.18r} \quad (3.7)$$

where the radial frequency $r = \sqrt{u^2 + v^2}$ is in cycles per degree of visual angle subtended. Figure 3.5 shows the shape of this visual spatial frequency response. However, this frequency response can only be used for coding systems using the *Fourier transform*. Since the DCT is an altered DFT, as discussed in last chapter, some correction for the $H(r)$ is needed which allows use of the DCT.

Nil1 also deduced the correction factor in closed mathematical form [58], as

$$|A(v)| = \left[\frac{1}{4} + \frac{1}{\pi^2} \left[\log_e \left(\frac{2\pi}{\alpha} v + \sqrt{\frac{4\pi^2}{\alpha^2} v^2 + 1} \right) \right]^2 \right]^{1/2} \quad (3.8)$$

where α is the reciprocal of the average pulse width. From geometry

$$\alpha^{-1} = 2 \arcsin \left[\frac{w}{2\sqrt{\frac{w^2}{4} + D^2}} \right] \quad (3.9)$$

where w is the average pulse width, and D is the distance from observer to displayed-scene. For a digitized scene that has been quantized to 8 bits of grey levels and is displayed on a CRT, w is on the order of 1.5 pixels, where a typical CRT display pixel is effectively 0.5 mm on a side; then, with a typical value for D of 0.5 m, it results in $\alpha = 11.636 (\text{degrees})^{-1}$. Substituting this result into Equation-3.8, the 1-D resulting $|A(v)|$ function is plotted in Figure-3.5.

Combining $H(r)$ with $|A(v)|$, a 2-D rotationally symmetric function $|A(r)| \cdot H(r)$ is also shown in Figure 3.5. A curve fit to $|A(r)| \cdot H(r)$ is given by Nil1 [58]

$$|A(r)| \cdot H(r) = \begin{cases} 0.05e^{r^{0.554}}, & \text{for } r < 7 \\ e^{-9[|\log_{10} r - \log_{10} 9|]^2}, & \text{for } r \geq 7. \end{cases} \quad (3.10)$$

This function can now be treated in image DCT coding system applications.

As can be seen in Figure 3.5, the effect of $|A(v)|$ is to translate $H(r)$ into the more positive frequency direction, resulting in a higher frequency peak, relatively less emphasis on the lower frequencies, and relatively more emphasis on the higher frequencies. This implies that the higher spatial frequencies in the *cosine transform domain* play a more important role in the corresponding human-observed image quality than they do in the "equivalent" *Fourier transform domain* (where $A(v) = 1.0$). Thus, encoding of image *cosine transform coefficients* for bit compression requires

more attention and emphasis on maintaining the fidelity of these higher frequencies than would be needed for image *Fourier transform coefficients* [58].

The HVS spatial frequency response has been successfully implemented in image data compression [4]. However, the HVS can only be used in those image compression systems where human viewers are the final observer, such as TV, teleconferencing and HDTV systems. For scientific imagery such as satellite remote sensing imagery, synthetic aperture radar imagery and medical X-ray imagery, higher fidelity compression might be required, since some high frequency details and sudden intensity changes are crucial in their applications [2, 7].

Chapter 4

IMAGE STATISTICS

Most image coding techniques are based upon removing the redundancy between neighbouring pixels. The *autocovariance function* can be used to measure the similarity of the gray level of successive pixels. It also serves as an *image statistical model* and is used extensively in *predictive coding* to design the predictor and calculate the coding gain [13]. *Entropy* is an information measurement quantity which specifies the bit rate lower bound of lossless coding of input data although, from information theory, it can never be achieved, unless infinite delay is introduced.

In this chapter, the autocovariance function and various entropies are calculated for the test images, Picture I and Picture II. Picture I, girl with hat, is a 512 by 512 standard test image which is widely used in digital image processing. Picture II is a composite satellite image (CSI) which is a satellite remote sensing image from NOAA polar orbiter (visual band) overlain with computer generated graphic products. The image size is 512 by 512 with 1 km/pixel resolution.

The spectra of the two test images are also presented to provide a comparison in the frequency domain.

Picture I: Girl with hat.

Picture II: Composite satellite image.

4.1 Pixel Correlation

The *autocovariance function* (ACF) $\sigma^2(k)$ is a quantity which is used to measure the similarity of the gray level of successive pixels:

$$\sigma^2(k) = \frac{1}{M} \sum_{i=0}^{M-|k|-1} [f(i) - \bar{f}] [f(i + |k|) - \bar{f}] \quad (4.1)$$

where $\bar{f}$ is the mean of $f(i)$ (for notational simplicity, we use the one dimensional case). In other words, $\sigma^2(k)$ is a measure of the redundancy of an image. All data compression techniques attempt to remove this redundancy to obtain a lower bit rate.

The *autocovariance function* $\sigma^2(k)$ is often normalized as:

$$\rho(k) = \sigma^2(k)/\sigma^2(0). \quad (4.2)$$

The quantity ρ_k , which is called the *correlation coefficient*, has a value in the range $(-1, +1)$ and $\rho_0 = 1$. A value of $\rho_k = 1$ implies perfect positive correlation such as that between any two samples of a constant waveform. A value of $\rho_k = -1$ suggests perfect negative correlation such as that between any pair of adjacent samples of a sinusoid sampled only at its (positive and negative) peaks. A value of $\rho_k = 0$ means total decorrelation as between samples of a totally random white noise process. The *adjacent pixel correlation* ρ_1 is a very useful parameter. It corresponds to one pixel separation, or $k = 1$.

Figure 4.1 shows a plot of the measured correlation coefficients of Picture I, where ρ_h , ρ_v and ρ_d represent the horizontal, vertical and diagonal correlations respectively. Figure 4.2 displays the measured correlation coefficients of Picture II, with and without overlays. Their values are listed in Table 4.1. It can be seen that the girl's image has higher correlations than that of CSI, thus, a higher compression ratio can be achieved for Picture I. From Figure 4.2, it can be seen that the overlay reduces the pixel correlation substantially.

Figure 4.1: Correlation coefficients of "girl's" image.

(The straight lines represent first-order Markov sources.)

Figure 4.2: Correlation coefficients of CSI (with/without overlays).

(The straight lines represent first-order Markov sources.)

Girl's Image															
k	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$\rho_v(k)$	.97	.93	.89	.86	.83	.80	.78	.75	.73	.71	.69	.67	.65	.63	.62
$\rho_h(k)$	.96	.93	.89	.85	.81	.77	.73	.69	.66	.63	.60	.57	.54	.51	.49
$\rho_d(k)$	.95	.91	.87	.83	.80	.76	.72	.68	.65	.62	.59	.56	.53	.50	.47
Composite Satellite Image (with overlays)															
k	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$\rho_v(k)$	.80	.74	.70	.67	.64	.61	.59	.57	.55	.54	.52	.51	.50	.48	.47
$\rho_h(k)$	.84	.71	.65	.61	.57	.54	.52	.49	.47	.46	.44	.42	.41	.39	.38
$\rho_d(k)$	.79	.68	.62	.58	.56	.52	.48	.45	.43	.41	.40	.39	.38	.36	.34
Composite Satellite Image (without overlays)															
k	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$\rho_v(k)$	.90	.84	.79	.76	.72	.70	.67	.65	.63	.61	.59	.58	.56	.54	.53
$\rho_h(k)$	.88	.78	.72	.67	.64	.60	.58	.55	.53	.51	.49	.47	.45	.43	.43
$\rho_d(k)$	.86	.78	.72	.68	.63	.60	.57	.54	.52	.50	.48	.46	.44	.42	.41

Table 4.1: Values of correlation coefficients of "girl's" image and CSI (with/without overlays).

For a *first order Markov process*, the covariance function $\rho(k) = \rho_1^k$ [13,33]. As seen in Figure 4.1, for the "girl's" image, the data points fit well with a *Markov covariance function* with $\rho_h = 0.97$ and $\rho_v = 0.96$ for the horizontal and vertical correlations respectively. For the satellite image (without overlay), Figure 4.2, the plots can be fit approximately by a *Markov covariance function*, with $\rho_h \approx 0.93$ and $\rho_v \approx 0.91$.

If the horizontal and vertical correlations are exactly separable, the diagonal correlation should be equal to the product of horizontal and vertical correlations [13],

$$\rho_d(k) = \rho_v(k) \cdot \rho_h(k). \quad (4.3)$$

From Figure 4.1 and Figure 4.2, this condition does not hold.

Alternatively, if the image correlation is isotropic, the relation will be: $\rho_d = \rho_h^{\sqrt{2}}$, Figure 4.3. The data fits this relation quite well, Figure 4.1 and Figure 4.2, which implies that their autocorrelations are almost isotropic. This distinct statistical

Figure 4.3: A 2×2 grid of pixels.

property is an important discovery, which will be used later in developing our coding algorithms.

Correlation coefficients are widely used in *differential pulse code modulation* (DPCM) to calculate the optimal linear predictor coefficients. In DPCM coding, the adjacent pixels are employed to linearly predict the next pixel. For example, Figure 4.3, pixels B , C , and D can be used to predict pixel A . It has been proven that the image statistic model is quite close to the first order Markov process, which implies that most of the pixel correlations are between adjacent pixels. Thus, in image coding, non-adjacent pixels are usually not employed in the linear predictor.

Table 4.2 lists the performances of different order predictors, where the adjacent pixel correlation ρ is used to calculate the prediction coefficients. In Table 4.2, the *prediction gain*, G_p , represents the MSE improvement in DPCM over PCM for the same bit rate [13,108]. The ACF models, separable or isotropic, also affect the prediction gain.

For Picture I and Picture II, the adjacent pixel correlations ρ are about 0.965 and 0.9, respectively. The corresponding predictor coefficients and prediction gain for separable and isotropic ACF are listed in Table 4.3.

It can be seen, Table 4.3, that the separable ACF source has much higher pre-

	ACF	Predictor Coefficients	Prediction Gain G_p
1'st order predictor $\hat{x}_a = b \cdot x_b$		$b_{opt} = \rho$	$\frac{1}{1 - \rho^2}$
2'nd order predictor	Separable	$b_{opt} = c_{opt} = \frac{\rho}{1 + \rho\sqrt{2}}$	$\frac{1 + \rho^2}{1 - \rho^2}$
$\hat{x}_a = b \cdot x_b + c \cdot x_c$	Isotropic	$b_{opt} = c_{opt} = \frac{\rho}{1 + \rho^2}$	$\frac{1 + \rho\sqrt{2}}{1 + \rho\sqrt{2} - 2\rho^2}$
3'rd order predictor	Separable	$b_{opt} = c_{opt} = \rho$ $d_{opt} = -\rho^2$	$\frac{1}{(1 - \rho^2)^2}$
$\hat{x}_a = b \cdot x_b + c \cdot x_c + d \cdot x_d$	Isotropic	$b_{opt} = c_{opt} = \frac{\rho(1 - \rho\sqrt{2})}{1 + \rho\sqrt{2} - 2\rho^2}$ $d_{opt} = \rho\sqrt{2} - \frac{2\rho^2(1 - \rho\sqrt{2})}{1 + \rho\sqrt{2} - 2\rho^2}$	$[1 + 2b_{opt}^2(1 + \rho\sqrt{2}) + d_{opt}^2 + 4b_{opt}\rho(d_{opt} - 1) - 2d_{opt}\rho\sqrt{2}]^{-1}$

Table 4.2: Predictor coefficients and prediction gain of different predictors [13].

	ACF	Girl's Image $\rho = 0.965$		Satellite Image $\rho = 0.9$	
		Opt. Coef.	G_p (dB)	Opt. Coef.	G_p (dB)
1'st order predictor $\hat{x}_a = b \cdot x_b$		$b_{opt} = 0.965$	11.6	$b_{opt} = 0.9$	7.21
2'nd order predictor $\hat{x}_a = b \cdot x_b + c \cdot x_c$	Separable	$b_{opt} = c_{opt} = 0.495$	14.5	$b_{opt} = c_{opt} = 0.483$	9.79
	Isotropic	$b_{opt} = c_{opt} = 0.500$	13.4	$b_{opt} = c_{opt} = 0.497$	8.87
3'rd order predictor $\hat{x}_a = b \cdot x_b + c \cdot x_c + d \cdot x_d$	Separable	$b_{opt} = c_{opt} = 0.965$ $d_{opt} = -0.931$	23.3	$b_{opt} = c_{opt} = 0.9$ $d_{opt} = -0.81$	14.4
	Isotropic	$b_{opt} = c_{opt} = 0.536$ $d_{opt} = -0.084$	13.5	$b_{opt} = c_{opt} = 0.516$ $d_{opt} = -0.067$	8.89

Table 4.3: Predictor coefficients and prediction gain for test images.

diction gain. For $\rho = 0.965$, the prediction gains for first, second and third order predictors are 11.6, 14.5 and 23.3 dB, respectively. For the isotropic ACF source, they are 11.6, 13.4 and 13.5 dB. There is almost no gain for a third order predictor over a second order for isotropic ACF signals. As expected, the highly correlated image, girl's image $\rho \approx 0.965$, has higher prediction gain, $G_p = 13.5$ dB. The satellite image with $\rho \approx 0.9$ has only $G_p = 8.89$ dB for a third order predictor with isotropic ACF source.

4.2 Entropy

Entropy is a useful measure for data compression which gives the minimum average bit rate for perfect reproduction, or lossless coding. Consider a digital image source

X of size $N \times N$, i.e.,

$$X = \{X_{ij}, i, j = 0, 1, \dots, N-1\}.$$

If each pixel is quantized to K gray levels, i.e., $\log_2 K$ bits/pixel, then a total of $K^{N \times N}$ possible image patterns can be generated by the image source. Let $p(x)$ represent the probability of a specific image pattern,

$$x = \{x_{ij}, i, j = 0, 1, \dots, N-1\},$$

where x_{ij} is the (i, j) th element of x . The average information of the source is specified by the entropy defined as [33],

$$H(X) = -\frac{1}{N \times N} \sum_{\text{all } x} p(x) \log_2 p(x) \quad \text{bits/pixel.} \quad (4.4)$$

It has been shown [13] that the source entropy $H(x)$ is lower bounded by 0 and upper bounded by $\log_2 K$, that is,

$$0 \leq H(X) \leq \log_2 K. \quad (4.5)$$

The left-side equality holds only if all the probabilities, $p(x)$, except one are equal to zero, in which case the source is totally predictable. The redundancy of the source can be defined as the difference between the quantity $\log_2 K$ and the source entropy [13],

$$\text{redundancy} = \log_2 K - H(X). \quad (4.6)$$

If the pixels of the image are statistically independent, then the probability of a specific image pattern x can be expressed as

$$p(x) = \prod_{i,j=0}^{N-1} p_{ij}(x_{ij}). \quad (4.7)$$

Here, $p_{ij}(x_{ij})$ represents the probability that the (i, j) th element of the image source X , X_{ij} has a value equal to x_{ij} . Substituting equation (4.7) into (4.4),

$$\begin{aligned}
 H(X) &= -\frac{1}{N \times N} \sum_{\text{all } x} \left\{ \prod_{i,j=0}^{N-1} p_{ij}(x_{ij}) \log_2 \prod_{i,j=0}^{N-1} p_{ij}(x_{ij}) \right\} \\
 &= -\frac{1}{N \times N} \sum_{\text{all } x} \left\{ \prod_{i,j=0}^{N-1} p_{ij}(x_{ij}) \sum_{i,j=0}^{N-1} \log_2 p_{ij}(x_{ij}) \right\} \\
 &= -\frac{1}{N \times N} \sum_{\text{all } x} \sum_{i,j=0}^{N-1} \prod_{u,v=0}^{N-1} p_{uv}(x_{uv}) \log_2 p_{ij}(x_{ij}) \quad (4.8) \\
 &= -\frac{1}{N \times N} \sum_{i,j=0}^{N-1} \sum_{\text{all } x} \prod_{u,v=0}^{N-1} p_{uv}(x_{uv}) \log_2 p_{ij}(x_{ij}) \\
 &= -\frac{1}{N \times N} \sum_{i,j=0}^{N-1} \sum_{\text{all } x} p_{ij}(x_{ij}) \log_2 p_{ij}(x_{ij}).
 \end{aligned}$$

If the pixels of the image are identically and independently distributed (i.i.d), i.e.,

$$p_{ij}(x_{ij}) = p_{uv}(x_{uv}), \quad \text{for all } i, j \text{ and } u, v, \quad (4.9)$$

the entropy (4.8) can then be expressed as,

$$H(X) = - \sum_{k=0}^{N-1} P_k \log_2 P_k. \quad (4.10)$$

In this case, P_k is simply the probability of occurrence of the k th gray level value.

In practice the statistical information of the image, $p(x)$, is, however, not easily measured or modeled and, therefore, the true entropy of image is, in general, very difficult to obtain. In image processing, the true entropy is often replaced by the first order entropy, $H^1(X)$, which is measured on a pixel-by pixel basis and defined as,

$$H^1(X) = - \sum_{k=0}^{N-1} P_k \log_2 P_k. \quad (4.11)$$

From equation (4.10) and (4.11), it can be seen that only when the image source pixels are identical and independently distributed (i.i.d), is the first order entropy $H^1(X)$ equal to the true entropy.

The calculated first order entropies for Picture 1 and Picture 2 are listed in Table-4.4.

Image data always exhibit statistical dependencies between adjacent pixels. To take these dependencies into account conditional entropy, which is based on conditional probabilities, is introduced. The first order conditional entropy

$$H(X_n|X_{n-1}, \dots, X_{n-m+1}) = - \sum_{x_n=0}^{N-1} \dots \sum_{x_{n-m+1}=1}^{N-1} P(x_n|x_{n-1}, \dots, x_{n-m+1}) \cdot \log_2 P(x_n|x_{n-1}, \dots, x_{n-m+1}) \quad (4.12)$$

measures the uncertainty about a pixel's present value under the assumption that its m most recent values have been observed. It can be shown [109] that conditional entropy is non-increasing with m ,

$$\begin{aligned} H(X_n|X_{n-1}, \dots, X_{n-m+1}) &\leq H(X_n|X_{n-1}, \dots, X_{n-m+2}) \leq \\ &\leq \dots \leq H(X_n|X_{n-1}, X_{n-2}) \leq H(X_n|X_{n-1}) \leq H(X). \end{aligned} \quad (4.13)$$

Table 4.4 lists some conditional entropies of the test images. It can be seen that the "girl's" image has higher first order entropy, but has much lower conditional entropies (about half a bit less) than that of CSI. This, again, proves that CSI has less redundancy and contains more information.

Table 4.5 shows some differential entropies, which measure the entropies of the output of linear predictors described in the last section. They are the lower bound of the lossless DPCM coding of test images. It can be seen that, for the third order predictor which uses the three adjacent pixels to linearly predict the next pixel, CSI

	Pixel Grid	Entropy for Girl's Image (Bits)	Entropy for CSI (Bits)
$H^1(x)$		7.220	6.967
1st Order Entropies	$\begin{smallmatrix} O \\ X \end{smallmatrix}$	4.31822	5.0176
$H^1(x_n x_{n-1})$	OX	4.6534	5.0176
	$\begin{smallmatrix} O \\ X \end{smallmatrix}$	4.8471	5.2322
2nd Order	$\begin{smallmatrix} O \\ OX \end{smallmatrix}$	3.3084	3.6796
Conditional	$\begin{smallmatrix} O \\ OX \end{smallmatrix}$	3.3673	3.9463
Entropies	$\begin{smallmatrix} OO \\ X \end{smallmatrix}$	3.6981	3.9832
$H^1(x_n x_{n-1}, x_{n-2})$	OOX	3.5397	4.0175
	$\begin{smallmatrix} O \\ O \\ X \end{smallmatrix}$	3.4827	3.9661
	$\begin{smallmatrix} O \\ O \\ X \end{smallmatrix}$	3.4827	3.9661
	$\begin{smallmatrix} O \\ O \\ X \end{smallmatrix}$	3.4827	3.9661

Table 4.4: Entropies and conditional entropies.

	Pixel Grid	Entropy (girl's image) (Bits)	Entropy (CSI) (Bits)
1st Order Differential Entropys	$\begin{smallmatrix} O \\ X \end{smallmatrix}$	4.51132	5.6761
$H^1(x_n - h_1 x_{n-1})$	$\begin{smallmatrix} O & X \end{smallmatrix}$	4.9247	5.7184
	$\begin{smallmatrix} O \\ X \end{smallmatrix}$	5.1311	5.9676
2nd Order Differential Entropies $H^1(x_n - h_1 x_{n-1} - h_2 x_{n-2})$	$\begin{smallmatrix} O \\ O & X \end{smallmatrix}$	4.4892	5.4408
	$\begin{smallmatrix} O \\ O & X \end{smallmatrix}$	4.9406	5.7427
	$\begin{smallmatrix} O & O \\ X \end{smallmatrix}$	4.6684	5.6982
	$\begin{smallmatrix} O & O & X \end{smallmatrix}$	4.9607	5.1583
	$\begin{smallmatrix} O \\ O \\ X \end{smallmatrix}$	4.6975	5.8585
	$\begin{smallmatrix} O \\ O \\ X \end{smallmatrix}$	5.3479	6.1124
3rd Order Differential Entropies $H(x_n - h_1 x_{n-1} - h_2 x_{n-2} - h_3 x_{n-3})$	$\begin{smallmatrix} O & O \\ O & X \end{smallmatrix}$	4.4697	5.4371

Table 4.5: Differential entropies.

has about one bit higher differential entropy than that of “girl’s” image. It may also be noticed that second order predictor, which uses the two adjacent pixels (previous pixel on the same scan line and the one above on the previous scan line) has almost the same differential entropies as the third order predictor does. This is the result of the isotropic pixel correlation described in the last section. Since the correlation is isotropic, knowing the two adjacent pixels gives almost all the information of pixelwise correlation and the third adjacent pixel contributes very little. Thus, for the isotropic correlation case, a second order predictor may be used, instead of a third order, for its computational simplicity.

4.3 Spectrum

Figures 4.4 and 4.5 display the spectra of the two test images where full image two-dimensional FFT are implemented. They provide a comparison in the frequency domain.

It can be seen that CSI spectrum has much higher non-DC components than the girl’s spectrum. The girl’s spectrum decreases dramatically with increase in spatial frequency so that most of the power is concentrated at low frequencies, while the CSI spectrum is relatively “flat” over the frequency domain. This fact, on the other hand, proves the “busyness” of the CSI in the spatial domain. The girl’s image contains more redundancy than CSI which one could expect to remove by using some coding techniques to compress the data.

Figure 4.4: Spectrum of girl's image.

Figure 4.5: Spectrum of composite satellite image.

Chapter 5

BTC-VQ-DCT HYBRID CODING OF DIGITAL IMAGES

This chapter presents a hybrid BTC-VQ-DCT image coding algorithm which combines the simple computation and edge preservation properties of BTC and the high fidelity and high compression ratio of adaptive DCT with the high compression ratio and good subjective performance of VQ, and which may be implemented with significantly lower coding delays than either VQ or DCT alone.

The input image first undergoes BTC encoding, which codes the input image into a bit-map and two reduced-sized grey-level images which are composed of the “high-mean” and “low-mean” pixels from each block. The bit-map is then decomposed into a set of vectors and VQ is implemented. Since the space of the BTC bit-map is much smaller than that of the original 8-bit image, a LUT-based VQ encoder has been designed to “fast encode” the bit-map. Block moments are preserved in the VQ coding, as they were in the BTC. Adaptive DCT coding is implemented to encode the high-mean and low-mean sub-images. A residual error feedback technique is used to increase the fidelity and reduce the bit rate of the DCT coding.

BTC-VQ-DCT coding combines the advantages of all three coding algorithms. The overall computational complexity of BTC-VQ-DCT coding is much less than

either DCT and VQ, while the fidelity performance is competitive. It has strong edge preserving ability, because of the implementation of BTC as a pre-compressor. The total compression ratio is about 10:1.

5.1 Motivation of BTC-VQ-DCT Coding

There are three criteria that may be used to evaluate the performance of an image compression algorithm: computational complexity, compression ratio, and fidelity. Low computational complexity implies shorter coding delay and low hardware expense. High compression ratio means greater channel efficiency and reduced transmission time. High fidelity indicates low distortion, or coding errors, and high reliability. Unfortunately, all three can not be achieved uniformly. Channel efficiency can be achieved by increasing the complexity and coding delay; or by the adoption of a less stringent fidelity requirement which will reduce the amount of information to be transmitted. The algorithm described in the chapter combines the attributes of several techniques to achieve reasonable compression ratios quickly while maintaining high spatial fidelity.

Block Truncation Coding (BTC) has very few computations, edge preserving ability and single pixel resolution; but only a medium compression ratio. Vector Quantization (VQ) provides high compression ratio; fast, Look-Up-Table (LUT)-based decoding and competitive subjective performance; but the encoding procedure is quite time consuming. Adaptive Discrete Cosine Transform (DCT) coding has high compression ratio, especially at low bit rates, and good Mean Square Error (MSE) performance but both encoding and decoding processes are computationally complex. All three of the coding algorithms are robust in the presence of transmission errors, which implies high reliability on real channels.

BTC as developed by Delp and Mitchell [45] is, essentially, a 1-bit adaptive moment-preserving quantization process, which preserves certain statistical moments of small blocks of the input image. Udpikar and Raina introduced BTC-VQ coding [91], which achieved an 8:1 compression ratio, but it did not remove the inter-block redundancy. BTC-VQ-DCT coding combines the advantages of all three coding algorithms, where DCT coding is implemented to remove both the inter- and intra-block redundancy. The overall computational complexity of BTC-VQ-DCT coding is much less than either DCT or VQ, while the fidelity performance is competitive. The compression ratio is about 10:1.

5.2 BTC-VQ-DCT Coding

BTC-VQ-DCT coding has three phases:

- **Phase I:** BTC coding of the input image, which compresses blocks of the original $N \times N$ image into a “bit-map” and two reduced-sized grey-level images which are composed of the “high-mean” and “low-mean” pixel values of each BTC block, Picture III(c). The primary compression ratio of Phase-I is 4:1 for 4×4 blocks.
- **Phase II:** VQ is implemented on the bit-map to further remove the intra-block redundancy. A universal 64 K codebook is built to quickly vector quantize the bit-map through the use of a LUT. The compression ratio in this phase is between 1.6:1 and 2.67:1.
- **Phase III:** Adaptive DCT is implemented to compress the $N/4$ by $N/4$ high- and low-mean sub-images, which removes both inter- and intra- block correlations among high- and low-mean pixel values. Residual error feedback is used

to reduce the MSE and bit rate. The compression ratio is about 3:1.

5.2.1 Phase-I: BTC Encoding of the Input Image

BTC as described in Chapter 2 is, essentially, a 1-bit adaptive moment-preserving quantization process [91], which preserves certain statistical moments of small blocks of the input image in the quantized output. The original algorithm preserves the mean and standard deviation in each block [45]. Other quantization schemes result in less *mean square error* (MSE) [49], *mean absolute error* (MAE) [48], and Minimax Error [2].

The Absolute Moment BTC described in [48] quantizes the pixels in each block into high-mean and low-mean, which will preserve the Mean Absolute Moment or minimize the Mean Absolute Error in each block. This algorithm has the potential of low computational complexity with no multiplications involved.

The input image is first partitioned into non-overlapping $m \times m$ pixel blocks, in which there will likely be fairly high pixel-to-pixel correlation [45-49]. In each pixel block, the block mean

$$\bar{X} = \frac{1}{l} \sum_{i=1}^l x(i), \quad (5.1)$$

is used as the threshold for the 1-bit quantizer, where $x(i)$ is the value of i th pixel in the block and l is the number of pixels in a block, i.e., $m \times m$. If

$$x(i) \leq \bar{X} \text{ or } x(i) > \bar{X}, \quad (5.2)$$

$\bar{X}_L$ or $\bar{X}_H$, respectively, is assigned to the quantizer output levels, where

$$\bar{X}_L = \frac{1}{l-q} \sum_{x(i) \leq \bar{X}} x(i), \quad (5.3)$$

$$\bar{X}_H = \frac{1}{q} \sum_{x(i) > \bar{X}} x(i), \quad (5.4)$$

11	12	13	14	0	0	0	0	13	13	13	13
13	14	15	16	0	0	0	1	13	13	13	17
15	16	17	18	0	1	1	1	13	17	17	17
17	18	19	20	1	1	1	1	17	17	17	17

(a) Original data (b) Bit map (c) Decoded data
Threshold: $\bar{X} = 15$, High mean: $\bar{X}_H = 17$, Low mean: $\bar{X}_L = 13$.

Figure 5.1: A pixel block data before and after BTC.

and q is the number of pixels having value higher than $\bar{X}$. Hence, the outputs of the BTC for each block include two numbers which specify the high-mean, $\bar{X}_H$, and low-mean, $\bar{X}_L$; and a bit-map which specifies the state, $\bar{X}_H$ or $\bar{X}_L$, of each output pixel. Figure 5.1 shows a pixel block data before and after BTC.

In this Chapter, 4×4 block size is used and each pixel is represented by an 8-bit number. The resulting bit rate is

$$\frac{2 \times 8 + 4 \times 4}{4 \times 4} = 2 \text{ bits/pel;}$$

Thus, the compression ratio of Phase-I is 4:1.

5.2.2 Phase-II: VQ Encoding of the Bit-Map

Generally, as described in Chapter 2, VQ requires the following major steps [77]:

1. *Vector Formation*: The input image is decomposed into a set of vectors corresponding to spatially contiguous, non-overlapping, blocks of $m \times m$ pixels.
2. *Training sequence*: Either the entire set of vectors formed from the input image is taken as the training sequence for an *image adaptive VQ* codebook [90]; or a set of vectors that are statistically typical of those that will arise in the images

to be encoded are extracted from a number of images to create an *universal VQ* codebook [76].

3. *Codebook Generation*: The codebook is generated by applying the LBG clustering algorithm [94] on the training sequence where, usually, MSE or Euclidean distance is used as the distortion measure. The typical codebook sizes are 64, 128, or 256 codewords.
4. *Quantization*: For each input vector, the closest codeword in the codebook is selected and the corresponding label, which identifies the address in the codebook, is transmitted.
5. *Decoding*: At the receiving end, the coded image is simply reconstructed by using the labels as the entries to select the codeword and construct the corresponding block in the reconstructed image.

The typical VQ block size is 4×4 . For 8 bits/pel digital imagery, there are 2^{128} representation vectors in the image sample space. It is impossible to build an encoding LUT to map each of these input vectors into a corresponding codeword. This implies that the encoder must perform an exhaustive search through the entire codebook to find the minimum distortion codeword for each input vector. The computation involved in this vector matching procedure is enormous.

In our special case, VQ is implemented on the bit-map, thereby drastically reducing the VQ coding complexity in three ways: (1) VQ can be realized by a simple LUT operation; (2) *universal codebook VQ*, which does not require codebook generation or an update process for each input image, can be used instead of *image adaptive VQ*; (3) Hamming distance¹ may be used to measure the distortion rather than Euclidean

¹Suppose that vector $\mathbf{x} = (x_1, \dots, x_N)$, where $x_n = \{0, 1\}$, differs from vector $\mathbf{y} = (y_1, \dots, y_N)$, where $y_n = \{0, 1\}$, in exactly d positions. The number d is then said to be the *Hamming distance*

distance,² which can speed up the process of codebook and encoding-LUT generation.

The bit-map, from Phase-I, is decomposed into 4×4 blocks. Each block can be packed into a 16-bit word forming a 16-dimensional vector. Actually, this vector formation procedure can be embedded in the Phase-I coding to save memory space. The total number of representation vectors is only $2^{16} = 64K$. It is reasonable to design a 64-Kword encoding LUT to fast-encode the input vector. Once the LUT is built, it takes almost no time to quantize a word-packed vector.

Since we are encoding a bit-map, simple Hamming distance may be used as the distortion measurement instead of Euclidean distance. Hence, only *exclusive OR* and *addition* are needed to calculate the distortion, instead of *multiplication* and *addition* in MSE measurement which is more time consuming. Moreover, the *bit-wise exclusive OR* operation can be performed on the word-packed vector, which further reduces memory access time.

Because we have a relatively small vector sample space, 64K, the controversy of whether the universal codebook VQ (VQU) or the adaptive codebook VQ (VQA) approach is superior is actually phased out. For conventional VQA coding, the overhead, which is the codebook, may require a large number of bits. In our case, the codebook needs at most one-eighth as many bits, since our input is a bit-map instead of an 8 bits/pel image. On the other hand, the small sample space of the input vectors results in VQA performance only slightly better than VQU, even though they

¹between vector $\mathbf{x}$ and $\mathbf{y}$. *Hamming distance* can be calculated by a modulo-two term-by-term sum, $\mathbf{b} = \mathbf{x} \oplus \mathbf{y}$, followed by a table look up to find out how many binary 1's in $\mathbf{b}$.

²The *Euclidean distance* between vector $\mathbf{x} = (x_1, \dots, x_N)$, where $x_n = \{0, \dots, 255\}$, and vector $\mathbf{y} = (y_1, \dots, y_N)$, where $y_n = \{0, \dots, 255\}$, is

$$\|\mathbf{x} - \mathbf{y}\| = \sqrt{\sum_{n=1}^N (x_n - y_n)^2}.$$

use the same size of codebooks. If VQ is implemented directly on a gray-level image instead of a bit-map, VQU usually needs a larger codebook than VQA for the same MSE performance [101].

The MSE performances are shown in Table 5.1, where a codebook with $N_c = 256$ codewords is implemented. It can be seen that the *normalized mean square error* (NMSE) of BTC-VQU and BTC-VQA differ by only 0.5 dB, when applied to Picture-III (a). Comparing the computational complexity of VQA, where the codebook must be generated for each image by implementing the time consuming LBG algorithm [94], over VQU, where there is no codebook generation during the coding procedure, VQU seems a better choice.

In our codebook generation and encoding-LUT generation procedures, there is the possibility that there could be more than one codeword with the same Hamming distance from the input vector. Among them, we choose the one with the same number of “0’s” (or “1’s”) in the bit-map block. This way, the moments are still preserved in each BTC block.

The codebook used in VQU was generated from a group of training images that includes standard images, satellite images, and radar images. A 64-Kword encoding-LUT was built from the codebook through an exhaustive-search, based on minimum Hamming distance and the moments preserving scheme.

5.2.3 Phase-III: Adaptive DCT Encoding of the High- and Low-Mean Sub-Images

If we group the high-means, and the low-means, from each BTC block together as raster files, we can form two sub-sampled images — high-mean and low-mean sub-images. For a 512×512 input image with 4×4 blocks, the subsampled images are 128×128 in size. The subsampled-images have details and features which must

be preserved, since any distortion involved here will be distributed over all of the pixels in each reconstructed BTC block. The coding algorithm used to compress these subsampled-images must have high fidelity, even at the cost of a “high” bit rate; however, moderate computational complexity is acceptable, because the sub-image size is relatively small, and more bits, e.g., 2–3 bits/subsampled-pixel, can be allocated to code them since the real bit rate is the subsampled-image bit rate divided by the BTC block size, i.e., 16.

VQ and adaptive DCT may be considered as techniques to compress the sub-images. VQ has very good performance, among other coding algorithms, at low bit rates. At high bit rates, however, its performance degrades and codebook design becomes very difficult and time consuming. Adaptive DCT coding [4,70,71], see Chapter 2, which adaptively allocates bits in the transform domain based on fidelity requirements, may have very high fidelity, and seems a good candidate for the compression of the subsampled-images. Although DCT coding tends to smear edges because some high frequency components are discarded as part of the encoding procedure, the edges in the original image have already been preserved by the BTC pre-processing.

There are three ways to DCT encode the high- and low-mean sub-images from Phase-I:

1. direct coding;
2. direct-differential coding;
3. direct-differential coding with residual error feedback.

These are diagrammed in Figure 5.2.

Figure 5.2: Adaptive DCT coding of high- and low-mean sub-images.

Error Analysis for Three DCT Coding Algorithms

Method-I: Adaptive DCT [70] is directly implemented on both the high-mean sub-image, S_H , and the low-mean sub-image, S_L , Figure 5.2(a). Let $\hat{S}_H$ and $\hat{S}_L$ be reconstructed mean-subsampled-images, then

$$\hat{S}_H = Q(S_H) = S_H + \mathbf{e}_H, \quad (5.5)$$

and

$$\hat{S}_L = Q(S_L) = S_L + \mathbf{e}_L, \quad (5.6)$$

where $Q(\cdot)$ represents the coding/quantization process which can be thought as a re-quantization of the input sub-images, $\mathbf{e}_H$ and $\mathbf{e}_L$ are quantization errors. Assuming $\mathbf{e}_H$ and $\mathbf{e}_L$ are *independent identically distributed (i.i.d.) random variables (RVs)*, the total average coding error energy is

$$\overline{e^2}(L, H) = \overline{e_L^2 + e_H^2} = 2\overline{e^2}. \quad (5.7)$$

Method-II: Adaptive DCT is implemented on S_L or S_H , and the difference image,

$$S_{H-L} = S_H - S_L, \quad (5.8)$$

in Figure 5.2(b). Since a difference image usually has less entropy, i.e.,

$$H(S_{H-L}) \leq H(S_H) \text{ and } H(S_{H-L}) \leq H(S_L),$$

less bits can be used to code S_{H-L} than S_H or S_L for the same fidelity. Thus,

$$\hat{S}_{H-L} = Q(S_{H-L}) = S_{H-L} + \mathbf{e}_{H-L}. \quad (5.9)$$

From (5.6) and (5.9),

$$\begin{aligned} \hat{S}'_H &= Q(S_L) + Q(S_{H-L}) \\ &= S_{H-L} + S_L + \mathbf{e}_{H-L} + \mathbf{e}_L. \end{aligned} \quad (5.10)$$

From (5.6) and (5.10), the total error energy is

$$\overline{e^2}(L, H - L) = \overline{e_{H-L}^2 + e_L^2 + e_L^2} = 3\overline{e^2}. \quad (5.11)$$

Method-III: Adaptive DCT is implemented on S_L and the difference image $S_H - S_L$, using residual error feedback [110,111,112] as shown in Figure 5.2(c), where S_L is the reconstructed S_L . Hence,

$$\hat{S}_{H-L} = Q(S_H - S_L), \quad (5.12)$$

Inserting (5.6),

$$\hat{S}_{H-L} = Q(S_H - S_L + e_L) = Q(S_{H-L} + e_L). \quad (5.13)$$

For simplicity, assuming the quantization process $Q(\cdot)$ is linear,

$$\hat{S}_{H-L} \approx \hat{S}_{H-L} + Q(e_L) = S_{H-L} + e_{H-L} + Q(e_L). \quad (5.14)$$

From (5.6) and (5.14),

$$\overline{e^2}(L, H - \hat{L}) = \overline{e_{H-L}^2 + e_L^2 + Q^2(e_L)} = 2\overline{e^2} + \overline{Q^2(e)}. \quad (5.15)$$

$Q(e)$ is the quantizer output due to the quantization error e . Usually, e can be modelled as *white Gaussian noise* [13], which has a “flat” spectrum, or relatively “wide” bandwidth. Since most of quantizers have a low-pass type frequency response, $Q(e)$ must have less energy than e , i.e.,

$$\overline{Q^2(e)} < \overline{e^2}. \quad (5.16)$$

On the other hand, e has a *Gaussian distribution* [113], in the time domain, with zero mean. Most of the time, its amplitude is relatively small. When e is less than the quantizer step size, or threshold, there will be no output from the coding/decoding process $Q(\cdot)$,

$$\overline{Q^2(e)} \approx 0, \quad \text{when } e \approx 0. \quad (5.17)$$

From (5.16) and (5.17),

$$0 < \overline{Q^2(e)} < \overline{e^2}. \quad (5.18)$$

Thus,

$$2\overline{e^2} \leq \overline{e^2}(L, H - \hat{L}) \leq 3\overline{e^2}. \quad (5.19)$$

Comparing (5.7), (5.11) and (5.19), the error performance of the three methods can be related as

$$\overline{e^2}(L, H) \leq \overline{e^2}(L, H - \hat{L}) \leq \overline{e^2}(L, H - L). \quad (5.20)$$

Method-I has the best MSE performance and least computational complexity, but the highest bit rate since the correlations between the two sub-images are not removed. Method-II has the worst MSE, moderate computational complexity and low bit rate. Method-III has moderate MSE, high computational complexity (about 25 % more computation than the other methods) and low bit rate. However, since the size of the sub-images is relatively small (16 times less than the original image) the computational complexity is trivial. Considering low bit rate and MSE performance method-III is the best candidate.

Simulation Results

The MSE simulation results, displayed in Table 5.1, satisfy the above analysis. From (5.7), row 1 and 4 of Table 5.1, $\overline{e^2} = (31.9 - 25.1)/2 = 3.4$. From the previous mathematical analysis, the MSE of BTC-DCT_{L,H-L} is expected to be higher than that of BTC-LCT_{L,H} by about 3.4, which is almost exactly the case in Table 5.1. Meanwhile, the MSE of BTC-DCT_{L,H-L} is much closer to that of BTC-DCT_{L,H}, the lower bound of (5.20) than that of BTC-DCT_{L,H-L}, thus method-III is selected for use in the hybrid algorithm.

2

MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS
STANDARD REFERENCE MATERIAL 1010a
(ANSI and ISO TEST CHART No. 2)

As mentioned before, the sub-sampled images must be coded with high fidelity, since any distortion involved here will be distributed over all of the pixels in each BTC block. Simulation results show that the DCT coding bit rate should be around 2–3 bits/subsampled-pixel, or higher, to achieve satisfactory subjective results for the reconstructed image. If the DCT coding bit rate is lower than that, fault contours will appear in the reconstructed image, which then looks like a coarsely quantized image.

The entire system diagram of BTC-VQ-DCT coding is shown in Figure 5.3.

5.3 System Performances

5.3.1 Computational Complexity of BTC-VQ-DCT vs. Direct VQ and DCT

In direct VQ, the *computational complexity*, multiplication/addition per pixel, is proportional to N_c , which is the number of codewords in the codebook. For direct DCT with an $N \times N$ block size, the *computation complexity* is about $4 \cdot \log_2 N$ multiplications/additions per pixel. For BTC-VQ-DCT, the BTC part requires only two additions per pixel; the LUT based VQ needs no computation but only vector formation and addressing; the DCT part, with residual error feedback, requires

$$\frac{10 \cdot \log_2 N}{N_b \times N_b} \text{ multiplications/additions per pixel,}$$

where N_b is the BTC block size. Table 5.2 compares the computational requirements of the three coding algorithms, for $N_b = 4$, $N = 16$, and $N_c = 256$; where k_n , $n = 1, 2, 3$ is a scale constant, which represents other computer operations, e.g., memory access, data type conversions, etc. Usually, $1 < k_n < 2$. From Table 5.2, the computation associated with BTC-VQ-DCT with residual error feedback is 6.4

	MSE	NMSE		bits/pel
BTC	25.1	0.0700 %	31.55 dB	2.0
BTC-VQU	37.9	0.1058 %	29.76 dB	1.5
BTC-VQA	33.2	0.0928 %	30.32 dB	1.5156
BTC-DCT _{L,H}	31.9	0.0891 %	30.50 dB	1.375
BTC-DCT _{L,H-L}	35.3	0.0985 %	30.07 dB	1.3125
BTC-DCT _{L,H-L̄}	32.1	0.0896 %	30.48 dB	1.3125
BTC-VQU-DCT _{L,H}	45.8	0.1278 %	28.93 dB	0.875
BTC-VQU-DCT _{L,H-L}	48.8	0.1362 %	28.66 dB	0.8125
BTC-VQU-DCT _{L,H-L̄}	46.2	0.1289 %	28.90 dB	0.8125
BTC-VQA-DCT _{L,H}	40.8	0.1139 %	29.43 dB	0.875
BTC-VQA-DCT _{L,H-L}	43.9	0.1225 %	29.12 dB	0.8125
BTC-VQA-DCT _{L,H-L̄}	41.2	0.1151 %	29.39 dB	0.8125

Table 5.1: MSE performance of computer simulation results for the Girl image, codebook size = 256.

	Computational Complexity	Multi/Add per Pixel	$k_n = 1$
VQ	$k_1 \cdot N_c$	256	$N_c = 256$
DCT	$k_2 \cdot 4 \cdot \log_2 N$	16	$N_b = 4$
BTC-VQ-DCT	$k_3 \cdot (10 \cdot \log_2 N) / N_b^2$	2.5	$N = 16$

Table 5.2: Computation complexities of VQ, DCT and BTC-VQ-DCT.

times less than that of DCT and 102.4 times less than that of VQ. The advantage of BTC-VQ-DCT is quite clear.

5.3.2 Encoding and Decoding Delay of VQ, DCT and BTC-VQ-DCT

Coding delay, usually, is directly related to the computational complexity. Among the three coding algorithms, VQ has the most computational complexity. However, most of its computations are at the encoding end, the decoding process is simply a LUT. For DCT, the computations are almost equally divided between encoder and decoder. For BTC-VQ-DCT with residual error feedback, encoding costs about 2/3 of the computation and decoding 1/3. Hence, from Table 5.2, considering encoding delay, the favorable order is BTC-VQ-DCT, DCT and VQ; for decoding delay, it is VQ, BTC-VQ-DCT and DCT; while for the overall coding delay, it is BTC-VQ-DCT, DCT and VQ.

R_b bpp	R_m bpp	R_{total} bpp	NMSE (%) Picture III (a)	NMSE (%) Picture V (a)
1.0	0.3125	1.3125	0.0896	1.397
0.625	0.3125	0.9375	0.1100	1.651
0.5625	0.3125	0.875	0.1197	1.882
0.5	0.3125	0.8125	0.1289	2.325
0.4375	0.3125	0.75	0.1373	2.836
0.375	0.3125	0.6875	0.1596	3.500

Table 5.3: Simulation results of BTC-VQ-DCT.

5.3.3 Error Performance of VQ, DCT and BTC-VQ-DCT

Table 5.3 lists the test results of BTC-VQ-DCT coding for different VQ block sizes, where R_b , R_m , and R_{total} are bit rates of coding the bit-map, mean sub-images, and total bit rate respectively. Two test images are used, Picture-III (a), a standard test image, and Picture-V (a), a satellite image overlain with computer graphics which contains sharp edges and even single-pixel-width gray level sudden changes.

Figure 5.3 displays the rate distortion curves of VQ [13,14], DCT [10] and BTC-VQ-DCT. It can be seen that, for the simple image shown in Picture-III (a), VQ and DCT have better MSE performance than BTC-VQ-DCT. However, for the complex image shown in Picture-V (a), BTC-VQ-DCT has competitive NMSE at a bit rate higher than 0.8 bits/pixel, because its BTC pre-coding has high edge-preserving ability. For bit rates lower than 0.8 bits/pixel, its VQ of bit-map simply cannot provide satisfactory fidelity; however, because BTC-VQ-DCT has much less computational complexity than either VQ or DCT, it may be used satisfactorily at bit rates greater than 0.8 bits/pixel.

The MSE performances are listed in Table 5.1. Picture IV (a) and Picture V (b)

Figure 5.4: Rate distortion curves of VQ, DCT and BTC-VQ-DCT.

Picture III (a)	Bit-Rate	NMSE	Computation-Complexity
VQ	0.406	0.134	64
DCT	0.400	0.094	16
BTC-VQ-DCT	0.813	0.135	2.5
Picture V (a)	Bit-Rate	NMSE	Computation-Complexity
VQ	0.625	2.201	256
DCT	0.651	2.243	16
BTC-VQ-DCT	0.813	2.325	2.5

Table 5.4: Bit rate, NMSE and computational complexity.

show the reconstructed images at a bit rate of 0.8125 bits/pixel and 0.875 bits/pixel.

5.4 Bit Rate, NMSE and Computational Complexity

An ideal coding algorithm should have a low bit rate that saves transmission time, while producing reconstructed images are still subjectively acceptable; a low NMSE value which indicates high fidelity; and a low computational complexity which means short coding delay and low hardware expense.

It is common to use the rate-distortion curve, $R(D)$, Figure 5.3, to display the performance of an image coder. $R(D)$ gives the relationship between the bit rate and NMSE, but does not indicate the computational complexity.

The rate-distortion curves are source dependent, since different sources carry dif-

ferent amounts of information. A pre-fixed distortion value (NMSE) will result in different bit rates for different sources.

For computational complexity, multiplication/addition per pixel probably is the most suitable measure, since it is the most basic operation in digital signal processing and most state-of-the-art digital signal processors can finish it in one instruction cycle.

The bit rate, NMSE value and computational complexity (multiplication/addition per pixel) for VQ, DCT and BTC-VQ-DCT coding are listed in Table 5.4, where test images, Picture-III(a) and Picture-V(a), are used. It can be seen that, for the same NMSE and compatible bit rate, BTC-VQ-DCT has the lowest computational complexity.

As for hardware realization of BTC-VQ-DCT coding, BTC and VQ can be implemented by general purpose DSP chips, e.g., Intel's TMS320CXX. Inmos has produced an 8 by 8 DCT chip [128] that could be used to compress the sub-sampled images. However, since the DCT is implemented on sub-sampled images, it can also be calculated by a DSP chip, which is one of the advantages of BTC-VQ-DCT coding.

5.5 Summary

This chapter presented a multi-stage, hybrid image coding algorithm which combines the simple computation and edge preservation advantages of BTC and the high fidelity and compression ratio of adaptive DCT and the high compression ratio and good subjective performance of VQ, with an overall computational complexity that is much less than either DCT and VQ, while the fidelity performance is competitive. Good performances at a compression ratio of about 10:1 are demonstrated.

Picture III: (a) Original image; (b) Bit-map; (c) High- and low-mean images, and their difference image.

Picture IV: (a) Reconstructed image; (b) Coded bit-map; (c) Coded high- and low-mean images, and their difference image.

Picture V: (a) Composite satellite image; (b) Reconstructed image.

Chapter 6

MULTI-LEVEL BTC USING A MINIMAX ERROR CRITERION FOR HIGH FIDELITY COMPRESSION OF DIGITAL IMAGES

This Chapter describes an encoding technique, Multi-Level Block Truncation Coding, that preserves the spatial details in digital images while achieving a reasonable compression ratio. An adaptive quantizer-level allocation scheme is introduced, which minimizes the maximum quantization error in each block and substantially reduces the computational complexity in allocation of optimal quantization levels. A 3.2:1 compression can be achieved by the multi-level block truncation coding itself.

The truncated, or re-quantized, data are further compressed in a second pass using combined techniques of predictive coding, entropy coding and vector quantization. The second pass compression can be lossless or lossy. The total compression ratios are about 4.1:1 for lossless second pass compression, and 6.2:1 for lossy second pass compression.

The subjective results of the coding algorithm are quite satisfactory, with no

perceived visual degradation. In addition, the technique is designed so that the compressed images are easily and quickly decoded on general purpose computers, as appropriate for widely disseminated products.

6.1 Introduction

Most image coding algorithms, such as transform coding, predictive coding [13,32], and vector quantization [23-27], are based on a Mean Square Error (MSE) fidelity criterion [120,121]. The major disadvantage of MSE-based coding schemes is that they act, more or less, like low-pass filters. The high frequency details are blurred or even filtered out in the reconstructed image since they normally have low probability and, thus, may be discarded without a large contribution to the MSE. This low-pass filter effect is acceptable for some images, such as television and teleconferencing pictures. On the other hand, some scientific images such as satellite remote sensing, synthetic aperture radar (SAR) and medical imagery contain very high resolution artifacts coinciding with sharp edges and sudden single-pixel-width changes in gray level which can be crucial in the image application and must be preserved. As well, these images have large data sets which require massive storage space and result in substantial transmission times when the images are communicated over data networks. Finding a way to compress such images while maintaining high fidelity is very important; and, because the images are widely distributed, the decompression algorithms are expected to run efficiently on general purpose computers, with no special purpose hardware required.

As mentioned in Chapter 5, Block Truncation Coding (BTC) [45-49] uses a two-level moment-preserving quantizer that adapts to local properties of the image. It has the advantages of preserving single pixel resolution and edges and having low

computational complexity. The original algorithm preserves the block mean and the block standard deviation [45,46]. Other choices of the moments result either in less MSE or less computational complexity [48,49]. However, a two-level quantization is simply not able to code a highly busy image with satisfactory fidelity [51]. To achieve high fidelity, we use a multi-level quantizer. In Section 6.2, an algorithm using a minimax quantization error criterion is developed to allocate the quantizer levels to obtain a close replica of the original image while being relatively insensitive to the statistical distribution within each block. A fast searching algorithm is introduced to find the optimal quantizer levels, based on minimax quantization error. Sections 6.3 – 6.6 discuss the post-quantization compression of the coded data using combined techniques of predictive coding, entropy coding and vector quantization.

The images used to test the coding algorithm are a standard test image, Picture I, and composite satellite imagery (CSI) [3], Picture II. CSI is a pictorial satellite remote sensing image overlain with computer generated graphics. The graphical portions are very distinctive in gray level relative to the surrounding pixels, and should be coded losslessly so that, in the reconstructed image, they can be identified. The pictorial portion contains high frequency details which must be well preserved.

The contents of CSI are very random with first order entropy around 7 bits per pixel and first order conditional entropies of about 5 bits per pixel. For the standard image, Picture II, the first order entropy is also around 7 bits per pixel but the conditional entropy is only about 4 bits per pixel.

The test results of our coding algorithm are quite satisfactory. No visual degradation is perceived.

6.2 Minimax Error Multi-Level BTC and A Fast Quantizer Level Allocation Scheme

6.2.1 Minimax Error Multi-Level BTC

In Multi-Level Block Truncation Coding, the image is partitioned into $M \times M$ pixel blocks. The *random variable (r.v.)* $\mathbf{X}$ is assumed to represent the pixel values in a block and, in the first case, to be continuously distributed over an interval $[x_{\min}, x_{\max}]$, Fig. 6.1. A N -level quantizer is used to quantize $\mathbf{X}$. The N output values of the quantizer are

$$\{y_n\} = y_0, y_1, \dots, y_{N-1}$$

where

$$y_0 \leq y_1 \leq \dots \leq y_{N-1}. \quad (6.1)$$

The $N - 1$ threshold values are

$$\{x_n\} = x_1, x_2, \dots, x_{N-1}$$

and

$$x_1 \leq x_2 \leq \dots \leq x_{N-1}. \quad (6.2)$$

An output value $y_r, y_r \in \{y_n\}$, is assigned to represent the interval $[x_r, x_{r+1}]$, where $x_r, x_{r+1} \in \{x_n\}$. Thus

$$x_r \leq y_r \leq x_{r+1}. \quad (6.3)$$

The maximum absolute quantization error on the interval is

$$|e_{r \max}| = \max \{x_{r+1} - y_r, y_r - x_r\}.$$

To minimize $e_{r \max}$, the y_r must be

$$y_r = \frac{x_r + x_{r+1}}{2}, \quad (6.4)$$

Figure 6.1: Probability distribution functions.

which means that y_r must be set to the mid-point of $[x_r, x_{r+1}]$. Therefore,

$$\min\{|e_{r \max}|\} = x_{r+1} - y_r = y_r - x_r = \frac{x_{r+1} - x_r}{2} \quad (6.5)$$

PROOF 1: If y_r is not set to the mid-point, then

$$y_r < \frac{x_r + x_{r+1}}{2} \text{ or } y_r > \frac{x_r + x_{r+1}}{2}. \quad (6.6)$$

From Eq-6.6

$$x_{r+1} - y_r > y_r - x_r \text{ or } y_r - x_r > x_{r+1} - y_r,$$

Hence

$$|e_{r \max}| = x_{r+1} - y_r \text{ or } |e_{r \max}| = y_r - x_r. \quad (6.7)$$

Substitute Eq-6.7 into Eq-6.6,

$$|e_{r \max}| > \frac{x_{r+1} - x_r}{2}. \quad (6.8)$$

Compare Eq-6.8 and Eq-6.5, $e_{r \max}$ is not the minimum value, for $y_r \neq \frac{x_r + x_{r+1}}{2}$. This inversely proves that, if

$$y_r = \frac{x_r + x_{r+1}}{2},$$

then

$$\min \{ |e_{r \max}| \} = \frac{x_{r+1} - x_r}{2}.$$

END OF PROOF.

If the entire sample interval $[x_{\min}, x_{\max}]$ is partitioned into N equal-length intervals by $\{x_n\}$, i.e.,

$$\begin{aligned} x_n &= x_{\min} + \frac{x_{\max} - x_{\min}}{N} \cdot n, \\ n &= 1, 2, \dots, N-1. \end{aligned} \quad (6.9)$$

then

$$\begin{aligned} x_1 - x_{\min} &= x_2 - x_1 = \dots = x_r - x_{r-1} = \\ &= \dots = x_{\max} - x_{N-1} = \frac{x_{\max} - x_{\min}}{N}. \end{aligned} \quad (6.10)$$

If the N output values $y_0, y_1, \dots, y_{N-1}$ are set to the mid-point of each interval, $(x_{\min}, x_1), (x_1, x_2), \dots, (x_{N-1}, x_{\max})$, respectively, from Eq-6.4 and Eq-6.10, on each interval (x_r, x_{r+1}) ,

$$|e_{r \max}| = \frac{x_{r+1} - x_r}{2} = \frac{x_{\max} - x_{\min}}{2N}, \quad (6.11)$$

which also means

$$|e_{0 \max}| = |e_{1 \max}| = \dots = |e_{N-1}| = \frac{x_{\max} - x_{\min}}{2N}. \quad (6.12)$$

Thus

$$\min \{ |e_{\max}| \} = \frac{x_{\max} - x_{\min}}{2N}. \quad (6.13)$$

PROOF 2: If $[x_{\min}, x_{\max}]$ is not partitioned into N equal-length intervals, there is, at least, one interval, $[x_l, x_{l+1}]$, where $x_l, x_{l+1} \in \{x_{\min}, x_{\max}\}$, on which

$$x_{l+1} - x_l > \frac{x_{\max} - x_{\min}}{N}. \quad (6.14)$$

From Eq-6.11 and Eq-6.14,

$$|e_{l \max}| = \frac{x_{l+1} - x_l}{2} > \frac{x_{\max} - x_{\min}}{2N}, \quad (6.15)$$

Figure 6.2: A block histogram with $x_{min} = 5$, $x_{max} = 8$ and $N = 3$.

Hence

$$|e_{max}| = |e_{lmax}| > \frac{x_{max} - x_{min}}{2N}. \quad (6.16)$$

To satisfy

$$|e_{max}| = \frac{x_{max} - x_{min}}{2N}, \quad (6.17)$$

$[x_{min}, x_{max}]$ must be partitioned into N equal-length intervals, which will result in

$$\min \{ |e_{max}| \} = \frac{x_{max} - x_{min}}{2N}.$$

END OF PROOF.

One important point of the above results is that only the continuity of the *pdf* of the input *r.v.* $\mathbf{X}$ over the interval $[x_{min}, x_{max}]$ is mentioned, but not the type of the *pdf*. This means that Eq-6.17 is valid as long as the *pdf* is continuous. More generally, even if there are singular points on the interval, or a finite number of “zeros” on it, see Fig. 6.1, Eq-6.17 still holds.

In digital image processing, the input data are usually discrete signals with 8 bits per pixel, i.e., 256 possible gray levels, which are equally spaced over the interval $[0, 255]$.

If the input data has a *discrete pdf*, $f(x_n)$, defined over the interval $[x_{min}, x_{max}]$,

Figure 6.3: Block histogram with "zero" intervals.

where $x_{min}, x_{max} \in [0, 255]$, and

$$f(x_n) \neq 0 \quad x_n \in [x_{min}, x_{max}], \quad (6.18)$$

Eq-6.17 still holds for the discrete case, if $N \leq x_{max} - x_{min}$ and a rounding scheme is used. For example, $x_{min} = 5$, $x_{max} = 8$ and $N = 3$, from Eq-6.17, see Fig. 6.2,

$$\min \{ |e_{max}| \} = \frac{8 - 5}{2 \times 3} \simeq 1.$$

From now on, only the discrete case is considered.

In the more general case, portions of the *pdf* which are zero within the interval $[x_{min}, x_{max}]$,¹ i.e., if $\{x_m\}$ is a subset of $\{x\}$, and $f(x_m) = 0$, might result in, see Fig. 6.3,

$$|e_{max}| < \frac{x_{max} - x_{min}}{2N} \quad (6.19)$$

In an extreme case, i.e., when there are less than N non-zero points on the interval $[x_{min}, x_{max}]$,

$$|e_{max}| = 0. \quad (6.20)$$

¹There are always zero components on the block histogram of CSI images, especially for "busy" blocks and blocks with graphic overlays on them.

Figure 6.4: In the case that non-zero components can be partitioned into N equal-length interval.

Hence Eq-6.17 is the upper bound of $\min \{ |e_{\max}| \}$, and Eq-6.20 is the lower bound. Thus

$$0 \leq \min \{ |e_{\max}| \} \leq \frac{x_{\max} - x_{\min}}{2N}. \quad (6.21)$$

For the general case, when there are K block histogram components, where $K \leq x_{\max} - x_{\min} - 1$, on the interval $[x_{\min}, x_{\max}]$ that are equal to zero ($f(x_{\max}) \neq 0$ and $f(x_{\min}) \neq 0$), then Eq-6.20 should be

$$\frac{x_{\max} - x_{\min} - K}{2N} \leq \min \{ |e_{\max}| \} \leq \frac{x_{\max} - x_{\min}}{2N}. \quad (6.22)$$

The equality on the left side of Eq-6.21 holds only if the *non-zero components* can be partitioned into N equal-length intervals, see Fig. 6.4.

PROOF 3: If the above statement is not true, there is, at least, one interval S_l , which is

$$S_l > \frac{x_{\max} - x_{\min} - K}{N}, \quad (6.23)$$

and on that interval

$$|e_{l\max}| > \frac{x_{\max} - x_{\min} - K}{2N}. \quad (6.24)$$

Thus Eq-6.24 gives the lower bound of $|e_{max}|$. Combining Eq-6.19 and Eq-6.24,

$$\frac{x_{max} - x_{min} - K}{2N} \leq \min \{ |e_{max}| \} \leq \frac{x_{max} - x_{min}}{2N}.$$

END OF PROOF.

Actually, Eq-6.22 is a more general expression of Eq-6.21. For example, if $N = 10$, $x_{max} - x_{min} - K = 7$ (less than N). Then

$$\min \{ |e_{max}| \} = \frac{x_{max} - x_{min} - K}{2N} = \frac{7}{20} \simeq 0.$$

(Note: $x_{max} - x_{min} - K$ equals the number of non-zero components over the interval $[x_{min}, x_{max}]$.)

6.2.2 Fast Quantizer Level Allocation Scheme

So far, the lower and upper bounds of $\min \{ |e_{max}| \}$ have been found, Eq-6.22. The next step is to find the best threshold allocation to obtain $\min \{ |e_{max}| \}$.

Fig. 6.5(a) is a typical input data histogram $f(x)$ over the interval $[x_{min}, x_{max}]$. Assuming a 4-level quantizer is used, $N = 4$. There will be $N - 1 = 3$ thresholds, x_1 , x_2 , and x_3 . Using Eq-6.9, the thresholds are equally placed with spacing

$$\frac{x_{max} - x_{min}}{N},$$

see Fig. 6.5(b). This partition will give the upper bound of $\min \{ |e_{max}| \}$, the right side of Eq-6.22. However, there are “zeros” on the sample interval, which means, from Eq-6.22, that $\min \{ |e_{max}| \}$ can be less.

Since threshold x_3 is located on a “zero” interval, if we “dilate” x_3 to mask out “zeros”, see Fig. 6.5(c), then y_3 will provide a better error performance, where

$$y_3 = \frac{x'_{3r} + x_{max}}{2}.$$

Figure 6.5: An example of the threshold-dilation algorithm to "fast search" output levels which minimize $|e_{\max}|$ over a pixel block.

Now, the new threshold x'_3 occupies an interval (x'_{3l}, x'_{3r}) . The next step is to re-allocate x'_1 and x'_2 to partition $[x_{min}, x'_{3l}]$ into equal-length intervals, see Fig. 6.5(c). Again, x'_1 can be dilated and mask out some "zeros". y_0 can be set as

$$y_0 = \frac{x_{min} + x'_{1l}}{2}.$$

The new x''_2 is allocated using

$$x''_2 = \frac{x'_{1r} + x'_{3l}}{2},$$

and

$$y_1 = \frac{x'_{1r} + x''_2}{2},$$

$$y_2 = \frac{x''_2 + x'_{3l}}{2}.$$

Thus

$$\min |e_{max}| = x''_2 - y_1 = y_2 - x''_2 < \frac{x_{max} - x_{min}}{2N}.$$

From Fig. 6.5(d), not all "zeros" has been masked out. Hence, Eq-6.22 should be modified as,

$$\frac{x_{max} - x_{min} - K'}{2N} \leq \min \{|e_{max}|\} \leq \frac{x_{max} - x_{min}}{2N}. \quad (6.25)$$

where K' is the number of "zeros" that have actually been masked out in the threshold dilation procedure. Eq-6.25 gives the new lower bound of $\min \{|e_{max}|\}$. The equality on the left side of Eq-6.25 holds only if, after the zero-masking (or threshold dilation), the unmasked interval can be partitioned into equal-length intervals.

The above example also gives an computationally efficient method to reduce $|e_{max}|$. For a sample interval $[x_{min}, x_{max}]$, the direct search for $\min \{|e_{max}|\}$ will take $C_{x_{max}-x_{min}+1}^N$ iterations, where

$$C_{x_{max}-x_{min}+1}^N = \frac{(x_{max} - x_{min} + 1)!}{N! (x_{max} - x_{min} + 1 - N)!}. \quad (6.26)$$

When $N = 4$ and $x_{\max} - x_{\min} + 1 = 40$,

$$C_{40}^4 = \frac{40 \times 39 \times 38 \times 37}{4 \times 3 \times 2} = 91,390.$$

Using the “zero-masking” scheme, only $N - 1 = 3$ iterations are needed.

In Eq-6.25, the larger K' is, the smaller the lower bound of $\min \{ |e_{\max}| \}$ could be. K' , however, is the number of “zeros” masked out. Thus, we should try to mask out as many “zeros” as possible.

Fig. 6.6 is a diagram of the “zero-masking” scheme. First, the $N - 1$ thresholds, x_n , are pre-set, equally-spaced, using Eq-6.9. Second, each threshold is dilated (zero masking) and the thickest one, x_t , which masked out most zeros is selected. Based on the x_t , the other $N - 2$ thresholds are re-allocated and, then, step 2 is repeated until all the thresholds have been allocated.

6.2.3 Performance of Minimax Error Multi-Level BTC

In this section, the performance of Mini-Max Error Multi-Level BTC is tested on CSI. Different block sizes are used and the results are compared with other quantizers.

One of the advantages of the “zero masking” quantizer level allocation scheme is that it will assign the exact value to overlay pixels in CSI. The overlay pixel level is normally quite distinct from surrounding satellite imagery pixels, see Picture I. Thus, the “zero masking” scheme will automatically mask out the distinguishing “zeros”, Fig. 6.7, and assign the exact value to overlay pixels.

Fig. 6.8 shows the MSE performance of various four level quantizers. Max quantizers [13,120], which are optimized to Laplacian, Gaussian and uniform distributions, and multi-level BTC using absolute moment preserving quantizers, are compared with the minimax multi-level BTC. Different block sizes are used. It can be seen, Fig. 6.8, that the two multi-level BTC schemes have much better MSE performance

Figure 6.6: Diagram of "zero-masking" scheme.

Figure 6.7: Zero-masking scheme implemented on a block with overlays.

than the Max quantizers, because they are adaptive to the local statistics. The minimax multi-level BTC has the lowest MSE. From the results of the Max quantizers, we can conclude that the best model of CSI over a pixel block is the uniform distribution. This, on the other hand, demonstrates the randomness of the CSI.

Table 6.1 (a) displays a piece of 8×8 original data of CSI. Table 6.1 (b) shows the reconstructed data using 4-level 8×8 BTC, where $MSE = 27$. In Table 6.1 (c), conventional 2-level 4×4 BTC is implemented, where $MSE = 52$.

Fig. 6.9 displays the results of implementing different quantizers on CSI with and without overlays. The MSE gain of CSI without overlays over CSI with overlays is minimum for minimax multi-level BTC. This shows that minimax multi-level BTC is less sensitive to changes in images statistics.

The most attractive feature of BTC is its computational simplicity. No computation at all is necessary for decoding of BTC data. LUTs may be used to assign

Figure 6.8: MSE performance comparison of different quantizers (4-level).

56	44	65	91	61	44	44	67
53	45	61	90	255	255	46	63
49	46	65	81	57	39	56	67
55	49	66	76	53	37	58	65
58	53	73	73	58	36	60	63
56	53	78	68	55	40	57	67
52	60	81	68	52	45	57	68
58	66	83	65	47	46	56	72

(a) Original data.

60	42	60	81	60	42	42	60
60	42	60	81	255	255	42	60
42	42	60	81	60	42	60	60
60	42	60	81	60	42	60	60
60	60	81	81	60	42	60	60
60	60	81	60	60	42	60	60
60	60	81	60	60	42	60	60
60	60	81	60	42	42	60	81

(b) 4-level 8×8 BTC, MSE = 27.

50	50	76	76	54	54	54	54
50	50	50	76	255	255	54	54
50	50	76	76	54	54	54	54
50	50	76	76	54	54	54	54
55	55	72	72	61	44	61	61
55	55	72	72	61	44	61	61
55	55	72	72	44	44	61	61
55	72	72	72	44	44	61	61

(c) 2-level 4×4 BTC, MSE = 52.Table 6.1: (a) Original data; (b) 4-level 8×8 BTC; (c) 2-level 4×4 BTC

Figure 6.9: MSE performance comparison using CSI with/without overlay.

the pixel values. The whole process can be implemented on a standard byte-oriented general purpose computer, as appropriate for widely disseminated products [7].

6.3 Quantization Levels Data Compression

In Minimax Error Multi-Level Block Truncation Coding, a 4-level (2-bit) quantizer is used to quantize the image data based on the minimax quantization error scheme within each pixel block. The quantized data set includes a quaternary plane which defines the quantization level for each pixel and four 8-bit quantization level codes for each block. The bit rate for $M \times M$ block size is

$$\text{Bit Rate} = 2 + \frac{4 \times 8}{M \times M} \text{ bits/pixel},$$

which, for $M = 8$, is 2.5 bits/pixel.

There are correlations between the quantization levels and within the quantization map. This section will discuss the further, lossless, compression of the quantization levels.

Second order predictors are used to remove the inter- and intra-block correlations between quantization levels. A variable length code is designed to code the four quantization level prediction errors losslessly. The algorithm is not sensitive to statistical variations in the data, has less encoding and decoding delay than the Huffman code, and is relatively robust to transmission or storage errors.

6.3.1 Intra-Block Prediction

For each quantization level y_n , where $n = 0, 1, 2, 3$, the sample space is $[0, 255]$, that is

$$\{y_n\} \in [0, 255], \quad (6.27)$$

Figure 6.10: Triangular distribution.

and within each block

$$y_0 \leq y_1 \leq y_2 \leq y_3. \quad (6.28)$$

Individually, each quantization level is specified by 8 bits. For a 4-level quantizer, 32 bits are required to represent all four of them. However, the quantization levels are related. For example, once y_0 is allocated on the sample space $[0, 255]$, the sample space for y_1 reduces to $[y_0, 255]$. Thus,

$$\{y_0\} \in [0, 255]. \quad (6.29)$$

$$\{y_1\} \in [y_0, 255]. \quad (6.30)$$

$$\{y_2\} \in [y_1, 255]. \quad (6.31)$$

$$\{y_3\} \in [y_2, 255]. \quad (6.32)$$

Assuming y_1 has a triangular distribution over $[y_0, 255]$, Fig. 6.10, since y_1 is more likely to be located close to y_0 ,

$$P\{y_1 | y_0\} = \frac{2 \cdot (255 - y_1)}{(255 - y_0)^2} \quad (6.33)$$

If y_0 is uniformly distributed over $[0, 255]$,

$$\begin{aligned} P\{y_1, y_0\} &= P\{y_1 | y_0\} \cdot P\{y_0\} \\ &= \frac{2 \cdot (255 - y_1)}{256 \cdot (255 - y_0)^2} \end{aligned} \quad (6.34)$$

The first order conditional entropy is

$$H\{y_1 | y_0\} = - \sum_{y_0} \sum_{y_1} P\{y_1, y_0\} \log_2 P\{y_1 | y_0\}. \quad (6.35)$$

Inserting Eq-6.33 and Eq-6.34 into Eq-6.35,

$$H\{y_1 | y_0\} = - \sum_{y_0=0}^{255} \sum_{y_1=y_0}^{255} \frac{2(255 - y_1)}{256(255 - y_0)^2} \cdot \log_2 \frac{2(255 - y_1)}{(255 - y_0)^2} = 6.33 \text{ bits}, \quad (6.36)$$

which is less than 8 bits. Similarly, other pair-wise conditional entropies are

$$H\{y_2 | y_1\} = H\{y_3 | y_2\} = H\{y_1 | y_0\} = 6.33 \text{ bits}. \quad (6.37)$$

From Eq-6.37, if entropy coding is used to code the quantization levels, the estimated bit rate will be about

$$\begin{aligned} H\{y_0\} + \sum_{n=1}^3 H\{y_n | y_{n-1}\} \\ = 8 + 3 \times 6.33 = 26.99 \text{ bits/block}, \end{aligned} \quad (6.38)$$

which is less than the original 32 bits.

One way to remove the correlation between quantization levels is to use Differential PCM coding on the differences between quantization levels. If y_0 is known, $y_1 - y_0$ gives all the information required to obtain y_1 . Assuming y_n and y_{n-1} are two random variables with the same variance $\sigma_n^2 = \sigma_{n-1}^2$, the variance of their difference is:

$$\sigma^2(y_n - y_{n-1}) = \sigma_n^2 - 2\rho_{n,n-1}\sigma_n\sigma_{n-1} + \sigma_{n-1}^2 = 2\sigma_n^2[1 - \rho_{n,n-1}], \quad (6.39)$$

where $\rho_{n,n-1}$ is the cross correlation coefficient of $\{y_n\}$ and $\{y_{n-1}\}$. For 8×8 block size,

$$\rho(7) \leq \rho_{n,n-1} \leq \rho(1),$$

CSI (with overlays)															
m	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$\rho(m)$	.80	.74	.70	.67	.64	.61	.59	.57	.55	.54	.52	.51	.50	.48	.47
CSI (without overlays)															
m	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$\rho(m)$	.90	.84	.79	.76	.72	.70	.67	.65	.63	.61	.59	.58	.56	.54	.53
Girl with Hat															
m	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$\rho(m)$	.97	.93	.89	.86	.83	.80	.78	.75	.73	.71	.69	.67	.65	.63	.62

Table 6.2: Values of Pixel Correlation Coefficients of Different Images.

where $\rho(m)$ are pixel correlation coefficients and m indicates pixel distance. Table 6.2 shows some pixel correlation coefficients of different test images. It can be seen that, for $m \leq 7$, $\rho(m) > 0.5$. For simplicity, substituting the average of $\rho(m)$, $m = 1, 2, \dots, 7$, which is

$$\rho_{n,n-1} = \begin{cases} 0.679 & \text{satellite image with overlay} \\ 0.769 & \text{satellite image without overlay} \\ 0.866 & \text{the girl image} \end{cases} \quad (6.40)$$

into Eq-6.40 yields,

$$\sigma^2(y_n - y_{n-1}) = \begin{cases} 0.642 \sigma_n^2 & \text{satellite image with overlay} \\ 0.462 \sigma_n^2 & \text{satellite image without overlay} \\ 0.268 \sigma_n^2 & \text{the girl image.} \end{cases} \quad (6.41)$$

In all cases, $\sigma^2(y_n - y_{n-1}) < \sigma_n^2$, which means $y_n - y_{n-1}$ has less dynamic range than y_n (note: the girl's image has much less dynamic range than CSI).

In information theory [78,113],

$$\text{Entropy} \propto \log_2 \sigma$$

Thus, from Eq-6.41, $y_n - y_{n-1}$ has less entropy than y_n .

For the composite satellite imagery, the calculated results are

$$\begin{aligned} H\{y_1\} &= 6.41 \text{ bits,} \\ H\{y_1 - y_0\} &= 5.33 \text{ bits,} \\ H\{y_2 - y_1\} &= 5.11 \text{ bits,} \\ H\{y_3 - y_2\} &= 5.62 \text{ bits.} \end{aligned} \tag{6.42}$$

Comparing Eq-6.42 and Eq-6.37, it can be seen that the measured differential entropies $H\{y_n - y_{n-1}\}$ are lower than the theoretically deduced conditional entropies $H\{y_n | y_{n-1}\}$. This is because that, to obtain Eq-6.37, we assumed the sample intervals for y_{n-1} and y_n , where $n = 1, 2, 3$, are $[0, 255]$ and $[y_{n-1}, 255]$ respectively, while the sample intervals for $y_n - y_{n-1}$ in the real case are much less than that. In Eq-6.42, $H\{y_1 - y_0\}$ and $H\{y_3 - y_2\}$ have slightly higher entropies than $H\{y_2 - y_1\}$. This is the effect of overlay pixels, since whenever there are overlays on a block, either or both y_0 and y_3 will automatically be allocated to represent the overlays. This will result in wider sample intervals of $y_3 - y_2$ and $y_1 - y_0$, and, thus, increase the entropies.

From Eq-6.42, the lower bound on the number of bits required to represent the quantization levels is

$$6.41 + 5.33 + 5.11 + 5.62 = 22.47 \text{ bits per block.} \tag{6.43}$$

6.3.2 Inter-Block Prediction

So far, differential encoding has been used to remove the intra-block correlations between quantization levels. In Eq-6.38 there is, however, a first order entropy, $H\{y_0\}$, which has much higher value than the differential entropies. Inter-block correlation may be used to further reduce the entropy.

Assume blocks k and $k - 1$ are two adjacent blocks; $y_{m,k}$ and $y_{m,k-1}$, where $m = 0, 1, 2, 3$, are the quantization levels on blocks k and $k - 1$. Substituting into

Eq-6.39,

$$\sigma^2(y_{m,k} - y_{m,k-1}) = 2\sigma_{y_k}^2(m)[1 - \rho_{k,k-1}(m)], \quad (6.44)$$

Here, for 8×8 block size,

$$\rho(15) \leq \rho_{k,k-1} \leq \rho(1), \quad (6.45)$$

Following the same step as in Eq-6.41

$$\sigma^2(y_{m,k} - y_{m,k-1}) = \begin{cases} 0.82 \sigma_{y_{m,k}}^2 & \text{satellite image with overlay} \\ 0.66 \sigma_{y_{m,k}}^2 & \text{satellite image without overlay} \\ 0.47 \sigma_{y_{m,k}}^2 & \text{the girl image} \end{cases} \quad (6.46)$$

Again, in all cases, $\sigma^2(y_{m,k} - y_{m,k-1}) < \sigma_{m,k}^2$, which means that there will be coding gain by using inter-block differential coding.

Comparing Eq-6.41 with Eq-6.46,

$$\sigma^2(y_n - y_{n-1}) < \sigma^2(y_{m,k} - y_{m,k-1}) < \sigma_n^2, \quad (6.47)$$

which implies that the inter-block coding gain will be less than the intra-block gain. From Eq-6.46, satellite images with overlays are expected to have less coding gain than those without overlays.

In our minimax error multi-level block truncation coding, the two outermost quantization levels (y_0 and y_3) are most influenced by overlays. As mentioned before, whenever there are overlays on a block, either or both outermost quantization levels will be used to represent the overlays by our threshold dilation scheme, while the other quantization levels are less effected by the overlays. Therefore, for four level BTC,

$$\sigma^2(y_{m,k} - y_{m,k-1}) = \begin{cases} 0.82 \sigma_{y_{m,k}}^2 & m = 0, 3 \\ 0.66 \sigma_{y_{m,k}}^2 & m = 1, 2. \end{cases} \quad (6.48)$$

Thus, inter-block prediction on y_1 or y_2 can achieve higher coding gain than on y_0 or y_3 . For composite images, the test results are

$$\begin{aligned} H\{y_{0,k} - y_{0,k-1}\} &= 6.00 \text{ bits,} \\ H\{y_{1,k} - y_{1,k-1}\} &= 5.68 \text{ bits,} \\ H\{y_{2,k} - y_{2,k-1}\} &= 5.88 \text{ bits,} \\ H\{y_{3,k} - y_{3,k-1}\} &= 6.64 \text{ bits.} \end{aligned} \quad (6.49)$$

Comparing the results of Eq-6.42 and Eq-6.49 confirms that inter-block coding gain is less than the intra-block gain. Meanwhile, as indicated by Eq-6.48, the outermost quantization levels have higher entropy, or less coding gain.

From Eq-6.42 and Eq-6.49, the lower bound of the number of bits required to inter-block code y_1 and intra-block code the other levels is:

$$5.33 + 5.68 + 5.11 + 5.62 = 21.74 \text{ bits.} \quad (6.50)$$

A gain of 0.73 bits over Eq-6.43 is achieved. Tests on other images also show coding gains. For the girl's image, the gain is 0.48 bits.

In Eq-6.49, quantization levels in previous blocks are used to differentially code the present ones, which actually is first-order prediction. Usually, it is advantageous to implement a second order predictor. It not only will achieve a coding gain but also introduce an error "leakage" [13] as described in the next paragraph.

The diagram of a second-order, two-dimensional, predictor is shown in Fig. 6.11 (note: since this is a lossless coding algorithm, the prediction error is quantized losslessly, an "all-zero" predictor is used which does not have "error feedback", because there is no quantization "error"). If two previous blocks, the block on left, $y_m(k_v, k_h - 1)$, and the block on top of the present block, $y_m(k_v - 1, k_h)$ are used to predict the present block, $y_m(k_v, k_h)$, which may be expressed as

$$y_m(k_v, k_h) = e_m(k_v, k_h) + [y_m(k_v - 1, k_h) + y_m(k_v, k_h - 1)]/2,$$

Figure 6.11: 2nd order 2-dimensional predictor.

where e_m is the prediction error. Assuming transmission or storage error noise is involved in either $y_m(k_v - 1, k_h)$ or $y_m(k_v, k_h - 1)$,

$$\begin{aligned}
 y'_m(k_v, k_h) &= e_m(k_v, k_h) + \frac{y_m(k_v - 1, k_h) + y_m(k_v, k_h - 1) + \text{noise}}{2} \\
 &= e_m(k_v, k_h) + \frac{y_m(k_v - 1, k_h) + y_m(k_v, k_h - 1)}{2} + \frac{\text{noise}}{2} \\
 &= y_m(k_v, k_h) + \frac{\text{noise}}{2}
 \end{aligned}$$

It can be seen that noise will be attenuated by half [13]. Thus, for an 8-bit image, any reconstructed image can recover from the error within 8 prediction steps.

The coding gain of a two-dimensional predictor over one-dimensional is 0.20 bit for composite satellite images. The total prediction error entropy is:

$$5.33 + 5.48 + 5.11 + 5.62 = 21.54 \text{ bits.} \quad (6.51)$$

The coding gain is not very high because of the relatively low inter-block correlation. Therefore, it is not worth it to implement predictors of higher order than second for the inter-block prediction, since the computational complexity and memory space will increase and very small coding gain will be achieved. Test results support this argument. For composite satellite image the coding gain is only 0.04 bits for a third order predictor.

6.3.3 Higher Order Intra-Block Prediction

Eq-6.35 and Eq-6.36 show the first order conditional probability and entropy. In information theory, the conditional entropy of order n

$$H\{x_m \mid x_{m-1}, x_{m-2}, \dots, x_{m-n}\}$$

of a process x_m is the uncertainty about its present state under the assumption that its n most recent values have been observed. It can be shown [113] that higher order conditional entropy is always less than or equal to lower order,

$$H\{x_m \mid x_{m-1}, x_{m-2}, \dots, x_{m-n}, x_{m-n-1}\} \leq H\{x_m \mid x_{m-1}, x_{m-2}, \dots, x_{m-n}\}. \quad (6.52)$$

If y_1 and y_2 are known, it should be possible to better predict y_3 .

In our minimax error BTC, not considering the effects of the threshold dilation scheme, the quantization levels should be equally spaced. At this time, if two quantization levels, e.g., y_1 and y_2 , are known, we can calculate the exact values of y_0 and y_3 , which are

$$\begin{aligned} y_0 &= y_1 - (y_2 - y_1) & \text{and} \\ y_3 &= y_2 + (y_2 - y_1); \end{aligned} \quad (6.53)$$

or

$$\begin{aligned} y_1 - y_0 - (y_2 - y_1) &= 0 & \text{and} \\ y_3 - y_2 - (y_2 - y_1) &= 0. \end{aligned} \quad (6.54)$$

Figure 6.12: Symmetric triangular distribution.

With the effects of threshold dilation, we have

$$\begin{aligned} y_1 - y_0 - (y_2 - y_1) &= e_0 \quad \text{and} \\ y_3 - y_2 - (y_2 - y_1) &= e_3. \end{aligned} \quad (6.55)$$

where e_0 and e_3 are prediction errors.

For simplicity, assume e_3 has a symmetric triangular distribution, see Fig. 6.12,

$$\{e_3\} \in [y_2, 3y_2 - 2y_1], \quad (6.56)$$

$$P\{y_3 | y_2, y_1\} = \begin{cases} \frac{y_3 - y_2}{(y_2 - y_1)^2} & y_3 \leq 2y_2 - y_1 \\ \frac{-y_3 + 3y_2 - 2y_1}{(y_2 - y_1)^2} & y_3 > 2y_2 - y_1. \end{cases} \quad (6.57)$$

From Eq-6.34,

$$\begin{aligned} P\{y_2, y_1\} &= P\{y_2 | y_1\} P\{y_1\} \\ &= \frac{2(255 - y_2)}{256(255 - y_1)^2}. \end{aligned} \quad (6.58)$$

Since

$$H\{y_3 | y_2, y_1\} = - \sum_{y_1} \sum_{y_2} \sum_{y_3} P\{y_3, y_2, y_1\} \log_2 P\{y_3 | y_2, y_1\}, \quad (6.59)$$

inserting Eq-6.57 and Eq-6.58,

$$H\{y_3 | y_2, y_1\} =$$

$$= \begin{cases} - \sum_{y_1=0}^{255} \sum_{y_2=y_1}^{255} \sum_{y_3=y_2}^{3y_2-2y_1} \frac{y_3 - y_2}{(y_2 - y_1)^2} \frac{2(255 - y_2)}{256(255 - y_1)^2} \log_2 \frac{y_3 - y_2}{(y_2 - y_1)^2}, & \text{for } y_3 \leq 2y_2 - y_1, \\ - \sum_{y_1=0}^{255} \sum_{y_2=y_1}^{255} \sum_{y_3=y_2}^{3y_2-2y_1} \frac{-y_3 + 3y_2 - 2y_1}{(y_2 - y_1)^2} \frac{2(255 - y_2)}{256(255 - y_1)^2} \log_2 \frac{-y_3 + 3y_2 - 2y_1}{(y_2 - y_1)^2}, & \text{for } y_3 > 2y_2 - y_1, \end{cases}$$

$$= 3.51 \text{ bits},$$

(6.60)

which is an 1.88 bit gain over Eq-6.36. Similarly,

$$H\{y_0 | y_1, y_2\} = H\{y_2 | y_1, y_0\} = H\{y_3 | y_2, y_1\} = 3.51 \text{ bits}.$$

From Eq-6.55, following the same procedure of Eq-6.39 to Eq-6.41, we can also prove that e_0 and e_3 , the second order prediction errors, have smaller variances than those in Eq-6.41, the first order prediction errors, and, therefore, have smaller entropies. The entropy measurement also confirms that,

$$\begin{array}{lll} H(e_0) = 3.97, \text{ bits} & H(e_3) = 4.41, \text{ bits} & \text{for composite satellite imagery;} \\ H(e_0) = 3.25, \text{ bits} & H(e_3) = 3.24, \text{ bits} & \text{for the girl image.} \end{array} \quad (6.61)$$

Comparing Eq-6.60 and Eq-6.61 shows that the experimental measurements are close to the theoretically deduced result. (Note: Eq-6.61 also shows that threshold dilation scheme works well, otherwise the prediction error $e_n = 0$, Eq-6.54, and $H(e_n) = 0$.)

The total measured prediction error entropy now is

$$\begin{aligned} 3.97 + 5.48 + 5.11 + 4.41 &= 18.97 \text{ bits} && \text{for composite satellite imagery;} \\ 3.25 + 5.79 + 4.58 + 3.24 &= 16.86 \text{ bits} && \text{for the girl image.} \end{aligned} \quad (6.62)$$

The coding gains over Eq-6.51 are 2.57 and 2.83 bits respectively.

6.3.4 Coding of Prediction Errors

We have, so far, successfully reduced the entropies by implementing intra- and inter-block prediction coding. The next step is to losslessly code the prediction errors. Eq-6.62 gives the lower bound of the bit rate. The most straightforward way is to entropy code, i.e., Huffman code, the prediction errors, e_n , $n = 0, 1, 2, 3$, individually, which will result in a bit rate quite close to Eq-6.62. However, the penalties of using an entropy code are:

1. The code cannot recover from transmission and storage errors (some "markers" must be used which will increase the bit rate);
2. Entropy coding has long coding delays. For encoding, the whole image has to be gone through once to obtain the statistics which are needed to generate the code book; for decoding, the coded data has to be read in bit-by-bit to follow the code tree, which makes the decoding process very slow running on a general purpose computer;
3. A code book has to be stored or transmitted for decoding. For a 512 by 512 image, 8×8 block size and 4 level quantizer, each prediction error has a dynamic range $[-255, +255]$, or 511 possible code book entries, requiring 9 bits. If Huffman coding is used, the longest codeword might be more than 9 bits. On the other hand, in any practical situation the maximum code length will be

limited by the hardware configuration [114–118], e.g. to 16 bits. Assuming the maximum code length is 16 bits, then each code book entry will have 16 bits, and

$$\text{Average overhead bits per block} = \frac{511 \times 16 \times 4 \times 8 \times 8}{512 \times 512} \simeq 8.0 \text{ bits/block.} \quad (6.63)$$

Our solution is to develop a sub-optimal code which has fixed length overhead for each block and variable size code words. According to Shannon's rate-distortion theory, a better performance is always achievable in theory by coding vectors instead of scalars, even though the the data source is memoryless [78]. A 4-dimensional quantizer is used to quantize the prediction errors, $\{\mathbf{e}_0, \mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$ simultaneously, which results in a saving of overhead bits.

An bit allocation algorithm is used to efficiently allocate code bits. The allocation algorithm has the following steps:

Step 1: For a $L \times L$ block image, calculate the code-bit vector $\{\mathbf{b}_0, \mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3\}_{L \times L}$, where b_n , $n = 0, 1, 2, 3$, is the number of bits required to represent each prediction error $\{\mathbf{e}_n\}$, and

$$b_n = 1 + \log_2 |\mathbf{e}_n| \text{ bits.}$$

The added 1 bit is a sign bit.

Step 2: Map each vector $\{\mathbf{b}_0, \mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3\}_{L \times L}$ into a 4-dimensional space $\{E_b\}_{4D}$,

$$\{ E_b[\mathbf{b}_0][\mathbf{b}_1][\mathbf{b}_2][\mathbf{b}_3] \}.$$

Step 3: Search for the optimal bit allocation

$$\{n_0, n_1, n_2, n_3\},$$

where the number of bits in the code, $N = n_0 + n_1 + n_2 + n_3$, and

$$\{E_N\}_{4D} \in \{E_b\}_{4D},$$

and calculate

$$M = \max (\sum_{n_0, n_1, n_2, n_3} E_N[n_0][n_1][n_2][n_3]).$$

Step 4: If $M > (L \times L)/20$ and $\{n_0, n_1, n_2, n_3\}$ is not an entry in the bit allocation map, enter it into the bit allocation map and go back to Step 3. Otherwise continue.

Step 5: $N = N + 1$, prepare for next search.

Step 6: If $N < 32$ go to Step 3, search for next bit allocation vector. Otherwise continue.

Step 7: Stop.

The flow-chart of the algorithm is given in Fig. 6.13.

The bit allocation program was run on a series of satellite images and standard images, such as the girl's image. The resulting bit allocation maps are very similar. Table 6.3 is a typical bit allocation map.

From the bit allocation map in Table 6.3, more bits are allocated to e_1 and e_2 than to the others, e_0 and e_3 , because they usually have wider dynamic range (note: e_1 is the output of the inter-block predictor where there is less correlation and e_2 is from a first order predictor, while the others, e_0 and e_3 , are from second order predictors). In Table 6.3, overhead code 1111_b is actually an escape sequence for those blocks with ill-distributed quantization levels where prediction fails and the error dynamic range is more than 8 bits. Since each prediction error has a 9-bit dynamic range, $[-255, +255]$,

Figure 6.13: Flowchart of the optimal bit allocation algorithm.

multi-dimension variable block code bit allocation map						
No.	overhead code (binary)	bit allocation map				code bits & overhead bits
		e_0	e_1	e_2	e_3	(per block)
1	$(0000)_b$	2	3	3	2	10
2	$(0001)_b$	2	4	3	3	12
3	$(0010)_b$	1	2	8	1	12
4	$(0011)_b$	3	3	4	2	12
5	$(0100)_b$	3	4	4	3	14
6	$(0101)_b$	2	5	4	3	14
7	$(0110)_b$	3	5	4	4	16
8	$(0111)_b$	4	4	5	3	16
9	$(1000)_b$	3	6	5	4	18
10	$(1001)_b$	4	4	5	5	18
11	$(1010)_b$	4	6	5	5	20
12	$(1011)_b$	5	6	6	5	22
13	$(1100)_b$	3	5	5	9	22
14	$(1101)_b$	5	7	6	6	24
15	$(1110)_b$	9	7	5	3	24
16	$(1111)_b$	8	8	8	8	32
error entropy (CSI)		4.0	5.5	5.1	4.4	19.0 bits
error entropy (girl)		3.2	5.8	4.6	3.2	16.8 bits

Table 6.3: Multi-dimension variable block code bit allocation map.

images	error entropies				total entropy (per block)	original bits (per block)	code bits (per block)	code bits & overhead bits
	e_0	e_1	e_2	e_3				
CSI (visual)	4.0	5.5	5.1	4.4	19.0	32	16.4	20.4
CSI (no OL)	3.5	5.3	4.9	3.9	17.6	32	15.1	19.1
CSI (Nir)	4.0	5.5	5.1	4.3	18.9	32	16.4	20.4
CSI (4mic)	3.6	5.3	5.3	4.4	18.6	32	16.2	20.2
CSI (11mic)	3.7	5.3	4.9	4.0	17.9	32	15.6	19.6
CSI (12mic)	3.9	5.3	5.0	4.3	18.5	32	16.1	20.1
Girl	3.2	5.8	4.6	3.2	16.8	32	15.2	19.2

Table 6.4: Quantization level prediction error entropies and bit rates.

the maximum possible number of code bits for the four prediction errors is 36 bits, which is more than the original 32 bits. Thus, when the number of code bits required is more than 32, it is better to send the original code, which not only saves bits, but also saves decoding time. The four bytes, 32 bits, after 1111, are used to represent the four quantization levels. Code 13, {3, 5, 5, 9}, and code 15, {9, 7, 5, 3}, are for blocks with white and black overlays respectively. Code 3, {1, 2, 8, 1}, is mainly for bi-level blocks. From the last two rows of Table 6.3, the error entropies of $H(e_1)$ and $H(e_2)$ are higher than $H(e_0)$ and $H(e_3)$. Thus, there are more bits allocated to quantize e_1 and e_2 than e_0 and e_3 .

Table 6.4 shows quantization level prediction error entropies and code bit rates for different images, where the bit allocation map in Table 6.3 is used. The first and second rows show the data for visual-band CSI with and without overlays. The third to sixth rows are for CSIs of different spectrum and the last row is for the Girls image. From Table 6.4, visual band CSI has the highest entropy and bit rate. It can be seen that the existence of overlays increases the entropy and, thus, the bit rate by 1.3 bits/block. Comparing the second and last rows, even CSI without overlays has a higher entropy than the “standard” image.

code bits per block	CSI Vis	CSI Nir	CSI 4mic	CSI 11mic	CSI 12mic	girl
optimized bit allocation map	16.4	16.3	16.0	15.5	16.0	14.9
bit allocation map of CSI Vis	16.4	16.4	16.2	15.6	16.1	15.2

Table 6.5: Bit rate change due to statistical mis-match.

Although the bit allocation map in Table 6.3 is originally optimized to code visual-band CSI, it works well for other images. Table 6.5 shows the test results of code bits per block of different images using a statistically optimized bit allocation map and a bit allocation map optimized to visual-band CSI. It can be seen that the bit rate increase due to statistical mis-match is within 0.3 bits/block, or 0.0047 bits/pixel.

Another interesting result is that, comparing total entropy and code bits in Table 6.4, the number of code bits is actually less than the entropy. This is because the multi-dimensional quantizer provides a simultaneous quantization of four quantization levels, while the entropy measures the four prediction errors separately.

The overhead of our algorithm is 4 bits/block, which equals only 1 bit for each prediction error. From Eq-6.63, the overhead is about 8 bits/block for a 512 by 512 image if a Huffman code is used to code the prediction errors.

Table 6.6 displays the code distribution for visual-band CSI. From the last column of Table 6.6, the first 6 bit allocation maps cover 50 % of the total number of blocks which means that prediction error vectors usually have small magnitude. From the eighth column of Table 6.6, it can be seen that each bit allocation map entry has a different frequency. It is possible to implement entropy coding on the overhead code to further reduce the bit rate; but that will increase the encoding and decoding delay, and the coding gain is less than 0.15 bits/block, or 0.00234 bits/pixel for Table 6.5.

For our coding algorithm, as long as no errors occur in the overhead bits, effects of errors in the inter-block code attenuate through “error leakage” and errors in the

No.	bit allocation				number of blocks	total blocks	percentage (%)	accumulate (%)
	e_0	e_1	e_2	e_3				
1	2	3	3	2	2369	2369	15.07 %	15.07 %
2	2	4	3	3	1410	3779	8.97 %	24.04 %
3	1	2	8	1	524	4303	3.33 %	27.37 %
4	3	3	4	2	968	5271	6.16 %	33.53 %
5	3	4	4	3	1603	6874	10.20 %	43.73 %
6	2	5	4	3	999	7873	6.36 %	50.09 %
7	3	5	4	4	901	8774	5.73 %	55.82 %
8	4	4	5	3	930	9704	5.96 %	61.78 %
9	3	6	5	4	1492	11196	9.49 %	71.27 %
10	4	4	5	5	446	11642	2.84 %	74.11 %
11	4	6	5	5	616	12258	3.92 %	78.03 %
12	5	6	6	5	935	13193	5.95 %	83.98 %
13	3	5	5	9	697	13890	4.43 %	88.41 %
14	5	7	6	6	456	14346	2.90 %	91.31 %
15	9	7	5	3	558	14904	3.55 %	94.86 %
16	8	8	8	8	816	15720	5.19 %	100.0 %

Table 6.6: Code distribution for CSI (visual band).

intra-block code do not propagate into the next block. As well there is a possibility that recovery from error hits in the overhead bits may occur. It can be seen with reference to the second and seventh columns of Table 6.3, that if the overhead $(0100)_b$ of code 5 is hit by an error and changed into $(0101)_b$, the overhead of code 6, the decoding process will not loss synchronism since code 5 and code 6 have the same number of bits and, thus, the error is recoverable. Statistically, overhead code occupies $1/5$ of the total bits. Assuming the transmission error probability is p , the probability P_e that the 4-bit overhead code is hit by error is

$$P_e = \sum_{m=1}^4 C_4^m \cdot p^m = 1 - (1 - p)^4. \quad (6.64)$$

Omitting the high order terms,

$$P_e \approx C_4^1 \cdot p = 4p. \quad (6.65)$$

For codes $(0100)_b$ and $(0101)_b$, the unrecoverable error probability is about $3p$. The total unrecoverable error probability is

$$P_e(\text{total}) = P(\text{overhead}) \cdot \sum_{n=1}^{16} \frac{a_n p \cdot P(n)}{4}, \quad (6.66)$$

where $P(n)$ is the code occurrence frequency (8th column of Table 6.6), $P(\text{overhead})$ is the occurrence of overhead (about $1/5$) and $a_n p$ specifies the unrecoverable error probability of each overhead code, e.g., $3p$ for code $(0100)_b$. For the composite satellite images,

$$P_e(\text{total}) = 0.17 \cdot p. \quad (6.67)$$

From Eq-6.67, the unrecoverable error probability is about 5.9 times lower than that of a Huffman code, which is essentially p .

In this section a lossless coding algorithm has been developed to compress the quantization levels of the minimax error 4-level BTC coding. Second order predictors

are used to remove the inter- and intra-block correlations between quantization levels. A variable length code is designed to 4-dimensionally code the four quantization level prediction errors losslessly. The algorithm is not sensitive to the statistical variations in the data, has shorter encoding and decoding delay, and is relatively robust to transmission and storage errors. The compression ratio is about 2 : 1 (not including the 4 bits overhead), or 1.6 : 1 (including the overhead).

6.4 Lossless Quaternary Plane Data Compression

Multi-Level Block Truncation Coding can be understood as block-wise adaptive quantization. A four-level quantizer, Fig. 6.14(a), is used to adaptively quantize the input data based on minimax error algorithm. The quantized data set includes a 2-bit quaternary plane which defines the quantization levels for each pixel and four 8-bit quantization levels for each block. In the last section, we discussed compression of the quantization levels data. This section will introduce an algorithm which further compresses the quaternary plane. The correlation of quaternary plane elements is evaluated mathematically which shows that there is redundancy between quaternary plane elements and that most of the redundancy in the plane is among very close neighbours. Based on this result, a 4-element packing scheme is used to map the quaternary plane into a byte-oriented data set which, then, is coded using a variable length block code.

6.4.1 Correlation of Quaternary Plane Elements

As mentioned before, the input image data can be modeled as a sample of a first-order Markov process for which the correlation between pixels is proportional to their geometric separation. The covariance function for the 2-dimensional Markov process

is [33]

$$R_{x,y}(\tau_x, \tau_y) = R(0,0) \cdot \exp\{-\sqrt{\alpha_x^2 \tau_x^2 + \alpha_y^2 \tau_y^2}\} \quad (6.68)$$

where $R(0,0)$ is an energy scaling constant and α_x and α_y are scaling constants. To simplify the problem 1-dimensional correlation is used and its normalized form, the correlation coefficient, is

$$\rho(\tau) = \frac{R_x(\tau_x)}{R(0)} = e^{-\alpha \cdot \tau}, \quad (6.69)$$

where $\alpha = -\ln \rho(1)$ ranges between 0.05 and 0.3 [47]. For CSI $\alpha = -\ln 0.90 = 0.1$.

However, direct analysis of the autocorrelation coefficients of 4-level truncated data is mathematically difficult. To simplify the problem, three 2-level quantizers, or hard limiters, Fig. 6.15(a), are used to model the truncation procedure, Fig. 6.14(b). The hard limiter **A** decides the state of the left bit of output, while **B** and **C** decide the right bit. The sample space of the truncated data is { 00, 01, 10, 11 }.

It can be understood that the state of the left bit is totally independent of the right bit, thus, **A** is independent of **B** and **C**. Meanwhile, **B** and **C** are mutually independent. Based on these facts, Fig. 6.14(b) can be further simplified as a hard limiter **D** followed by a random number generator $n(t)$, Fig. 6.14(c). The output of the hard limiter, $d(t)$, is independent of $n(t)$. It should be noted that the system in Fig. 6.14(c) is built for purely mathematical purposes, the signals in it have the same sample space, mean and autocorrelation function as the system in Fig. 6.14(b), but it might have a totally different output data stream.

In Fig. 6.14(c), the hard limiter **D** has a output sample space $\{-2, +2\}$ and $n(t)$ is a binary random telegraph signal, Fig. 6.15(b) (note: the telegraph signal $n(t)$ is a binary random process formed based on Poisson points t_i such that $n(t) = 1$ if the number of points in the interval $(0, t)$ is even, and $n(t) = -1$ if the number is odd

Figure 6.14: Mathematic model of a 4-level truncator.

Figure 6.15: (a) Hard limiting model (b) Poisson process and telegraph signal.

[113].) which has sample space $\{0, 1\}$ and autocorrelation function

$$R_n(\tau) = e^{-2\alpha\tau}. \quad (6.70)$$

If the input data is a Gaussian process with correlation as shown in Eq-6.70, the correlation coefficient of the output of hard limiter $\mathbf{d}(t)$ is [113]

$$\rho_d(\tau) = \frac{2}{\pi} \sin^{-1}(e^{-\alpha|\tau|}). \quad (6.71)$$

The output of Fig. 6.14(c) is

$$\mathbf{e}(t) = \mathbf{d}(t) + \mathbf{n}(t), \quad (6.72)$$

and its autocorrelation function is

$$\begin{aligned} R_e(\tau) &= E\{[\mathbf{d}(t) + \mathbf{n}(t)][\mathbf{d}(t + \tau) + \mathbf{n}(t + \tau)]\} \\ &= E\{\mathbf{d}(t) \cdot \mathbf{d}(t + \tau)\} + E\{\mathbf{n}(t) \cdot \mathbf{n}(t + \tau)\} \\ &\quad + E\{\mathbf{d}(t) \cdot \mathbf{n}(t + \tau)\} + E\{\mathbf{n}(t) \cdot \mathbf{d}(t + \tau)\}. \end{aligned} \quad (6.73)$$

Since $\mathbf{d}(t)$ and $\mathbf{n}(t)$ are independent, the last two terms are equal to zero. Thus,

$$\begin{aligned} R_e(\tau) &= E\{\mathbf{d}(t) \cdot \mathbf{d}(t + \tau)\} + E\{\mathbf{n}(t) \cdot \mathbf{n}(t + \tau)\} \\ &= R_d(\tau) + R_n(\tau). \end{aligned} \quad (6.74)$$

Inserting Eq-6.70 and Eq-6.71,

$$R_e(\tau) = \frac{2}{\pi} \sin^{-1}(e^{-\alpha|\tau|}) + e^{-2\alpha\tau}. \quad (6.75)$$

Normalizing to $R_e(0) = 2$,

$$\rho_e(\tau) = \frac{R_e(\tau)}{R_e(0)} = \frac{1}{\pi} \sin^{-1}(e^{-\alpha|\tau|}) + \frac{1}{2}e^{-2\alpha\tau}. \quad (6.76)$$

The first term of Eq-6.76 comes from hard limiter **D** and second term from $n(t)$. It can be proved that

$$\frac{1}{2}e^{-2\alpha\tau} \geq \frac{1}{\pi} \sin^{-1}(e^{-\alpha|\tau|}). \quad (6.77)$$

PROOF: let

$$F(\tau) = \frac{1}{2}e^{-2\alpha\tau} - \frac{1}{\pi} \sin^{-1}(e^{-\alpha|\tau|}). \quad (6.78)$$

Taking derivative

$$\begin{aligned} F'(\tau) &= -\alpha e^{-2\alpha\tau} + \frac{\alpha}{\pi} \cdot \frac{e^{-\alpha\tau}}{\sqrt{1 - e^{-2\alpha\tau}}} \\ &= -\alpha e^{-2\alpha\tau} \left(1 - \frac{1}{\pi} \cdot \frac{1}{\sqrt{1 - e^{-2\alpha\tau}}}\right). \end{aligned} \quad (6.79)$$

From Eq-6.79,

$$F'(\tau) \leq 0, \quad (6.80)$$

over the interval $[1, \infty)$, which means $F'(\tau)$ does not change polarity and $F(\tau)$ is uniquely decreasing over the interval $[1, \infty)$. In Eq-6.78,

$$F(\infty) = 0. \quad (6.81)$$

From Eq-6.80 and Eq-6.81, we can draw the conclusion that

$$F(\tau) = \frac{1}{2}e^{-2\alpha\tau} - \frac{1}{\pi} \sin^{-1}(e^{-\alpha|\tau|}) \geq 0, \quad (6.82)$$

which proves Eq-6.77.

END OF PROOF

Adding $\frac{1}{\pi} \sin^{-1}(e^{-\alpha\tau})$ to both sides of Eq-6.77,

$$\frac{1}{\pi} \sin^{-1}(e^{-\alpha\tau}) + \frac{1}{2}e^{-2\alpha\tau} \geq \frac{2}{\pi} \sin^{-1}(e^{-\alpha|\tau|}), \quad (6.83)$$

which is equivalent to

$$\rho_c(\tau) \geq \rho_d(\tau) \quad (6.84)$$

or

$$\rho_{4-level}(\tau) \geq \rho_{2-level}(\tau). \quad (6.85)$$

Since

$$e^{-\alpha \cdot \tau} \geq e^{-2\alpha \cdot \tau}, \quad (6.86)$$

from Eq-6.69, Eq-6.77 and Eq-6.85,

$$\rho_{raw\ data}(\tau) \geq \rho_{4-level}(\tau) \geq \rho_{2-level}(\tau). \quad (6.87)$$

Eq-6.87 shows that the correlation of 4-level truncated data is higher than that of 2-level truncated, but less than that of the input data.

It can be seen, Fig. 6.16, that correlation of 4-level truncated data decays much faster than that of the input data. It also indicates that most of the redundancy in the quaternary plane is among very close neighbours. This is an important feature of the quaternary plane.

Because of the rapidly decaying correlation, we cannot use run length coding [29] to losslessly compress the quaternary plane, since there are not many long runs. Meanwhile, the entropy of the quaternary plane H_b is as high as 1.96 bit/element, so that directly implementing entropy coding cannot obtain much coding gain.

However, as mentioned before, there is correlation among close neighbours and this correlation tends to be isotropic. If we partition the quaternary plane into very small blocks, e.g., 2×2 , and pack each block into a byte, we form another data set, in which each byte actually is a 4-dimensional vector and, intuitively, there should be some correlations between those vector bytes.

Figure 6.16: Correlation functions of raw and truncated data.

Figure 6.17: 4-dimensional vector-byte file histogram.

6.4.2 Lossless Coding of 4-dimensional Vector Pattern

The motivation for mapping the quaternary plane into a 4-dimensional vector-byte file can also be traced back to rate distortion theory, which indicates that better performance can be achieved by coding vectors, instead of scalars.

Fig. 6.17 shows the histogram of a byte-vector file. It can be seen that some of the vector components have much higher frequency than others, which means that some of the 2×2 patterns occur much more often in the quaternary plane, because of the existence of local correlations. The uneven and spiky distribution of the histogram indicates that entropy coding can be used to compress the 4-element packed data. The entropy of the histogram in Fig. 6.17 is 6.07 bits/vector or 1.52 bits/element, which is less than the previous 1.96 bits/element. Table 6.7 and Table 6.8 list some high probability patterns of CSI and the Girl image, where the patterns are sorted by the order of their appearance frequency.

In Table 6.7 and Table 6.8, the patterns are classified into 4 groups:

- Group 1: the first 4 high probability patterns.
- Group 2: the next 16 patterns after Group 1.
- Group 3: the next 32 patterns after Group 2.
- Group 4: the rest of patterns.

There are a total of 256 possible patterns. Among them, patterns in Group 1, Group 2 and Group 3 each cover approximately 25 % of the total probability, Table 6.7 and Table 6.8. This feature repeats on many test images, i.e. satellite images, SAR radar images, TV or teleconferencing images. Based on the fact, a variable block entropy code is designed to encode the 4-element patterns, Fig. 6.18.

A 2-bit header, or prefix, is used to indicate the group number from 1 to 4. After the *header-code*, a variable length *pattern-code* is used to represent the patterns. For Group 1 patterns, a 2-bit pattern-code is used to indicate the four different patterns. For Groups 2, 3, and 4, the pattern-codes are 4-bit, 5-bit and 8-bit respectively. It can be seen, Fig. 6.18, that the Group 4 header-code is an escape sequence, the pattern-codes following it are actually carrying the original 8-bit pattern. Thus, there is no need to send Look Up Table (LUT) contents for Group 4 patterns for decoding. On the other hand, Group 1, 2 and 3 patterns must be transmitted as overhead to the decoding end to form a decoding LUT. There are a total of 52 8-bit, byte, patterns in the LUT. However, from our experiments, the Group 1 patterns are always uni-level patterns, i.e., $\begin{smallmatrix} 0 & 0 \\ 0 & 0 \end{smallmatrix}$, $\begin{smallmatrix} 1 & 1 \\ 1 & 1 \end{smallmatrix}$, $\begin{smallmatrix} 2 & 2 \\ 2 & 2 \end{smallmatrix}$ and $\begin{smallmatrix} 3 & 3 \\ 3 & 3 \end{smallmatrix}$, since their probabilities are several times higher than the others. Although their order in Group 1 might change, compare Table 6.7 and Table 6.8, it is irrelevant to select the order as long as these uni-level patterns are assigned in Group 1. Under this condition, a pre-set order in the LUT can be used without affecting coding/decoding and coded data size. Thus, an extra 4 out

No.	pattern	prob.	total prob.	No.	pattern	prob.	total prob.
The first 4 high prob. patterns: G 1				25	$\begin{smallmatrix} 2 & 2 \\ 1 & 2 \end{smallmatrix}$	1.157 %	7.209 %
0	$\begin{smallmatrix} 0 & 0 \\ 0 & 0 \end{smallmatrix}$	12.77 %	12.77 %	26	$\begin{smallmatrix} 1 & 1 \\ 1 & 0 \end{smallmatrix}$	1.139 %	8.348 %
1	$\begin{smallmatrix} 1 & 1 \\ 1 & 1 \end{smallmatrix}$	6.932 %	19.70 %	27	$\begin{smallmatrix} 0 & 1 \\ 1 & 1 \end{smallmatrix}$	1.128 %	9.476 %
2	$\begin{smallmatrix} 2 & 2 \\ 2 & 2 \end{smallmatrix}$	6.745 %	26.45 %	28	$\begin{smallmatrix} 2 & 2 \\ 2 & 1 \end{smallmatrix}$	1.109 %	10.58 %
3	$\begin{smallmatrix} 3 & 3 \\ 3 & 3 \end{smallmatrix}$	5.889 %	32.34 %	29	$\begin{smallmatrix} 1 & 2 \\ 2 & 2 \end{smallmatrix}$	1.107 %	11.69 %
The next 16 patterns: G 2				30	$\begin{smallmatrix} 3 & 2 \\ 3 & 2 \end{smallmatrix}$	1.103 %	12.79 %
4	$\begin{smallmatrix} 0 & 0 \\ 1 & 1 \end{smallmatrix}$	1.803 %	1.803 %	31	$\begin{smallmatrix} 3 & 2 \\ 3 & 3 \end{smallmatrix}$	1.062 %	13.86 %
5	$\begin{smallmatrix} 1 & 1 \\ 0 & 0 \end{smallmatrix}$	1.674 %	3.477 %	32	$\begin{smallmatrix} 2 & 2 \\ 3 & 2 \end{smallmatrix}$	1.013 %	14.87 %
6	$\begin{smallmatrix} 1 & 1 \\ 2 & 2 \end{smallmatrix}$	1.643 %	5.121 %	33	$\begin{smallmatrix} 3 & 3 \\ 2 & 3 \end{smallmatrix}$	0.991 %	15.86 %
7	$\begin{smallmatrix} 0 & 0 \\ 1 & 0 \end{smallmatrix}$	1.612 %	6.762 %	34	$\begin{smallmatrix} 2 & 3 \\ 2 & 3 \end{smallmatrix}$	0.954 %	16.81 %
8	$\begin{smallmatrix} 2 & 2 \\ 1 & 1 \end{smallmatrix}$	1.549 %	8.311 %	35	$\begin{smallmatrix} 2 & 3 \\ 2 & 2 \end{smallmatrix}$	0.908 %	17.72 %
9	$\begin{smallmatrix} 0 & 1 \\ 0 & 0 \end{smallmatrix}$	1.530 %	9.841 %	36	$\begin{smallmatrix} 3 & 2 \\ 2 & 2 \end{smallmatrix}$	0.868 %	18.59 %
10	$\begin{smallmatrix} 0 & 1 \\ 0 & 1 \end{smallmatrix}$	1.483 %	11.32 %	37	$\begin{smallmatrix} 3 & 2 \\ 2 & 3 \end{smallmatrix}$	0.846 %	19.43 %
11	$\begin{smallmatrix} 1 & 0 \\ 1 & 0 \end{smallmatrix}$	1.475 %	12.80 %	38	$\begin{smallmatrix} 2 & 3 \\ 3 & 3 \end{smallmatrix}$	0.781 %	20.22 %
12	$\begin{smallmatrix} 2 & 1 \\ 2 & 1 \end{smallmatrix}$	1.466 %	14.27 %	39	$\begin{smallmatrix} 3 & 3 \\ 3 & 2 \end{smallmatrix}$	0.779 %	20.99 %
13	$\begin{smallmatrix} 1 & 1 \\ 2 & 1 \end{smallmatrix}$	1.429 %	15.69 %	40	$\begin{smallmatrix} 0 & 0 \\ 2 & 2 \end{smallmatrix}$	0.586 %	21.58 %
14	$\begin{smallmatrix} 1 & 0 \\ 0 & 0 \end{smallmatrix}$	1.429 %	17.12 %	41	$\begin{smallmatrix} 1 & 0 \\ 2 & 1 \end{smallmatrix}$	0.512 %	22.09 %
15	$\begin{smallmatrix} 0 & 0 \\ 0 & 1 \end{smallmatrix}$	1.420 %	18.54 %	42	$\begin{smallmatrix} 2 & 1 \\ 3 & 2 \end{smallmatrix}$	0.475 %	22.57 %
16	$\begin{smallmatrix} 1 & 2 \\ 1 & 1 \end{smallmatrix}$	1.323 %	19.87 %	43	$\begin{smallmatrix} 1 & 2 \\ 0 & 1 \end{smallmatrix}$	0.456 %	23.02 %
17	$\begin{smallmatrix} 2 & 2 \\ 3 & 3 \end{smallmatrix}$	1.310 %	21.18 %	44	$\begin{smallmatrix} 2 & 2 \\ 0 & 0 \end{smallmatrix}$	0.447 %	23.47 %
18	$\begin{smallmatrix} 1 & 2 \\ 1 & 2 \end{smallmatrix}$	1.303 %	22.48 %	45	$\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}$	0.447 %	23.92 %
19	$\begin{smallmatrix} 2 & 1 \\ 2 & 2 \end{smallmatrix}$	1.288 %	23.77 %	46	$\begin{smallmatrix} 2 & 3 \\ 1 & 2 \end{smallmatrix}$	0.407 %	24.32 %
The next 32 patterns: G 3				47	$\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}$	0.385 %	24.71 %
20	$\begin{smallmatrix} 1 & 0 \\ 1 & 1 \end{smallmatrix}$	1.264 %	1.264 %	48	$\begin{smallmatrix} 2 & 1 \\ 1 & 0 \end{smallmatrix}$	0.383 %	25.09 %
21	$\begin{smallmatrix} 3 & 3 \\ 2 & 2 \end{smallmatrix}$	1.234 %	2.498 %	49	$\begin{smallmatrix} 1 & 2 \\ 2 & 1 \end{smallmatrix}$	0.379 %	25.47 %
22	$\begin{smallmatrix} 2 & 1 \\ 1 & 1 \end{smallmatrix}$	1.201 %	3.699 %	50	$\begin{smallmatrix} 0 & 1 \\ 1 & 2 \end{smallmatrix}$	0.375 %	25.84 %
23	$\begin{smallmatrix} 1 & 1 \\ 1 & 2 \end{smallmatrix}$	1.190 %	4.889 %	51	$\begin{smallmatrix} 2 & 1 \\ 1 & 2 \end{smallmatrix}$	0.367 %	26.21 %
24	$\begin{smallmatrix} 1 & 1 \\ 0 & 1 \end{smallmatrix}$	1.163 %	6.052 %	The rest of patterns: G 4			17.68 %

Table 6.7: High probability patterns (CSI).

No.	pattern	prob.	total prob.	No.	pattern	prob.	total prob.
The first 4 high prob. patterns: G 1				25	$\begin{smallmatrix} 2 & 2 \\ 2 & 1 \end{smallmatrix}$	1.174 %	7.378 %
0	$\begin{smallmatrix} 3 & 3 \\ 3 & 3 \end{smallmatrix}$	9.384 %	9.384 %	26	$\begin{smallmatrix} 2 & 1 \\ 2 & 2 \end{smallmatrix}$	1.165 %	8.542 %
1	$\begin{smallmatrix} 0 & 0 \\ 0 & 0 \end{smallmatrix}$	6.331 %	15.72 %	27	$\begin{smallmatrix} 3 & 2 \\ 3 & 3 \end{smallmatrix}$	1.152 %	9.694 %
2	$\begin{smallmatrix} 2 & 2 \\ 2 & 2 \end{smallmatrix}$	4.231 %	19.95 %	28	$\begin{smallmatrix} 0 & 1 \\ 1 & 1 \end{smallmatrix}$	1.152 %	10.84 %
3	$\begin{smallmatrix} 1 & 1 \\ 1 & 1 \end{smallmatrix}$	3.696 %	23.64 %	29	$\begin{smallmatrix} 2 & 2 \\ 1 & 1 \end{smallmatrix}$	1.117 %	11.96 %
The next 16 patterns: G 2				30	$\begin{smallmatrix} 1 & 1 \\ 0 & 1 \end{smallmatrix}$	1.113 %	13.07 %
4	$\begin{smallmatrix} 1 & 2 \\ 1 & 2 \end{smallmatrix}$	2.095 %	2.095 %	31	$\begin{smallmatrix} 0 & 1 \\ 0 & 0 \end{smallmatrix}$	1.074 %	14.15 %
5	$\begin{smallmatrix} 2 & 3 \\ 2 & 3 \end{smallmatrix}$	2.047 %	4.142 %	32	$\begin{smallmatrix} 1 & 1 \\ 2 & 2 \end{smallmatrix}$	1.047 %	15.19 %
6	$\begin{smallmatrix} 0 & 1 \\ 0 & 1 \end{smallmatrix}$	1.867 %	6.009 %	33	$\begin{smallmatrix} 1 & 1 \\ 1 & 0 \end{smallmatrix}$	1.043 %	16.24 %
7	$\begin{smallmatrix} 2 & 1 \\ 2 & 1 \end{smallmatrix}$	1.865 %	7.873 %	34	$\begin{smallmatrix} 1 & 0 \\ 1 & 1 \end{smallmatrix}$	1.034 %	17.27 %
8	$\begin{smallmatrix} 1 & 0 \\ 1 & 0 \end{smallmatrix}$	1.776 %	9.650 %	35	$\begin{smallmatrix} 3 & 3 \\ 2 & 2 \end{smallmatrix}$	1.036 %	18.27 %
9	$\begin{smallmatrix} 3 & 2 \\ 3 & 2 \end{smallmatrix}$	1.680 %	11.34 %	36	$\begin{smallmatrix} 0 & 0 \\ 1 & 0 \end{smallmatrix}$	0.951 %	19.22 %
10	$\begin{smallmatrix} 2 & 3 \\ 3 & 3 \end{smallmatrix}$	1.645 %	12.98 %	37	$\begin{smallmatrix} 2 & 2 \\ 3 & 3 \end{smallmatrix}$	0.916 %	20.14 %
11	$\begin{smallmatrix} 3 & 3 \\ 3 & 2 \end{smallmatrix}$	1.498 %	14.48 %	38	$\begin{smallmatrix} 0 & 0 \\ 1 & 1 \end{smallmatrix}$	0.807 %	20.94 %
12	$\begin{smallmatrix} 1 & 1 \\ 1 & 2 \end{smallmatrix}$	1.463 %	15.95 %	39	$\begin{smallmatrix} 1 & 1 \\ 0 & 0 \end{smallmatrix}$	0.772 %	21.72 %
13	$\begin{smallmatrix} 1 & 2 \\ 2 & 2 \end{smallmatrix}$	1.459 %	17.40 %	40	$\begin{smallmatrix} 2 & 1 \\ 1 & 2 \end{smallmatrix}$	0.715 %	22.73 %
14	$\begin{smallmatrix} 2 & 1 \\ 1 & 1 \end{smallmatrix}$	1.439 %	18.84 %	41	$\begin{smallmatrix} 1 & 2 \\ 2 & 3 \end{smallmatrix}$	0.711 %	23.14 %
15	$\begin{smallmatrix} 2 & 2 \\ 2 & 3 \end{smallmatrix}$	1.430 %	20.27 %	42	$\begin{smallmatrix} 1 & 2 \\ 2 & 1 \end{smallmatrix}$	0.663 %	23.81 %
16	$\begin{smallmatrix} 2 & 2 \\ 1 & 2 \end{smallmatrix}$	1.410 %	21.68 %	43	$\begin{smallmatrix} 0 & 1 \\ 1 & 2 \end{smallmatrix}$	0.632 %	24.44 %
17	$\begin{smallmatrix} 1 & 2 \\ 1 & 1 \end{smallmatrix}$	1.371 %	23.05 %	44	$\begin{smallmatrix} 3 & 2 \\ 2 & 1 \end{smallmatrix}$	0.599 %	25.04 %
18	$\begin{smallmatrix} 1 & 1 \\ 2 & 1 \end{smallmatrix}$	1.341 %	24.40 %	45	$\begin{smallmatrix} 2 & 3 \\ 3 & 2 \end{smallmatrix}$	0.580 %	25.62 %
19	$\begin{smallmatrix} 2 & 3 \\ 2 & 2 \end{smallmatrix}$	1.330 %	25.73 %	46	$\begin{smallmatrix} 2 & 1 \\ 1 & 0 \end{smallmatrix}$	0.579 %	26.20 %
The next 32 patterns: G 3				47	$\begin{smallmatrix} 3 & 2 \\ 2 & 3 \end{smallmatrix}$	0.531 %	26.73 %
20	$\begin{smallmatrix} 3 & 2 \\ 2 & 2 \end{smallmatrix}$	1.284 %	1.284 %	48	$\begin{smallmatrix} 1 & 2 \\ 0 & 1 \end{smallmatrix}$	0.474 %	27.20 %
21	$\begin{smallmatrix} 0 & 0 \\ 0 & 1 \end{smallmatrix}$	1.273 %	2.558 %	49	$\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}$	0.470 %	27.67 %
22	$\begin{smallmatrix} 3 & 3 \\ 2 & 3 \end{smallmatrix}$	1.246 %	3.803 %	50	$\begin{smallmatrix} 1 & 0 \\ 2 & 1 \end{smallmatrix}$	0.468 %	28.14 %
23	$\begin{smallmatrix} 1 & 0 \\ 0 & 0 \end{smallmatrix}$	1.216 %	5.019 %	51	$\begin{smallmatrix} 2 & 3 \\ 1 & 2 \end{smallmatrix}$	0.435 %	28.57 %
24	$\begin{smallmatrix} 2 & 2 \\ 3 & 2 \end{smallmatrix}$	1.185 %	6.204 %	The rest of patterns: G 4		22.05 %	

Table 6.8: High probability patterns (Girl).

Figure 6.18: Variable block entropy code for 4-element pattern encoding.

of 52 bytes, 4 %, can be saved in the overhead. Now, the overhead has 48 bytes, or 384 bits. For a 512×512 image, overhead takes only 0.0015 bits/pixel. Since those uni-level patterns will never appear in the overhead data, they can be assigned as Control Sequences, to be used later.

The variable length block code designed here is not entropy optimized, but it has several advantages over Huffman code:

- The size of the overhead, or codebook, is much smaller than it would be for a Huffman code, so that an image can be partitioned into smaller portions, and coded separately using several processing elements to increase the throughput.
- It has less decoding delay. For a Huffman code, the codewords have to be read in bit-by-bit to follow the coding tree, which makes the decoding process very slow if implemented on a general purpose computer.

- The coded data are less sensitive to channel errors. As long as the error does not hit the header-codes, it can be recovered automatically, without destroying the rest of the image.

6.4.3 Compression Overhead Using Pattern Symmetry Property

The symmetry of the image correlation function can also be used to reduce the overhead. For an image with *isotropic* pixel correlation, its rotationally symmetric patterns, e.g., $\begin{smallmatrix} 0 & 0 \\ 1 & 1 \end{smallmatrix}$, $\begin{smallmatrix} 1 & 1 \\ 0 & 0 \end{smallmatrix}$, $\begin{smallmatrix} 0 & 1 \\ 0 & 1 \end{smallmatrix}$ and $\begin{smallmatrix} 1 & 0 \\ 1 & 0 \end{smallmatrix}$, are expected to have similar probabilities which means that they very likely belong to the same Group. For example, in Table 6.7 Group 2 patterns, 12 out of the 16 patterns can be formed from 3 rotationally symmetric patterns, i.e., $\begin{smallmatrix} 0 & 0 \\ 1 & 1 \end{smallmatrix}$, $\begin{smallmatrix} 1 & 1 \\ 2 & 2 \end{smallmatrix}$ and $\begin{smallmatrix} 0 & 0 \\ 0 & 1 \end{smallmatrix}$, simply by rotating each of them 90°, 180° and 270° clockwise. As mentioned before, once a set of patterns belongs to the same Group, their order is not important, since they will be assigned to the same length of code, and changing their order within the Group would not affect the size of encoded data. For the rotationally symmetric patterns, we can simply send one out of four of them preceded by an escape code to indicate that the following patterns are rotated. In this case, up to 3:1 compression of overhead can be achieved.

As for *separable* correlation, patterns symmetric about vertical, horizontal and diagonal axes are expected to have similar probabilities. (Note: patterns $\begin{smallmatrix} 1 & 2 \\ 1 & 2 \end{smallmatrix}$ and $\begin{smallmatrix} 2 & 1 \\ 2 & 1 \end{smallmatrix}$ are symmetric about the vertical axis; patterns $\begin{smallmatrix} 1 & 2 \\ 2 & 2 \end{smallmatrix}$ and $\begin{smallmatrix} 2 & 2 \\ 1 & 2 \end{smallmatrix}$ are horizontally symmetric; and pattern $\begin{smallmatrix} 2 & 1 \\ 1 & 1 \end{smallmatrix}$ and $\begin{smallmatrix} 1 & 1 \\ 1 & 2 \end{smallmatrix}$ are diagonal symmetric.) These properties can also be used to compress overhead, with up to 2:1 compression.

Table 6.9 column two and three (column six and seven) show the LUTs of CSI (the Girl image) before and after compression. It can be seen that about 2:1 compression is achieved. (In Table 6.9, R, V, H, D, and N stands for Rotational, Vertical, Horizontal,

CSI				Girl's Image			
No.	patterns	code	code type	No.	patterns	code	code type
1	00R	000	CtrlR	1	12V	000	CtrlR
2	11R	000	R	2	13V	000	R
3	01R	111	R	3	23V	111	CtrlV
4	20R	222	R	4	01V	111	V
5	12R	333	CtrlD	5	21V	111	V
6	00R	333	D	6	10V	001	V
7	00R	222	N	7	33D	001	CtrlH
8	11R	222	N	8	33D	111	H
9	22R	000	CtrlR	9	12R	333	CtrlD
10	22R	111	R	10	12H	333	D
11	00R	333	R	11	22R	333	D
12	00R	333	R	12	22D	000	CtrlR
13	11D	333	R	13	22H	000	R
14	22N	111	CtrlV	14	12R	001	R
15	22R	111	V	15	11R	001	R
16	22N	333	V	16	22D	111	CtrlH
17	22R	333	CtrlH	17	22H	333	H
18	33N	000	H	18	00R	222	H
19	22D	333	CtrlD	19	33D	222	H
20	11D	333	D	20	10R	333	H
21	01R	333	D	21	22H	000	H
22	11V	000	D	22	33H	111	H
23	22R	333	D	23	22H	333	H
24	00R	333	N	24	33D	333	CtrlD
25	22V	333	N	25	01R	333	D
26	33N			26	22H	111	D
27	33R			27	11R	100	N
28	33H			28	00R	111	N
29	33R			29	11H		
30	33R			30	11R		
31	22V			31	10R		
32	22R			32	33H		
33	22R			33	00R		
34	22R			34	10H		
35	33R			35	00H		
36	33R			36	11H		
37	00H			37	11H		
38	22R			38	22D		
39	22D			39	22H		
40	01R			40	01R		
41	00H			41	33D		
42	00D			42	22H		
43	22D			43	22R		
44	00D			44	33H		
45	01R			45	00R		
46	22D			46	00N		
47	01R			47	22R		
48	22D			48	22N		

Table 6.9: Overhead code before and after compression.

Diagonal symmetry and Non-coded patterns. Patterns $\begin{smallmatrix} 0 & 0 \\ 0 & 0 \end{smallmatrix}$, $\begin{smallmatrix} 1 & 1 \\ 1 & 1 \end{smallmatrix}$, $\begin{smallmatrix} 2 & 2 \\ 2 & 2 \end{smallmatrix}$ and $\begin{smallmatrix} 3 & 3 \\ 3 & 3 \end{smallmatrix}$ are used as the Control Sequences Ctrl_R, Ctrl_V, Ctrl_H and Ctrl_D, respectively.)

As mentioned before, for 512×512 images, overhead is only 0.0015 bits/element without overhead compression. With this small amount of overhead, the compression seems trivial. However, if the image size is small, overhead compression will be important, e.g., for a 64×64 subimage, overhead is about 0.2 bits/element without compression, or 0.1 bits/element with compression.

On the other hand, overhead bits are vulnerable to channel errors. Any error hit in the overhead might result in total failure of reconstruction of the transmitted image. Compression of the overhead bits will reduce the risk of error hit, and, thus, increase the error robustness.

6.4.4 Block Adaptation and Parallelism

If we partition the 4-dimensional vector file into small blocks, and implement the variable length block code on each sub-block, the average entropy, or bit rate, might be less than that achieved by encoding the whole file without partitioning, because on each sub-block the sample space might be less than the whole image. Nevertheless, overhead for each sub-block must be transmitted for decoding. This will increase the bit rate. The entropy change and overhead bits increase are strongly dependent on the sub-block size. Larger block size results in less average overhead bit rate, but would not reduce entropy by much. Table 6.10 shows the result of using different block sizes, where the image size is 256×256 , and block sizes of 128×128 and 64×64 are used.

It can be seen, in Table 6.10, that small block size results in less average entropy and code bit rate. The increase of average overhead rate surpasses the decrease of

	CSI			The Girl Image		
(bits/element)	256 × 256	128 × 128	64 × 64	256 × 256	128 × 128	64 × 64
average entropy	1.5304	1.5090	1.4880	1.5602	1.5400	1.5199
code bit rate	1.5990	1.5885	1.5685	1.6317	1.6237	1.6195
overhead rate no comp.	0.0059	0.0234	0.0930	0.0059	0.0234	0.0936
overhead rate with comp.	0.0032	0.0151	0.0651	0.0038	0.0151	0.0658
total bit rate no comp.	1.6049	1.6119	1.6615	1.6376	1.6471	1.7059
total bit rate with comp.	1.6022	1.6036	1.6336	1.6355	1.6388	1.6817

Table 6.10: Block entropy and bit rates.

code rate, and the total bit rate actually increases slightly. On this basis it does not seem worthwhile to code data by blocks, but doing so gives an option of totally parallel processing. Several processing elements can be used to do the encoding separately without any “waiting” or “inter-communication”, which can achieve a large throughput and reduce the coding delay.

Comparing entropy and bit rate, although our code is not entropy optimized, the difference between entropy and bit rate is only about 0.07 bits.

6.4.5 Quaternary Plane Encoding Diagram.

Fig. 6.19 is a diagram of the quaternary plane encoder. It has following steps:

Step 1: Decompose the quaternary plane, 2 bits/element, into 2×2 patterns, and pack each pattern, a 4-dimensional vector, into a byte to form a 4-dimensional vector file.

Step 2: Calculate the histogram of 4-dimensional vector file.

Step 3: Sort histogram components in decreasing order.

Step 4: Search the sorted histogram for pattern symmetry properties.

Step 5: Based on sorted histogram and pattern symmetry properties, generate an encoding LUT which also, as the overhead, is to be transmitted to form the decoding LUT.

Step 6: Encode the vector file using the encoding LUT.

In Step 3, sorting 256 histogram components in decreasing order requires

$$\sum_{n=1}^{255} (256 - n) = 255 \cdot \frac{1 + 255}{2} = 32640 \quad (6.88)$$

searching loops. However, sorting the first 52 components is all that is really needed in our encoding algorithm, which requires

$$\sum_{n=1}^{52} (256 - n) = 52 \cdot \frac{255 + 204}{2} = 11934 \quad (6.89)$$

searching loops. This pruning scheme can achieve about 3:1 saving of searching computation.

6.4.6 The Possibility of Using Larger Pattern Size

So far, we have used a 2×2 size pattern on the quaternary plane. If a larger size, e.g., 3×3 , is used, there are 18 bits in the 9-dimensional vector and 262,144 possible patterns. If we sort histogram components in decreasing order, step in Fig. 6.19, there are

$$\sum_{n=1}^{262143} (262144 - n) = 262143 \cdot \frac{1 + 262143}{2} = 3.44 \times 10^{10} \quad (6.90)$$

searching loops! Even with the pruning, e.g., sorting the first 20 % of the components, Eq-6.89, there are

$$\sum_{n=1}^{262144/5} (262144 - n) \approx 52429 \cdot \frac{262143 + 209715}{2} = 1.24 \times 10^{10} \quad (6.91)$$

Figure 6.19: Quaternary plane encoding diagram.

searching loops, which proves to be several hours computation on a 10 MIPS machine. Meanwhile, the overhead will contain $262,144 \times 20\% = 52429$ 18-bit patterns, or 117,965 bytes, which is unrealistic.

On the other hand, as proved before, most of the redundancy in the quaternary plane is among very close neighbours. Increasing the pattern size might not obtain much coding gain. This can be proved by entropies. For 2×2 patterns, the average entropy is 1.52 bit/element, while, for 3×3 patterns, it is 1.46 bit/element. There is only 0.06 bit/element entropy gain.

Thus, for our quaternary plane coding algorithm, increasing the pattern size is pointless. It will increase the computational complexity, overhead data, and would not obtain much coding gain, because of the data correlation property.

6.4.7 Transmission Error Property

For our coding algorithm, as long as transmission errors do not hit the 2-bit header-codes, the errors will not propagate and the decoding process will not lose synchronization. Since the average bit rate is about 6 bits/pattern, the header bits occurrence probability $P(\text{header}) = 1/3$ of the total bits. Assuming the transmission Bit Error Rate (BER) is p , the probability P_e that the 2-bit header is hit by error is

$$P_e = \sum_{n=1}^2 C_2^n \cdot p^n = 1 - (1 - p)^2. \quad (6.92)$$

Omitting the high order terms,

$$P_e \approx 2p. \quad (6.93)$$

The unrecoverable error rate, which is the header bits hit by error, is

$$P_{e(\text{unrecoverable})} = 2p \cdot P(\text{header}) = \frac{2}{3}p = 0.667p, \quad (6.94)$$

which is lower than the corresponding probability, p essentially, for the Huffman code.

This section described a lossless coding algorithm for the quaternary plane. Based on the correlation property of the quaternary plane, where most of the redundancy in the plane is among very close neighbours, a 4-element, 4-dimensional, packing scheme is used to map the quaternary plane into a byte-oriented data set which is coded using a variable length block code. About 20 % to 30 % saving of code bits can be achieved. The pattern symmetries are used to compress the overhead (about 2:1 compression).

6.5 Rate Distortion Curves of Minimax Error Multi-Level Block Truncation Coding with Lossless Post-Quantization Compression

In previous sections, minimax error quaternary block truncation coding was introduced. One of the disadvantages of the original bi-level block truncation coding [45-46] is that it has a limited range of rate distortion-bit rate tradeoff by employing different block sizes [119]. Since we have extended the bi-level BTC into multi-level BTC, various quantization levels and block sizes can be implemented which will result in different bit rates and distortions. The rate distortion curves are shown in Fig. 6.20 and the MSE values are listed in Table 6.11. The results of implementing lossless post-quantization data compression, Section 6.3 and 6.4, are also displayed.

In Table 6.11, (L, N) stands for L-level, N by N block multi-level BTC. The original and reconstructed pictures are shown in Picture I & II and Picture VI & VII.

Figure 6.20: Rate distortion curves of BTC with various quantization levels and block sizes.

	no post-comp	with post-comp	Girl	CSI
(L,N)	bits/pixel	bits/pixel	NMSE	NMSE
(4,4)	4.0	2.9	0.014 %	0.170 %
(3,4)	3.08	2.4	0.029 %	0.358 %
(4,8)	2.5	1.8	0.042 %	0.465 %
(3,6)	2.25	1.75	0.054 %	0.615 %
(3,8)	1.955	1.5	0.077 %	0.868 %
(4,16)	2.125	1.5	0.162 %	0.909 %

Table 6.11: MSE performances of BTC with various quantization levels and block sizes.

Picture VI: Reconstructed image (4, 8).

Picture VII: Reconstructed image (4, 8).

6.6 Quaternary Plane Data Compression Using Vector Quantization

In the previous sections, we have shown that there are correlations, or redundancies, on the quaternary plane and an algorithm of lossless compression of the quaternary plane has been developed. This section will discuss the lossy compression of the quaternary planes.

Vector quantization (VQ) is a recently developed technique for data compression [76,77]. VQ provides high compression ratio; fast, Look-Up-Table (LUT)-based decoding and competitive subjective performance; but the encoding procedure is quite time consuming and sharp edges tend to be filtered out in the reconstructed images. Since the quaternary plane elements have very small dynamic range (2 bits/element), it is possible to implement a simple LUT-based VQ encoding procedure which speeds up the encoding process substantially. The sharp edges and gray level sudden changes in the original image have already been preserved by the preceding quaternary BTC, so that they are not filtered out by the subsequent VQ coding.

Generally, VQ has the following major steps [77], see Chapter 5:

1. *Vector Formation.*
2. *Training sequence.*
3. *Codebook Generation.*
4. *Quantization.*
5. *Decoding.*

The typical VQ block size is 4×4 . For 8 bits/pixel digital imagery, there are 2^{128} representation vectors in the image sample space. It is impossible to build an encoding

Figure 6.21: Rate distortion curves of BTC-VQ-DPCM coding and VQ.

LUT to map each of these input vectors into a corresponding codeword. This implies that the encoder must perform an exhaustive search through the entire codebook to find the minimum distortion codeword for each input vector. The computation involved in this vector matching procedure is enormous.

In our special case, VQ is implemented on the quaternary plane which can be realized by a simple LUT operation, drastically reducing the VQ coding complexity.

The quaternary plane is decomposed into 4×2 blocks. Each block can be packed into a 16-bit word forming an 8-dimensional vector. The total number of representation vectors is only $2^{16} = 64K$. It is reasonable to design a 64-Kword encoding LUT to quickly encode the input vector. Once the LUT is built, it takes almost no time to quantize a word-packed vector.

QBTC-VQ-DPCM Coding			Vector Quantization	
codebook size	NMSE (%)	bit rate	NMSE (%)	bit rate
QBTC(4,8)	0.4650	1.9	*	*
1024	1.6510	1.55	1.4184	2.625
512	1.8817	1.425	1.7899	1.5625
256	2.3259	1.30	2.2017	1.00
128	2.8364	1.175	2.5842	0.6875
64	3.4997	1.05	3.7612	0.50

Table 6.12: MSE performances of BTC-VQ-DPCM coding and VQ.

The MSE performances, for 256×256 CSI, are shown in Fig. 6.21 and Table 6.12 where, for comparison, the conventional VQ (VQA) performances are also listed. The codebook used in quaternary plane VQ is generated from a group of training images that includes standard images, satellite images, and radar images. A 64-Kword encoding-LUT for VQ of the quaternary plane can be built from the codebook through an exhaustive-search. Comparing Quaternary BTC-VQ-DPCM coding with conventional VQ, Table 6.12, our algorithm has comparable MSE performance at moderate bit rate. On the other hand, QBTC-VQ-DPCM coding has much less computational complexity, since VQ is implemented on the quaternary plane and a LUT-based encoding can be used.

6.7 Summary

Minimax Error Multi-Level BTC has strong edge preserving ability, high fidelity and low computational complexity. A fast-searching algorithm using a minimax quantization error criterion has been developed to allocate the quantizer levels to obtain a close replica of the original image. With the second stage compression of the BTC coded data using combined techniques of predictive coding, entropy coding and vector quantization, the compression ratio is about 6.3:1 with no perceived visual degradation. The algorithm is software oriented designed for running on general purpose computers. For a 512 by 512 image, running on a SUN-3 work station, the encoding part takes about 6 seconds and decoding about 3 seconds.

Chapter 7

SINGLE BIT-MAP BTC OF COLOUR IMAGES

This chapter describes a colour image data compression technique using Block Truncation Coding (BTC) and adaptive DCT called Single Bit-Map Block Truncation Coding (SBMBTC). The coding algorithm preserves the spatial details in the image content and has low computational complexity.

In this colour image coding technique, BTC is first implemented to encode a particular weighted trichromatic image. The bit-map generated is then used to compress each of the R, G and B planes separately. The weighted trichromatic plane is generated by properly weighting R, G and B planes so that the reconstructed R, G and B planes have minimum Mean Square Error (MSE). Another approach, using bit-wise majority rule on individual R, G and B BTC-generated bit-maps, is also discussed. The BTC coded data can be further compressed using adaptive DCT. Differential coding and error feedback techniques, as BTC-VQ-DCT coding in Chapter 5, are also implemented to reduce the bit rate and improve the MSE performance. The total compression ratio is about 12:1. By exploiting human visual system characteristics, a 14:1 compression of NTSC colour TV signals can be achieved.

7.1 Motivation for Single Bit-Map BTC

As discussed previously in Section 2.2 and 5.2, Block Truncation Coding (BTC) [45-49] uses a two-level moment-preserving quantizer that adapts to local properties to compress achromatic images. The BTC output data set includes a binary plane (bit-map) which defines the quantization level for each pixel and two 8-bit parameters; e.g. the block mean and standard deviation [45], or two reconstruction level codes [49]. The compression ratio for 4×4 blocks is 4:1, resulting in a transmission rate of 2 bits/pixel. If a 10-bit two-dimensional vector quantizer is used to coarsely quantize the mean and standard deviation [45], or the two reconstruction levels [48,91], a rate of 1.625 bits/pixel can be achieved, where the bit-map (1 bit/pixel) occupies more than 61 % of the output code.

Colour images, based upon the trichromatic nature of human vision [33,125,126], may be considered to consist of three planes — red, green and blue (R, G and B). There are very high correlations between the images in these planes [37]. Actually, in a colour image camera the R, G and B signals are generated by three identical image sensors with R, G or B optical filters placed in front of them. Thus, the three planes contain the same objects and, more importantly, by-and-large the same contours, although their intensity levels might be totally different. However, slight contour distortions or mis-registrations in the three colour planes might result from geometric distortions from the three sensors as well as the optical filters and some random noises from the different parts of the system.

If BTC is implemented to encode the R, G and B planes separately, the three resulting bit-maps will be highly correlated, since the BTC is a two-level quantizer that adapts to local properties of the image. This motivates the use of one bit-map to quantize all three of the colour planes. Thus, only one out of three bit-maps needs

to be preserved which means a saving of up to 40 % in the bit rate.

The problem with this approach is deciding which single bit-map should be used to quantize the three planes. As mentioned before, there are mis-registrations and random noises involved in the colour planes. Meanwhile, part of the colour components in one plane might not correlate with other planes. Thus, no one of the R, G or B bit-maps alone provides optimal performance. A fourth plane must be generated by properly weighting the corresponding R, G and B components such that it has minimal mis-registrations and random noises with respect to all three colour planes, and the uncorrelated components are best compromised. The BTC bit-map of the "Weighted plane" (W-plane) is, then, used to quantize the R, G and B planes and a better Mean Square Error (MSE) performance can be achieved.

In this chapter, Section 7.2 briefly reviews Block Truncation Coding. Section 7.3 discusses the generation of a W-plane theoretically based upon the least MSE guideline. The post-BTC data compression used to achieve a higher compression ratio is also presented. Section 7.4 describes the compression of NTSC colour TV signals. The chapter is summarized in Section 7.5.

7.2 Block Truncation Coding

In Lema and Mitchell's Absolute Moment BTC algorithm [48], the pixels in each block are sorted into high-mean and low-mean groups, preserving the Mean Absolute Moment or minimizing the Mean Absolute Error in each block. The mean intensity

$$\bar{X} = \frac{1}{l} \sum_{i=1}^l x(i) \quad (7.1)$$

is used as the threshold for the 1-bit quantizer, where $x(i)$ is the value of i th pixel in the block and l is the number of pixels in a block, i.e., $m \times m$. If

$$x(i) \leq \bar{X} \text{ or } x(i) > \bar{X}, \quad (7.2)$$

$\bar{X}_L$ or $\bar{X}_H$, respectively, is assigned to the quantizer output levels, where

$$\bar{X}_L = \frac{1}{l-q} \sum_{x(i) < \bar{X}} x(i), \quad (7.3)$$

$$\bar{X}_H = \frac{1}{q} \sum_{x(i) \geq \bar{X}} x(i), \quad (7.4)$$

and q is the number of pixels having value higher than $\bar{X}$. Hence, the outputs of the BTC for each block include two numbers which specify the high-mean, $\bar{X}_H$, and low-mean, $\bar{X}_L$; and a bit-map which specifies the states, $\bar{X}_H$ or $\bar{X}_L$, of each output pixel. It can be understood that

$$\bar{X}_L < \bar{X}_H. \quad (7.5)$$

This algorithm has the potential of low computational complexity with no multiplications involved and real-time implementation has been reported [50].

A two-dimensional vector quantizer was used to further compress the $\bar{X}_L$ and $\bar{X}_H$. A 10-bit vector quantizer having satisfactory visual results has been demonstrated [46,91]. For a 4×4 block size, the resulting bit rate is

$$1 + \frac{10}{16} = 1.625 \text{ bits/pixel} \quad (7.6)$$

Halverson et al. [49] generalized BTC to include a family of moment preserving quantizers with the potential of improving performance. It has been proved that the selection of the quantizer threshold and levels can be generalized for much larger

classes of pairs and triples of sample moments, respectively. Performance improvements were demonstrated from the standpoint of peak signal-to-noise ratio. However, the application of the BTC algorithm for the higher moments case is often hampered by overflow problems and increased computational complexity.

In this chapter, Lema's algorithm is used, because of its computational simplicity (Note: no matter which algorithm is used, the resulting bit-maps are identical, since they all use the same quantization threshold $\bar{X}$).

7.3 BTC of Colour Images

Table 7.1 shows the MSE values resulting from implementing BTC using various bit-maps individually on each of the R, G and B planes of Picture VIII.

7.3.1 BTC OF Colour Images Using Single Bit-Map

As mentioned in Section 7.1, because of the similarities among the bit-maps, the BTC bit-map of one of the R, G or B planes can be used to quantize the other two colour planes. Let $QBM(i)$ represent the quantization bit-map and q the number of 1's on the bit-map in one $m \times m$ block. Then Eq-7.3 and 7.4 are modified as

$$\bar{X}_1 = \frac{1}{l-q} \sum_{QBM(i)=0} x(i), \quad (7.7)$$

$$\bar{X}_2 = \frac{1}{q} \sum_{QBM(i)=1} x(i), \quad (7.8)$$

Here we use $\bar{X}_1$ and $\bar{X}_2$ instead of $\bar{X}_L$ and $\bar{X}_H$, since there is possibility that $\bar{X}_1 > \bar{X}_2$. For instance, although the three bit-maps have the same contours, the 0's and 1's in some blocks of certain maps might be inverted. Originally, 0's represent low values in a block and 1's represent high, Eq-7.3 and 7.4, when direct BTC is implented. However, if there is no mis-registration and noise involved, implementing a bit inverted map in

Eq-7.7 and Eq-7.8 will not result in any mistake in the decoded plane since, in this case, $\bar{X}_1 > \bar{X}_2$.

The MSE results of using the bit-map derived from the R, G or B plane as the quantization bit-map are displayed in Table 7.1. It can be seen that the average MSE of reconstructed R, G and B planes are up to 50 % higher than that resulting from direct BTC of the colour planes. Using different bit-maps (R, G or B) as the quantization bit-map also results in different average MSE.

Table 7.1 also shows the correlations between the quantization bit-map and the direct BTC bit-maps [122], where

$$\text{Correlation } \rho = \frac{\text{number of identical bits} - \text{number of inverted bits}}{\text{total bits}}$$

and Inverted Blocks, where $\bar{X}_1 > \bar{X}_2$. These two parameters partially indicate the similarity between bit-maps. However the number of inverted bits and blocks can never be zero because of the existence of mis-registrations, random noises and uncorrelated colour components. But their effects can be reduced, as shown in the next section.

7.3.2 Minimum MSE Approach

Each colour plane is assumed to consist of two parts,

$$\begin{aligned} R &= R' + n_R \\ G &= G' + n_G \\ B &= B' + n_B \end{aligned} \quad (7.9)$$

where n_R , n_G and n_B represent the equivalent "noises" caused by mis-registrations, random noises and uncorrelated colour components, which are assumed to be independent identically distributed (i.i.d.) random variables [113]; and R' , G' and B' are colour components that have very high inter colour-plane correlations such that,

Quantization Map		R plane	G plane	B plane	Y plane	W plane	Ave. of RGB
direct BTC	MSE	26.27	42.71	29.78	32.42	27.54	32.92
R-map	MSE	26.27	63.46	59.93	*	*	48.89
	Bit-Map Corr. ρ	1	0.39	0.20	*	*	
	Inverted Blocks	0	12.03 %	23.96 %	*	*	
G-map	MSE	35.81	42.71	46.68	*	*	41.73
	Bit-Map Corr. ρ	0.39	1	0.49	*	*	
	Inverted Blocks	10.67 %	0	2.554 %	*	*	
B-map	MSE	49.24	62.84	29.78	*	*	47.29
	Bit-Map Corr. ρ	0.20	0.49	1	*	*	
	Inverted Blocks	26.41 %	2.351 %	0	*	"	
W-map	MSE	33.05	46.41	39.75	*	27.54	39.74
	Bit-Map Corr. ρ	0.47	0.74	0.66	*	1	
	Inverted Blocks	5.280 %	0.052 %	0.195 %	*	0	
M-map	MSE	34.19	45.73	43.34	*	*	41.09
	Bit-Map Corr. ρ	0.55	0.84	0.65	*	*	
	Inverted Blocks	4.207 %	0.053 %	0.624 %	*	*	
Y-map	MSE	32.34	44.25	45.89	32.42	*	40.82
	Bit-Map Corr. ρ	0.50	0.85	0.52	1	*	
	Inverted Blocks	3.523 %	0.000 %	1.307 %	0	*	

Table 7.1: MSE Performance of BTC of R, G and B planes.

if BTC were used to code each of them, the resulting bit-maps would be identical; or, in other words, image components that can be exactly represented by a single quantization bit-map.

Our purpose is to generate a fourth plane, W , by properly weighting the R , G and B planes, i.e.,

$$\begin{aligned} W &= W' + \mathbf{n}_W \\ &= \alpha R + \beta G + \gamma B, \\ &= (\alpha R' + \beta G' + \gamma B') + (\alpha \mathbf{n}_R + \beta \mathbf{n}_G + \gamma \mathbf{n}_B), \end{aligned} \quad (7.10)$$

where α , β and γ are constants and

$$\alpha + \beta + \gamma = 1, \quad (7.11)$$

to reduce the effects of $\mathbf{n}_R$, $\mathbf{n}_G$ and $\mathbf{n}_B$, and to minimize the MSE in the reconstructed planes.

The differences between the W -plane and the colour planes are

$$\begin{bmatrix} W - R \\ W - G \\ W - B \end{bmatrix} = \begin{bmatrix} \alpha - 1 & \beta & \gamma \\ \alpha & \beta - 1 & \gamma \\ \alpha & \beta & \gamma - 1 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}. \quad (7.12)$$

Assuming the uncorrelated components in the difference planes are $\mathbf{n}_{W-R}$, $\mathbf{n}_{W-G}$ and $\mathbf{n}_{W-B}$, the standard deviations are

$$\begin{bmatrix} \sigma^2(\mathbf{n}_{W-R}) \\ \sigma^2(\mathbf{n}_{W-G}) \\ \sigma^2(\mathbf{n}_{W-B}) \end{bmatrix} = \begin{bmatrix} (\alpha - 1)^2 & \beta^2 & \gamma^2 \\ \alpha^2 & (\beta - 1)^2 & \gamma^2 \\ \alpha^2 & \beta^2 & (\gamma - 1)^2 \end{bmatrix} \begin{bmatrix} \sigma^2(\mathbf{n}_R) \\ \sigma^2(\mathbf{n}_G) \\ \sigma^2(\mathbf{n}_B) \end{bmatrix}. \quad (7.13)$$

The total "noise" energy is

$$\begin{aligned} \sigma_{total}^2 &= \sigma^2(\mathbf{n}_{W-R}) + \sigma^2(\mathbf{n}_{W-G}) + \sigma^2(\mathbf{n}_{W-B}) \\ &= (1 - 2\alpha + 3\alpha^2)\sigma^2(\mathbf{n}_R) + (1 - 2\beta + 3\beta^2)\sigma^2(\mathbf{n}_G) + (1 - 2\gamma + 3\gamma^2)\sigma^2(\mathbf{n}_B). \end{aligned} \quad (7.14)$$

To minimize σ_{total}^2 , taking partial derivatives respect to α , β and γ and setting the results to zero,

$$\frac{\partial \sigma_{total}^2}{\partial \alpha} = (-2 + 6\alpha)\sigma^2(\mathbf{n}_R) = 0, \quad (7.15)$$

$$\frac{\partial \sigma_{total}^2}{\partial \beta} = (-2 + 6\beta)\sigma^2(\mathbf{n}_G) = 0, \quad (7.16)$$

$$\frac{\partial \sigma_{total}^2}{\partial \gamma} = (-2 + 6\gamma)\sigma^2(\mathbf{n}_B) = 0. \quad (7.17)$$

Since

$$\sigma^2(\mathbf{n}_R) \neq 0, \quad \sigma^2(\mathbf{n}_G) \neq 0, \quad \sigma^2(\mathbf{n}_B) \neq 0, \quad (7.18)$$

from Eq-7.15, 7.16 and 7.17,

$$\alpha = \beta = \gamma = \frac{1}{3}, \quad (7.19)$$

which also satisfies Eq-7.11. Thus, the MSE optimized W-plane is

$$W = \frac{R + G + B}{3}. \quad (7.20)$$

Actually, Eq-7.20 is an averaging process, which calculates the average of the corresponding pixels from the R, G and B planes and uses them as W-plane pixels. This process can reduce the effects of uncorrelated components in W-plane [113] relative to R, G, and B planes. For example, if

$$\sigma^2(\mathbf{n}_R) = \sigma^2(\mathbf{n}_G) = \sigma^2(\mathbf{n}_B) = \sigma^2, \quad (7.21)$$

then

$$\sigma^2(\mathbf{n}_W) = \frac{1}{9}\sigma^2 + \frac{1}{9}\sigma^2 + \frac{1}{9}\sigma^2 = \frac{1}{3}\sigma^2. \quad (7.22)$$

Substituting Eq-7.19 and Eq-7.21 into Eq 7.13,

$$\min \{\sigma^2(\mathbf{n}_{W-R,G,B})\} = \frac{2}{3}\sigma^2, \quad (7.23)$$

while the uncorrelated components between the two colour planes, e.g. between R and B, is

$$\sigma^2(\mathbf{n}_{R-B}) = \sigma^2(\mathbf{n}_R - \mathbf{n}_B) = 2\sigma^2, \quad (7.24)$$

which is three times higher than Eq-7.22.

We have minimized the error caused by mis-registrations, random noises and uncorrelated colour components between the W-plane and the colour planes. Since BTC is a 1-bit quantization process, there should be less differences between the BTC bit-maps of the W-plane and the bit-maps of other colour planes than that of other bit-maps. This can be shown by the reduction in the number of Inverted Bits, Inverted Blocks and, more important, the MSE values in Table 7.1, when the W-plane bit-map is used as the quantization map.

The system diagram of MSE-optimized single bit-map BTC of colour images is shown in Fig. 7.1. The input colour planes are first weighted together, using Eq. 7.10, to obtain a W-plane. Then, BTC is implemented on the W-plane to generate a quantization bit-map. Based on the bit-map, using Eq. 7.7 and 7.8, the R, G and B planes undergo BTC. The resulting quantization data from each block, $\bar{X}_{R1}$, $\bar{X}_{R2}$, $\bar{X}_{G1}$, $\bar{X}_{G2}$, $\bar{X}_{B1}$, and $\bar{X}_{B2}$ go through post-quantization data compression.

For 24 bits/pixel (R,G,B) input data and a 4×4 block size (if no post-quantization data compression scheme is implemented) the overall bit rate is

$$1 + \frac{3 \times 2 \times 8}{16} = 4 \text{ bits/pixel}, \quad (7.25)$$

or a 6:1 compression. If a 10-bit two-dimensional vector quantizer [46,91] is used to code the BTC output data $\bar{X}_{R,G,B1}$ and $\bar{X}_{R,G,B2}$, the overall bit rate is

$$1 + \frac{3 \times 10}{16} = 2.875 \text{ bits/pixel}. \quad (7.26)$$

The compression ratio is 8.35:1.

7.3.3 Bit-Map Majority Rule Approach

Another approach to generation of the quantization bit-map is using majority rule on R, G and B bit-maps. BTC is first implemented on the R, G and B planes. The

Figure 7.1: System diagram of MSE optimized single bit-map BTC of colour images.

resulting bit-maps are $BM_R(i, j)$, $BM_G(i, j)$ and $BM_B(i, j)$, where (i, j) specifies the bit-map element address and the bit-map sample space is $\{0, 1\}$. Let $QBM(i, j)$ represent the quantization bit-map, then

$$\begin{aligned} QBM(i, j) &= \text{Majority} \{BM_R(i, j), BM_G(i, j), BM_B(i, j)\} \\ &= BM_R(i, j) \cdot BM_G(i, j) + BM_R(i, j) \cdot BM_B(i, j) + BM_G(i, j) \cdot BM_B(i, j). \end{aligned} \quad (7.27)$$

The MSE performance of using the majority rule bit-map, M-map, to encode the R, G and B planes is shown in Table 7.1. It can be seen that bit-map correlations between the M-map and the R, G and B-maps are higher than other cases. On the other hand, the average MSE value is higher than with the minimum MSE approach.

To generate the quantization bit-map, three BTC processes must be implemented to code the R, G and B planes in the majority rule approach, while only one BTC process must be conducted in the minimum MSE approach. The total computational complexity of the majority rule approach is about 50 % higher than that of the minimum MSE approach.

From the above discussion, the minimum MSE approach is a better choice.

7.3.4 Post Quantization Data Compression Using Adaptive DCT

After BTC of the R, G and B planes using the W-plane bit-map as the quantization bit-map, there are seven sets of data to be transmitted, Fig. 7.1, i.e., one quantization bit-map, and, for each block, six quantization levels — $\bar{X}_{R1}$, $\bar{X}_{R2}$, $\bar{X}_{G1}$, $\bar{X}_{G2}$, $\bar{X}_{B1}$ and $\bar{X}_{B2}$. Each of these quantization levels is a “mean” of certain pixels inside a quantization block, as defined by Eq-7.7 and 7.8.

If quantization levels from each BTC block are grouped together as raster files, we can form six sub-sampled images — R_1 , R_2 , G_1 , G_2 , B_1 and B_2 . For a 512×512 input

Figure 7.2: Adaptive DCT coding of subsampled colour plane (red plane).

image with 4×4 blocks, each subsampled image is 128×128 . The subsampled-images have details and features which must be preserved, since any distortion involved here will be distributed over all of the pixels in each reconstructed BTC block of respected colour planes.

As discussed previously in Section 5.2.3, Adaptive DCT coding [4,70,71] with residual error feedback is implemented to compress the subsampled-images. the system diagram is shown in Fig. 7.2 and its MSE performances are listed in Table 7.2. The reconstructed image is displayed in Picture IX.

Computation Complexity of Single Bit-Map BTC and Adaptive DCT vs. direct Adaptive DCT

For SBMBTC-DCT coding, the BTC part requires relatively few computations per pixel with respect to DCT, which is, with residual error feedback, [63,66,67]

$$\frac{36 \cdot \log_2 N}{N_b \times N_b} \text{ multiplications/additions per pixel,} \quad (7.28)$$

Plane	Bit-Rate (bits/pixel)	MSE
R	0.369	44.5
G	0.400	56.8
B	0.356	50.3
Total (including bit-map)	2.125	50.5

Table 7.2: MSE performances of SBMBTC-DCT coding

where N_b is the BTC block size, N is the DCT block size. If $N_b = 4$ and $N = 16$, then

$$\frac{36 \cdot \log_2 N}{N_b \times N_b} = 9 \text{ multiplications/additions per pixel,} \quad (7.29)$$

For direct adaptive DCT coding, the number of computations is

$$12 \cdot \log_2 N \text{ multiplications/additions per pixel;}$$

which for $N = 16$ equals 48 multiplications/additions per pixel.

SBMBTC-DCT, with residual error feedback, is about five times faster than direct DCT. Without residual error feedback, it is about eight times faster than direct DCT.

Channel Error Effects on Single Bit-Map BTC Data

SBMBTC-DCT data comprises a quantization bit-map and six subsampled images which are further compressed using DCT coding, Fig. 7.1.

If transmission errors hit the bit-map, the only effects in the reconstructed images are the conversion of $\bar{X}_{R,G,B1}$ into $\bar{X}_{R,G,B2}$, or vice versa, for error contaminated bits. The errors will not propagate and the error effects are limited just like error effects to PCM coded data.

Since DCT coding is error robust, any error in the DCT-coded subsampled image data will be distributed over all the transform block [13]. The visual effects will be limited.

From the above discussions, SBMBTC-DCT coding has very strong error robustness.

7.4 NTSC Colour TV Signal Compression

In the NTSC colour television system, the R, G and B signals are combined to form a luminance signal Y and two chrominance signals I and Q, which are related by [33]

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = L \begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.274 & -0.322 \\ 0.211 & -0.523 & 0.312 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}. \quad (7.30)$$

The inverse transform is

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = L^{-1} \begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 1.000 & 0.956 & 0.621 \\ 1.000 & -0.272 & -0.647 \\ 1.000 & -1.106 & 1.703 \end{bmatrix} \begin{bmatrix} Y \\ I \\ Q \end{bmatrix}. \quad (7.31)$$

Since the human visual system is very sensitive to the spatial variations of Y, less sensitive to I, and much less sensitive to Q [33], we can low-pass filter the I signal with a 3×3 averaging filter and the Q signal with a 5×5 averaging filter, and code the resulting signals using SMBTC-DCT coding.

From Eq-7.30, Y is a combination of R, G and B, i.e.,

$$Y = 0.299 R + 0.587 G + 0.114 B, \quad (7.32)$$

Plane	Bit-Rate bits/pixel	MSE
Y	0.393	40.8
I	0.169	19.2
Q	0.119	14.9
Total (including bit-map)	1.675	

Table 7.3: MSE performances of SbMBTC-DCT coding of NTSC colour signal.

comparing with Eq-7.10,

$$\alpha = 0.299, \quad \beta = 0.587, \quad \gamma = 0.114. \quad (7.33)$$

To maintain a relatively high fidelity coding of Y to satisfy the human visual system, Y is directly coded by BTC and its bit-map is used as the quantization bit-map to code the low-pass filtered I and Q signals.

The system diagram is shown in Fig. 7.3. The MSE performance is listed in Table 7.3. Comparing with Table 7.2, a coding gain of approximately 0.5 bits/pixel can be obtained, because of the coarse coding of I and Q signals. The compression ratio is about 14:1. Picture-X displays the reconstructed image.

The bit rates of other test images are listed in Table 7.4. It can be seen that the test image "Baboon", which has a large region containing high frequency details, has a compression ratio about 12:1. The test image "Ken", which is a head and

Figure 7.3: System diagram of SBMBTC-DCT coding of NTSC signals.

	Bit-Rate (bits/pixel)		
Plane	Baboon	Kodak Girl	Ken
Y	0.531	0.392	0.326
I	0.394	0.213	0.138
Q	0.171	0.154	0.122
Total (including bit-map)	2.033	1.759	1.586
Compression Ratio	11.8:1	13.6:1	15.1:1

Table 7.4: Bit rate of SBMBTC-DCT coding of NTSC colour signal.

shoulder teleconferencing image, has a compression ratio about 15:1. The test image “Kodak Girl”, which contains a head and shoulder pose plus some other objects, has a compression ratio about 13.6:1. The average compression ratio of SBMBTC-DCT coding is between 12:1 to 15:1.

7.5 Conclusion

This chapter describes a colour image data compression technique using Block Truncation Coding (BTC). A single quantization bit-map is used to BTC all three colour planes, which removes the inter-colour correlations. The quantized data are further compressed using adaptive DCT coding. Differential coding and residual error feedback techniques are implemented to reduce the bit rate and improve the MSE performance. The compression ratio is about 12:1. For NTSC colour TV signals, the compression can be increased to 14:1 by taking advantage of the reduced bandwidth of the chrominance signals.

Picture VIII: Original image.

3

of/de

3

Picture IX: Reconstructed image (12:1 compression).

Picture X: NTSC reconstructed image (14:1 compression).

Chapter 8

SUMMARY AND FUTURE RESEARCH WORK

8.1 A Research Summary

In this thesis, three hybrid coding techniques, *BTC-VQ-DCT Hybrid Coding*, *Minimax Error Multi-Level Block Truncation Coding* and *Single Bit-Map Block Truncation Coding*, which achieve compromise between computational complexity, compression ratio, and fidelity have been presented. Each coding scheme contains more than one data compression stage, which removes the redundancy more completely than that of conventional single stage coding, and different coding techniques were implemented in each step.

BTC-VQ-DCT Coding

BTC-VQ-DCT coding combines the advantages of simple computation and edge preservation of BTC, the high fidelity and high compression ratio of adaptive DCT, and the high compression ratio and good subjective performance of VQ. It has been shown that the overall computational complexity of BTC-VQ-DCT coding is much less than either DCT and VQ, while the fidelity performance is competitive. The compression ratio is about 10:1.

Miminax Error Multi-Level BTC

Miminax Error Multi-Level BTC has strong edge preserving ability, high fidelity and low computational complexity. It is a general extension of Block Truncation Coding (BTC), wherein a multi-level quantizer is implemented instead of the usual two-level quantizer to obtain higher fidelity. A fast-searching algorithm using a minimax quantization error criterion has been developed to allocate the quantizer levels to obtain a close replica of the original image. The output of the multi-level BTC coding consists of a multi-level plane, which specifies the quantization levels assigned to each pixel, and the quantization level codes for each block. Second stage compression of this coded data using combined techniques of predictive coding, entropy coding and vector quantization has also been described. Both inter- and intra-block prediction have been used to predict the quantization levels and the prediction errors are coded by a specially designed lossless vector-entropy coding. Vector quantization has been implemented to further remove the redundancy within the multi-level plane. This compression can be lossless or lossy. The compression ratio is about 6:1, if lossless second stage compression is implemented. The coding algorithm has demonstrated very high fidelity, with no perceived visual degradation, and very low computational complexity. The minimax error algorithm is most suitable for coding the satellite remote sensing and synthetic aperture radar imagery.

Single Bit-Map Block Truncation Coding

Single Bit-Map Block Truncation Coding is a colour image data compression algorithm. BTC is first implemented to encode a weighted trichromatic image. The bit-map generated is then used to compress each of the R, G and B images separately, removing the inter-colour correlations. The weighted trichromatic image is generated by properly weighting the R, G and B images so that the MSE between

the original and reconstructed R, G and B images is minimized. Another approach using a bit-wise majority rule on the R, G and B BTC bit-maps has also been discussed. The algorithm preserves the spatial details in the image content and has low computational complexity. An 8:1 compression can be achieved. The truncated data is further compressed using adaptive DCT. Differential coding and error feedback techniques have also been implemented to reduce the bit rate and improve the MSE performance. The total compression ratio is about 12:1. By exploiting human visual system characteristics, a 14:1 compression of NTSC colour TV signals can be achieved.

8.2 Suggestions fo Futher Research Work

Some future research work based on the above coding algorithms are:

1. In "Multi-Level BTC", block sizes and quantization levels can be made adaptive to the local statistics which will either reduce the bit rate or improve the error performances, i.e., limiting the maximum error, limiting MSE, or MAE within each block.
2. In "Single Bit-Map BTC", MSE optimization is relative to the whole image. A block-wise MSE optimization might further reduce the MSE. This will increase the computational complexity. Neural networks might be a good candidate to find the best quantization bit-map.
3. In "Single Bit-Map BTC", instead of optimizing MSE, NMSE is another parameter to be optimized.
4. In "Single Bit-Map BTC", some work can be done to further remove the inter-colour correlations among subsampled planes, i.e., using VQ to code those

planes.

5. All of the coding algorithms introduced in this thesis are BTC based and are used to code the still images. They can be implemented to code sequential images, such as occur in TV and teleconferencing images. Motion compensation techniques can be implemented on the BTC bit-map, which will reduce the computational complexity substantially, since Hamming distance can be used as distortion measure instead of Euclidean distance.

Bibliography

- [1] Y. Wu and D. C. Coll, "BTC-VQ-DCT hybrid coding of digital images," submitted to *IEEE Trans. Commun.*
- [2] Y. Wu and D. C. Coll, "Multi-level block truncation coding for high fidelity compression of composite imagery," in *Proc. Canadian Conference on Electrical and Computer Engineering*, Vancouver, B.C., Canada, 1988.
- [3] D. C. Coll and Y. Wu, "High fidelity compression of composite meteorological imagery," *Proc. 14th Biennial Symposium on Communications*, Kingston, Ontario, Canada, May 1988.
- [4] Y. Wu, *Data Compression of Meteorological Satellite Photo Imagery for Transmission*, M. Eng. Thesis, Carleton University, 1986.
- [5] Y. Wu and D. C. Coll, "Single bit-map block truncation coding of digital images," submitted to *IEEE Trans. Commun.*
- [6] Wu, Y. and Coll, D. C., "Multi-Level Block Truncation Coding Using a Minimax Error Criterion for High Fidelity Compression of Digital Images", to be presented at *1990 Picture Coding Symposium*, Cambridge, Mass., USA, March 1990.
- [7] D. C. Coll, H. J. Schneider and D. M. Taylor, "A multi-purpose display station for meteorological products," *3rd International Conference on Interactive Information and Processing Systems for Meteorology, Oceanography and Hydrology*, New Orleans, LA. Jan. 1987.
- [8] R. W. Hamming, *Coding and Information Theory*, Prentice-Hall, 1980.
- [9] D. A. Huffman, "A method for the construction of minimum-redundancy codes," *Proc. IRE*, Sept. 1952, pp. 1098-1101.

- [10] R. G. Gallager, "Variations on a theme by Huffman", *IEEE Trans. on Inform.*, vol. 1, '4, Nov. 1978, pp. 668-674.
- [11] G. V. Cormack and R. N. Horspool, "Algorithms for adaptive Huffman codes", *Information Processing Letters*, March. 1984, pp. 159-165.
- [12] D. E. Knuth, "Dynamic Huffman coding," *Journal of Algorithms*, 1985, pp. 163-180.
- [13] N. S. Jayant and P. Noll, *Digital Coding of Waveforms — Principles and Applications to Speech and Video*, Prentice-Hall, 1984.
- [14] R. C. Gonzalez and P. Wintz, *Digital Image Processing*, Addison-Wesley, 1977.
- [15] I. Song and S. Ann, "An upper bound on run length coding entropy," *Proc. IEEE*, June 1985, pp. 1141-1142.
- [16] R. Hunter and A. H. Robinson, "International digital facsimile coding standard," *IEEE Trans. Commun.*, Vol. 68, No. 7, July 1980, pp. 856-867.
- [17] CCITT, "Terminal equipment and transmission for facsimile series", Series T Recommendations, Part II, 1980.
- [18] ALDEN, "Coded digital facsimile for weather charts," Jan. 1983.
- [19] T. S. Huang, "Run-length coding and its extensions," *Picture Bandwidth Compression*, edited by T. S. Huang, O. J. Tretiak, Gordon and Breach, 1972.
- [20] H. G. Musmann and D. Preuss, "Comparison of redundancy reducing codes for facsimile transmission of documents," *IEEE Trans. Commun.*, Vol. Com-25, No. 11, Nov. 1977, pp. 1425-1433.
- [21] M. Kunt and O. Johnson, "Block coding of graphics: a tutorial review," *Proc. IEEE*, July 1980, pp. 770-785.
- [22] R. Cederberg, "Chain-link coding and segmentation for raster scan devices," *Computer Graph. Image Processing*, Vol. 10, 1979, pp. 224-234.
- [23] T. H. Morrin, "Chain-link compression of arbitrary black-white image," *Computer Graph. Image Processing*, Vol. 5, 1976, pp. 172-189.

- [24] Y. Kim and J. K. Aggarwal, "Rectangular coding for binary images," *Proc. IEEE Conf. Computer. Vis. Pattern Recogn.*, 1983, pp.108-113.
- [25] I. Song and S. A. Kassam, "A new noiseless coding technique for binary images," *IEEE Trans. Acoustics, Speech, and Signal Processing*, August 1986, pp.944-951.
- [26] H. Samet, "Region representation quadtree from binary array," *Computer Graph. Image Processing*, Vol.13, June 1980, pp.88-93.
- [27] H. Samet, "The quadtree and related hierarchical data structures", *Computer Surveys*, June 1984, pp. 187-260.
- [28] S. W. Miller and S. S. Iyengar, "Representation of regions of map data for efficient comparison and retrieval," *Proc. IEEE Conf. Computer Vis. Pattern Recogn.*, 1983, pp. 102-107.
- [29] T. S. Huang, "Coding of two-tone images," *IEEE Trans. Commun.*, Vol. 11, Nov. 1977, pp.1410-1424.
- [30] D. Ting and B. Prasada, "Digital processing technique for encoding graphics," *Proc. IEEE*, July 1980, pp. 757-769.
- [31] D. R. Spencer and T. Huang, "Bit plane encoding of continuous-tone pictures," *Symp. on Comput. Process. in Commun.*, Polytechnic Inst. Brooklyn, New York, April, 1969.
- [32] A. K. Jain, "Image data compression: a review," *Proc. IEEE*, vol. 69, March 1981, pp. 349-389.
- [33] W. K. Pratt, *Digital Image Processing*, John Wiley and Sons, 1978.
- [34] W. K. Pratt, *Image Transmission Techniques*, Academic Press, New York, 1982.
- [35] E. N. Hall, *Computer Image Processing and Recognition*, Academic Press, 1979.
- [36] A. Rosenfeld and A. C. Kak, *Digital Picture Processing*, Academic Press, 1976.
- [37] A. N. Netravali and B. G. Haskell, *Digital Picture Representation and Compression*, Plenum Press, 1988.

- [38] A. N. Netravali and J. O. Limb, "Picture coding: a review," *Proc. IEEE*, vol. 68, March 1980, pp. 366-406.
- [39] C. C. Cutler, "Differential quantization for communication signals," *U.S. Patent 2,605,961*, July 29, 1952.
- [40] A. Habibi, "Compression of nth order DPCM encoder with linear transformation and block quantization techniques," *IEEE Trans. Commun.*, Dec. 1971, pp. 948-956.
- [41] J. B. O'Neal, "Predictive quantizing differential pulse code modulation for transmission of television signals," *Bell Syst. Tech. J.*, May 1976, pp. 689-722.
- [42] D. J. Connor, R. F. Pease and W. G. Scholes, "Television coding using two-dimensional spatial prediction," *Bell Syst. Tech. J.*, March 1971, pp. 1049-1061.
- [43] P. Pirsch, "Stability conditions for DPCM coders," *IEEE Trans. Commun.*, May 1982, pp. 1174-1184.
- [44] A. Habibi, "Survey of adaptive coding techniques," *IEEE Trans. Commun.*, vol. COM-25, Nov. 1977, pp. 1275-1284.
- [45] E. J. Delp and O. R. Mitchell, "Image compression using block truncation coding," *IEEE Trans. Commun.*, vol. COM-27, Sept. 1979, pp. 1336-1342.
- [46] D. J. Healy and O. R. Mitchell, "Digital video bandwidth compression using block truncation coding," *IEEE Trans. Commun.*, vol. COM-29, Dec. 1981, pp. 1809-1817.
- [47] G. R. Arce and N. C. Gallagher, "BTC image using median filter roots," *IEEE Trans. Commun.*, vol. COM-31, June 1983, pp. 784-793.
- [48] M. D. Lema and O. R. Mitchell, "Absolute moment block truncation coding application to colour images," *IEEE Trans. Commun.*, vol. COM-32, Oct. 1984, pp. 1148-1157.
- [49] D. R. Halverson, "A generalized algorithm for image compression," *IEEE Trans. Acoust. Speech and Signal Proc.*, vol. ASSP-32, June 1984, pp. 664-668.
- [50] H. H. Ko and C. W. Lee, "Real time implementation of block truncation coding for picture data compression," in *Proc. ICASSP 87*.

- [51] J. Raby, A. Brunswick and C. Cliper, "Advantage in block truncation coding with a applications to digital terrain elevation data," in *Proc. IEEE Conf. NACOM*, 1986.
- [52] A. N. Brown, *Compression and Bilevel Display of Digital Meteorological Satellite Imagery*, M. Sc. Thesis, Carleton University, 1986.
- [53] P. J. Burt, "Fast filter transforms for image processing," *Comput. Graphics, Image Processing*, 1981, pp. 20-51.
- [54] P. J. Burt and E. H. Adelson, "Laplacian pyramid as a compact image code," *IEEE Trans. Commun.*, vol. COM-31, April 1983, pp. 532-540.
- [55] N. Ahmed, T. Natarajan and K. R. Rao, "Discrete cosine transform," *IEEE Transaction on Computers*, Jan. 1974, pp. 90-93.
- [56] J. Tribolet and R. E. Crochiere, "Frequency domain coding of speech," *IEEE Trans. Acoust. Speech and Signal Proc.*, vol. ASSP-28, April 1979, pp. 512-530.
- [57] C. T. Fike, *Computer Evaluation of Mathematical Functions*, Prentice-Hall, 1984.
- [58] N. B. Nill, "A visual model weight cosine transform for image compression and quality assessment," *IEEE Trans. Commun.*, vol. COM-33, June 1985, pp. 551-557.
- [59] H. C. Andrews, "Multidimensional rotations in feature selection," *IEEE Trans. Computers*, vol. C-20, Sept. 1971, pp. 1045-1051.
- [60] J. Pearl, H. C. Andrews, and W. K. Pratt, "Performance measures for transform data coding," *IEEE Trans. Commun.*, June 1972, pp. 411-415.
- [61] W. K. Pratt, W. Chan and L. R. Welch, "Slant transform image coding," *IEEE Trans. Commun.*, vol. COM-22, August 1974, pp. 1075-1093.
- [62] A. K. Jain, "A sinusoidal family of unitary transforms," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. PAMI-1, Oct. 1979, pp. 356-365.
- [63] Z. Wang, "Reconsideration of a fast computational algorithm for the discrete cosine transform", *IEEE Trans. Commun.*, Jan. 1983, pp. 121-123.

- [64] W. Chen and C. H. Smith, "Adaptive coding of monochrome and color images," *IEEE Trans. Commun.*, vol. COM-25, Nov 1977, pp. 1285-1292.
- [65] W. Chen and S. C. Fralick, "Image enhancement using cosine transform filtering," *Proc. Symp. Current Mathematical Problems in Image Science*, Monterey, CA, Nov. 1976, pp. 934-936.
- [66] W. Chen, C. H. Smith and S. C. Fralick, "A fast computational algorithm for the discrete cosine transform," *IEEE Trans. Commun.*, vol. COM-25, Sept. 1977, pp. 1004-1009.
- [67] M. J. Navasimha and A. M. Peterson, "On the computation of the discrete cosine transform," *IEEE Trans. Commun.*, vol. COM-26, June 1978, pp. 934-936.
- [68] R. C. Reininger and J. D. Gibson, "Distributions of the two dimensional DCT coefficients for images," *IEEE Trans. Commun.*, vol. COM-31, June 1983, pp. 835-839.
- [69] S. Yuan and K. Yu, "Zonal sampling and bit allocation of HT coefficients in image data compression," *IEEE Trans. Commun.*, vol. COM-34, Dec. 1986, pp. 1246-1251.
- [70] W. H. Chen and W. Pratt, "Scene adaptive coder," *IEEE Trans. Commun.*, vol. COM-32, Mar. 1984, pp. 225-232.
- [71] CCITT Document No. 420, "Adaptive discrete cosine transform coding scheme for still image telecommunication services", Jan. 1988.
- [72] J. A. Saghi and A. G. Tescher, "Adaptive transform coding based on chain coding concepts," *IEEE Trans. Commun.*, vol. COM-34, Feb. 1986, pp. 225-232.
- [73] A. Habibi, "Hybrid coding of pictorial data," *IEEE Trans. Commun.*, May 1974, pp. 614-624.
- [74] A. Gersho and V. Cuperman, "Vector quantization: a pattern-matching technique for speech coding," *IEEE Commu. Mag.*, Dec. 1983, pp. 15-21.
- [75] R. Gray, "Vector quantization," *IEEE ASSP Mag.*, pp. 4-29, 1984.

- [76] J. Makhoul, S. Roucos, and H. Gish, "Vector quantization in speech coding," *Proc. IEEE*, vol. COM-73, pp. 1551-1588, Nov. 1985.
- [77] N. M. Nasrabadi and R. A. King, "Image coding using vector quantization: a review," *IEEE Trans. Commun.*, vol. COM-36, pp. 957-971, Aug. 1988.
- [78] C. E. Shannon "A mathematical theory of communications," *Bell Systems Technical Journal* 27, 1948, pp. 379-423, 623-656.
- [79] C. E. Shannon "Coding theorems for a discrete source with a fidelity criterion," *IRE National Convention Record*, Part 4, 1959, pp. 142-163.
- [80] R. G. Gallager, *Information Theory and Reliable Communication*, John Wiley, New York, 1968.
- [81] T. Berger, *Rate Distortion Theory*, Prentice-Hall, Englewood Cliffs, NJ, 1971.
- [82] L. D. Davisson, "Rate-distortion theory and application," *Proc. IEEE*, July, 1972, pp. 156-164.
- [83] A. Gersho and B. Ramamurthi, "Image coding using vector quantization," *Proc. IEEE ICASSP'82*, San Diego, CA, May 1982, pp. 428-431.
- [84] R. A. King and N. M. Nasrabadi, "Image coding using vector quantization in the transform domain," *Pattern Recognition Letters*, vol. 1, July 1983, pp. 323-329.
- [85] K. Aizawa, H. Harashima and H. Miyakawa, "Adaptive discrete cosine transform coding with vector quantization for colour images," *Proc. ICASSP'86*, April 1986, pp. 985-988.
- [86] M. Goldberg and H. Sun, "Image sequence coding using vector quantization," *IEEE Trans. Commun.*, vol. COM-34, July 1986, pp. 703-710.
- [87] H. M. Hang and J. W. Woods, "Predictive vector quantization of images," *IEEE Trans. Commun.*, vol. COM-33, Nov. 1985, pp. 1208-1219.
- [88] R. L. Baker and R. M. Gray, "Differential vector quantization of achromatic imagery," *Proc. Int. Picture Coding Symp.*, March 1983.
- [89] S. E. Bridge and R. L. Baker, "Compression of colour digital images using vector quantization," *Proc. ICASSP'85*, April 1985, pp. 129-132.

- [90] M. Goldberg, P. Boucher and S. Shlien, "Image compression using adaptive vector quantization," *IEEE Trans. Commu.*, Feb. 1986, pp. 180-187.
- [91] V. R. Udpikar and J. P. Raina, "BTC image coding using vector quantization," *IEEE Trans. Commun.*, vol. COM-35, pp. 352-356, Mar. 1987.
- [92] T. Murakmi, K. Asai and E. Yamazaki, "Vector quantization of video signals," *Electron. Lett.*, vol. 7, Nov. 1982, pp. 1005-1006.
- [93] S. P. Lloyd, "Least squares optimization in PCM," *IEEE Trans. Inform. Theory*, March 1982, pp. 128-137.
- [94] Y. Linde, A. Buzo, and R. M. Gray, "An algorithm for vector quantizer design," *IEEE Trans. Commun.*, vol. COM-28, Jan. 1980, pp. 84-95.
- [95] M. J. Sabin and R. M. Gray, "Product code vector quantizers for waveform and voice coding," *IEEE Trans. Acoustics, Speech, and Signal Processing*, vol. ASSP-32, June 1984, pp. 474-488.
- [96] T. Kohonen, *Self-Organization and Associative Memory*, New York: Springer-Verlag, 1984.
- [97] T. Kohonen, "The neural phonetic typewriter," *IEEE Comput.*, vol. 21, March 1988, pp. 11-21.
- [98] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller, "Equation of state calculations by fast computing machines," *J. Chem. Phys.*, vol. 21, no. 6, June 1953, pp. 1087-1092.
- [99] S. Kirkpatrick, C. D. Gelatt, Jr., M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, May 1983, pp. 671-680.
- [100] B. Ramamurthi and A. Gersho, "Classified vector quantization of images," *IEEE Trans. Commun.*, vol. COM-34, Nov. 1986, pp. 1105-1115.
- [101] R. Aravind and A. Gersho, "Image compression based upon vector quantization with finite memory," *Optical Engineering*, July 1987.
- [102] John Allnatt, *Transmitted-Picture Assessment*, John Wiley & Sons, 1983.

- [103] R. F. Dillon, W. M. Petrusic and A. R. Moran, *A Study of the Visible Effects and User Acceptability of Data Compression Procedures When Applied to Graphic Materials*, Department of Psychology Research Report, Carleton University, 1986.
- [104] CCIR, "Method for the subjective assessment of the quality of television picture," *Recommendation (500-1)*, 14th Plenary Assembly, 1978.
- [105] S. Hecht, "The visual discrimination of intensity and the Weber-Fechner law," *J. Gen. Physiol.*, July, 1924, pp. 241.
- [106] S. S. Stevens, *Handbook of Experimental Psychology*, Wiley, New York, 1951.
- [107] A. Kunt, A. Ikonomopoulos and M. Kocher, "Second generation image coding techniques," *Proc. IEEE*, vol. 73, April 1985, pp 549-574.
- [108] R. A. McDonald "Signal to noise and ideal channel performance of DPCM systems with particular application to voice signals," *Bell Systems Technical Journal* 27, Sept. 1966, pp. 1123-1151.
- [109] R. E. Blahut *Principles and Practice of Information Theory*, Addison-Wesley, 1987.
- [110] J. R. Jain and A. K. Jain, "Displacement measurement and its application in interframe image coding," *IEEE Trans. Commun.*, vol. COM-29, Dec. 1981, pp. 1799-1808.
- [111] J. A. Roese *et al.*, "Interframe cosine transform image coding," *IEEE Trans. Commun.*, vol. COM-25, Nov. 1977, pp. 1329-1338.
- [112] L. Wang and M. Goldberg, "Lossless progressive image transmission by residual vector quantization," *Proc. the 13th Biennial Symposium on Communications*, Kingston, Ont., June 1986, pp. c.1.9-c.1.12.
- [113] A. Papoulis, *Probability, Dandom Variables, and Stochastic Processes*, McGraw-Hill, 1984.
- [114] H. Murakami, S. Matsumoto and H. Yamamoto, "Algorithm for construction of variable length code with limited maximum word length," *IEEE Trans. Commun.*, vol. COM-32, pp. 1157-1159, Oct. 1984.

- [115] H. Garten, "On variable length codes under hardware constraints," *IEEE Trans. Commun.*, vol. COM-33, pp. 491-494, May 1985.
- [116] J. P. Lameillieure and I. Bruyland, "Comments on 'Algorithm for construction of variable length code with limited maximum word length'," *IEEE Trans. Commun.*, vol. COM-34, pp. 1252-1253, Dec. 1986.
- [117] H. Gharavi, "Conditional run-length and variable-length coding of digital pictures," *IEEE Trans. Commun.*, vol. COM-35, pp. 671-677, June 1987.
- [118] C. H. Lu, "Comments on 'Algorithm for construction of variable length code with limited maximum word length'," *IEEE Trans. Commun.*, vol. COM-36, pp. 373-375, March. 1988.
- [119] C. Y. Chen, R. Kwok and J. C. Curlander, "Image coding of SAR imagery," *Proceedings of IGARSS'87 Symposium*, pp. 699-704, Ann Arbor, May 1987.
- [120] J. Max, "Quantizing for minimum distortion," *IRE Trans. Infor. Theory*, vol. IT-6, pp. 7-12, March 1960.
- [121] F. Lu and G. L. Wise, "A further investigation of Max's algorithm for optimum quantization," *IEEE Trans. Commun.*, vol. COM-33, pp. 746-751, July 1985.
- [122] S. W. Golomb, et al., *Digital Communications with Space Applications*, Englewood Cliffs, NJ: Prentice-Hall, 1964.
- [123] J. A. Storer, *Data Compression: Methods and Theory*, Computer Science Press, 1988.
- [124] N. Abramson, *Information Theory and Coding*, McGraw-Hill, 1963.
- [125] G. White, *Video Techniques*, Heinemann Professional Publishing, 1988.
- [126] W. N. Sproson, *Colour Science in Television and Display Systems*, Adam Hilger Ltd, Bristol, 1983.
- [127] T. A. Welch, "A technique for high performance data data compression," *IEEE Computer*, vol. 17, pp. 8-19, June 1984.
- [128] M. L. Loiuh, "Visual telephony as an ISDN application," *IEEE Communication Magazine*, pp. 30-38, Feb. 1990.

END

1 8/02-9 1

FIN