HYPERSPECTRAL IMAGE CODING USING 3D TRANSFORMS

DMITRY MARKMAN

HYPERSPECTRAL IMAGE

CODING USING 3D TRANSFORMS

RESEARCH THESIS

SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE
IN ELECTRICAL ENGINEERING

DMITRY MARKMAN

SUBMITTED TO THE SENATE OF
THE TECHNION - ISRAEL INSTITUTE OF TECHNOLOGY

SIVAN, 5760 HAIFA JUNE, 2000

Acknowledgements

The research thesis was carried out under the supervision of Professor David Malah in the department of Electrical Engineering.

It is a pleasure for me to thank Prof. David Malah for his great involvement, devoted guidance and invaluable help throughout all stages of the research. His passion to knowledge and rich experience were a great source of inspiration to me.

I would also like to acknowledge the staff of the Signal and Image Processing Laboratory, Nimrod, Ziva, Avi and Tamara, for supplying their assistance and the resources needed to carry out this research.

I wish to express gratitude to my parents, my sister, all my friends and especially to my best friend Ilana, for their love, encouragement, constant support and confidence in me.

The generous help of the Technion is gratefully acknowledged.

Dedicated to my family

Contents

List of Figures

List of Tables

Abstract

This research addresses the issue of hyperspectral image coding using three dimensional (3D) transforms.

A *hyperspectral image cube* (HIC) is a set of hundreds of spectral images (also called spectral bands), in which each image corresponds to a different wavelength band (visible and near-infrared) of a sensed scene. Such images cover in great detail a wide spectral window and can provide valuable information about the region under investigation.

Due to the large number of bands the HIC may occupy hundreds to thousands of MBytes, giving rise to serious problems in their archival storage and especially in their transmission from satellite to an earth station. For example, HIC of size 2000x2000 pixels containing 200 spectral bands (8 bits per pixel) occupies 763MBytes, which is definitely a huge amount of data. To solve this problem data compression must be applied.

As a first step in this research we implemented two existing state of the art techniques for hyperspectral data compression. These algorithms served as benchmarks. Then, two approaches were investigated: A 3D-DCT-based approach and a three-dimensional *shape-adaptive* DCT (3D-SA-DCT) approach.

The HIC consists of 3D spatio-spectral data, therefore, in order to obtain good performance and high compression ratios we took advantage of the redundancies inherently existing in such imagery. The HIC was partitioned into 8x8x8 spatio-spectral cubes (block-based coding). The three-dimensional discrete cosine transform (3D-DCT) is used as a basic energy compaction technique. Applying the transform, results in a 8x8x8 cube of coefficients, consisting of one DC and 511 AC transform coefficients. Two algorithms for coding of the transform coefficients are proposed and investigated.

Concerning the first algorithm, this method utilizes a 3D quantization table (QT). The idea was to generate a "standard" quantization table containing the quantization step size of a uniform scalar quantizer for each one of 512 transform coefficients. The

quantization table values can be tuned by a *quality factor* to achieve a desired quality or coding bit-rate. After the lossy quantization of the transform coefficients, each set of 8x8x8 quantized coefficients was reordered into a single vector and is further losslessly compressed utilizing *run-size Huffman tables* like in JPEG (still image coding standard).

Concerning the second algorithm, it encodes transform coefficients reordered into sub-vectors by a set of vector quantizers. This technique utilizes the scan order from the QT method, since it achieves a good energy compaction of the transform coefficients near the DC coefficient. The reordered vector is partitioned into a set of sub-vectors, and *vector quantization* (VQ) is applied to each sub-vector. This means that each sub-vector is encoded by the index of the codevector from the corresponding codebook which is most close to the original sub-vector.

A novel approach utilizing a 3D shape-adaptive DCT is also proposed. This approach has a potential to efficiently compress high activity blocks by splitting them into lower activity regions. Although not optimal, a simple variance thresholding is used as the splitting criterion in order to distinguish between high and low activity blocks. Empirical experiments were performed to obtain the threshold value achieving the best performance with a chosen criterion. Low activity blocks were coded by a straightforward 3D-DCT approach (with the proposed QT or VQ algorithms), whereas high activity blocks were split into two lower activity regions utilizing a VQ (of size 2) clustering algorithm. Each one of the two regions was transformed with 3D-SA-DCT resulting in two sets of transform coefficients, encoded by either the QT or VQ-based encoders. The codebooks for the VQ technique as well as quantization table and Huffman tables for the QT technique were tuned for both types of transform coefficients - low activity blocks transform coefficients and split blocks transform coefficients.

A very important issue in this approach is the heavy overhead (side information). In order to distinguish between the two regions, the VQ clustering algorithm produces a describing bitmap. This information originally occupies 64 bits per each split block and it may not be distorted if correct inverse shape-adaptive transformation is to be achieved. Therefore, we compressed the bitmaps losslessly utilizing a *"snake" scan* (for reordering an 8x8 matrix into a vector) followed by run-length extraction and Huffman table encoding of runs of zeros and ones. The achieved 2:1 compression still results in side-information which is too high, especially for low bit-rates.

The performance of all the examined algorithms was measured by the *peak signal-to-*

noise ratio (PSNR) versus bit-rate, which can be referred as a quantitative coding performance measure. The *classification error* versus bit-rate was a second, application-based, performance measure. A common application for hyperspectral imagery is classification or material detection. Since developing a classification algorithm was not in the focus of this research, we applied a simple maximum likelihood (ML) algorithm for measuring the performance.

Concerning the PSNR performance, all the methods proposed in this work showed better results than the implemented benchmarks, while the QT method is superior to the rest due to a variable bit allocation it permits. As for the classification performance, the VQ scheme obtains usually better results.

The shape-adaptive approach did not achieve an advantage in PSNR performance over the 3D-DCT approach with the QT scheme, but it succeeded to outperform the VQ scheme slightly. This is mainly due to a relatively high side information required for bitmap transmission. The classification performance obtained by the SA approach is usually better, compared to non SA algorithms. Moreover, this method was found in certain instances to outperform the classification of the original HIC.

List of Symbols

CR	:Compression Ratio
$DPCM$	:Differential Pulse Code Modulation
DCT	:Discrete Cosine Transform
FFT	:Fast Fourier Transform
GS	:Gain-Shape
GGD	:Generalized Gaussian Distributions
GLA	:Generalized Lloyd Algorithm
HIC	:Hyperspectral Image Cube
$IDCT$	:Inverse Discrete Cosine Transform
KLT	:Karhunen-Loeve Transform
KP	:Kronecker Product
LBG	:Linde, Buzo, Gray
ML	:Maximum Likelihood
MSE	:Mean Squared Error
$MMSE$	:Minimum MSE
$PSNR$	:Peak Signal-to-Noise Ratio
Q_f	:Quality factor
QT	:Quantization Table
SA	:Shape-Adaptive
SNR	:Signal-to-Noise Ratio
$3D$	:Three Dimensions
TCQ	:Trellis Code Quantization
VQ	:Vector Quantization

R	:bit-rate
N	:size, number of vectors, length of the vector
g	:gain
s	:shape
$\mathbf{X}$	:input vector
$\hat{\mathbf{X}}$	:reconstructed input vector
N_G	:number of vectors in gain codebook
N_S	:number of vectors in shape codebook
E	:expectation
ρ	:correlation coefficient
μ	:mean value
σ^2	:variance value
Σ	:covariance matrix
ϵ_n	:nth error image
D	:distortion
q	:number of branches leaving each state in trellis diagram
P	:probability
Γ	:Gamma function
F	:cumulative distribution function
B	:number of bands
i	:intensity value of pixels
I	:transform coefficient value
I'	:quantized transform coefficient value
q	:quantization step size
$C, \beta_{in}, \beta_{out}$	:parameters of the quantization function
$\underline{\delta}$	:constant vector
Y_0^M	:initial codebook
A	:activity value
H	:entropy
θ	:variance threshold

Chapter 1

Introduction

1.1 Statement of the Problem

Multispectral images are of interest for a large number of applications, like geology, earth resource management, meteorology, biology, medicine and military surveillance. These applications are based on the availability of several monochrome images of the same scene taken at different wavelengths, thus supplying the simultaneous knowledge of spatial and spectral domains. Often such images are composed by only a few spectral images (also called spectral bands), but some modern sensors provide up to several hundreds of bands, in which case the images are called *hyperspectral*.

The hyperspectral image cube (HIC) is a set of hundreds of spectral bands, in which each band contains the image of the same scene sensed in a different wavelength. It can be also defined as a 2D array of "spectral voxels", as illustrated in Figure **??**. Color images are a particular case of hyperspectral images, where only three spectral regions are analyzed. A typical sensor collects more than 200 visible and near-infrared wavebands. Such images cover in great detail a wide spectral window and can provide more valuable information about the land cover of the area under investigation. The applications dealing with hyperspectral imagery usually interrelate spatial and spectral information by means of image analysis. Such algorithms must be able to distinguish between image pixels having a similar spectral distribution (spectral signature). That's why clustering or classification algorithms are natural analysis tools for hyperspectral imagery.

The major drawback of hyperspectral imaging is a huge amount of data to be transmitted and/or stored. Such images usually have both high spatial resolution and high

spectral resolution (a large number of bands). For example, a hyperspectral image cube of size 2000x2000 pixels containing 200 spectral bands (8 bit per pixel) occupies 763Mbytes! To solve this problem data compression must be applied.

Hyperspectral image compression algorithms must cope with high compression ratios needs and high reconstruction fidelity for subsequent exploitation (for example, classification or material detection). Furthermore, compression schemes can not take advantage of the human visual system (HVS) properties as in still image and video coding methods. That is the reason for defining performance criteria which is based on target application needs for evaluating coding schemes.

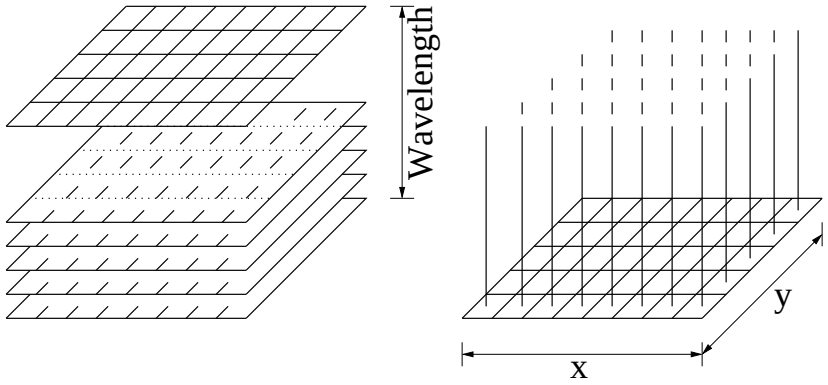


Figure 1.1:

Hyperspectral Image Cube (HIC)fig:fig18

1.2 Investigated Approaches

It is obvious that employing standard methods for still images compression, like JPEG [27, 28], can not achieve deep compression, while preserving good reconstructed image quality. Rather, techniques taking an advantage of both spatial and spectral redundancies, inherently existing in hyperspectral imagery, should be applied.

After reviewing the existing literature in the field of hyperspectral image compression, it was understood that lossless compression ([7]) is of interest here only after lossy compression (due to a low compression ratios). The Karhunen-Loeve transform-based methods ([5, 6, 30]) are not applicable because of the high computational complexity involved. In addition, there is, in principle, a difference between multispectral and hyperspectral imagery: The latter one enables more efficient coding due to higher correlation that exists

between adjacent spectral bands (high spectral resolution). Therefore, two state of the art algorithms for hyperspectral image coding ([1, 2, 3, 4]) were implemented to serve as benchmarks. The H.261 standard video coder ([40]) was also applied to hyperspectral image cubes.

In the second stage of the research we propose two techniques, based on 3D-DCT, for transform coefficients coding. The 3D-DCT, known as having good energy compaction characteristics, concentrates the dominant coefficients along the major axes near the DC coefficient. The proposed techniques introduce two different algorithms for quantization and compression of the coefficients. To preserve good reconstruction quality, the dominant coefficients must be fine quantized and the other AC coefficients may be discarded by a coarse quantization. The principal difference between the methods is that the first algorithm enables variable bit allocation, while the second one uses vector quantization (VQ) and applies a fixed rate allocation per each 8x8x8 transformed block of coefficients.

Since the 3D-DCT may not efficiently compact the energy of the high activity blocks (those having edges or texture), we propose a shape-adaptive approach for such blocks. The advantage of this approach is that active blocks can be split into lower activity regions and apply the 3D-SA-DCT on each region (followed by transform coefficients coding). The shape-adaptive transformation is also capable of concentrating the transform coefficients energy near the DC coefficient of each region, and, moreover, it can deal with non-contiguous regions. The main disadvantage of the approach is the relatively high side-information required, mostly due to the bitmap that has to be stored or transmitted for each split block for correct direct and inverse SA transformations.

Although it is not in the focus of this research, it was important to verify that the encoded images retain their characteristics for subsequent analysis. That is why a simple 3 class maximum likelihood (ML) classifier was introduced and misclassification results by the different examined algorithms were compared.

1.3 Main Contributions and Thesis Organization

The main contributions of this work can be divided into three parts:

1. Development of 3D-DCT-based schemes. Two novel techniques for coding the transform coefficients are proposed. These methods were not used earlier for hyperspectral imagery and, as shown by the simulations, they attain better results than two state of the art implemented benchmark algorithms.
2. Development of a 3D-SA-DCT-based coding scheme. The conventional 2D shape-adaptive approach was extended to 3D and a novel coding scheme is proposed. Although this scheme suffers from relatively high side-information, it still has a great potential for efficient coding of high activity blocks.
3. Considering an application oriented performance measure. We used a maximum likelihood (ML) classifier to examine the different algorithms performance, by applying the classifier on the reconstructed data.

The thesis is organized as follows:

Chapter 2 provides a brief summary of existing state of the art coding algorithms. Two such algorithms are discussed in detail and implemented. These two techniques served as benchmarks.

Chapter 3 introduces the 3D-DCT approach, and two novel algorithms for coding of the transform coefficients are proposed. The detailed schemes using a 3D quantization table (QT) and quantization via a set of vector quantizers (VQ) are presented. In addition, issues like parameter adjustment and optimization, the training of the codebooks and Huffman tables generation are explained.

Chapter 4 is devoted to the introduction of a novel approach utilizing 3D Shape-Adaptive (SA) DCT. The motivation, concept, transformation definitions and details are given. We center our discussion on the choice of the active blocks that pass through the SA scheme (splitting criterion), the algorithm that splits the chosen block into two lower activity regions and bitmap lossless compression algorithms. The main drawbacks of the approach are pointed out and a number of ways to deal with them are proposed too.

Chapter 5 presents the performance measures used for examining and comparing the different algorithms. A classification application is described. All the coding and classification results for all the proposed approaches and algorithms are summarized and compared to the related performance of the benchmarks.

Finally, **Chapter 6** provides conclusions drawn from the work presented in this thesis.

The main contributions are summarized and issues for further research are outlined.

Chapter 2

Existing Techniques for Hyperspectral Data Compression

In this chapter we review the literature related to multispectral and hyperspectral image compression. Various algorithms and techniques dealing with hyperspectral image coding are briefly reviewed. The two most promising algorithms are described in more detail and are implemented as benchmarks to which the algorithms proposed in this research will be compared to.

2.1 Introduction

A number of algorithms were recently proposed for compressing of multispectral and hyperspectral images. While Ryan and Arnold in [7] use lossless compression technique, we will be interested in this research in lossy compression algorithms. The main reason for this is that processing hyperspectral imagery means dealing with a huge amount of data requiring both a large storage space and a wide transmission bandwidth. Since we wish to save as much of these resources as possible we need to compress this kind of imagery as deep as is enabled by some quality criterion. Unfortunately, lossless compression does not permit large compression ratios, in contrast to the lossy coding allowing much greater compression by sacrificing the reconstructed image quality.

There are two main approaches in hyperspectral image coding algorithms presented recently: Variations on vector quantization (VQ) and variations on transform coding, sometimes combined with predictive coding like differential pulse-code modulation (DPCM)

technique. In particular, these techniques utilize spatial (intraband) and spectral (interband) redundancies inherently existing in such imagery to get higher compression ratios. In [5] Gelli and Poggi use tree-structured VQ to classify pixel spectra, then perform Karhunen-Loeve Transform (KLT) in the spectral domain (with different KLT matrix for each class) and finally two-dimensional discrete cosine transform (2D-DCT) in the spatial domain. Lee in [6] proposes quadtree optimization algorithm for determining transform block sizes, use KLT in spectral domain (with different KLT matrix for each block) before 2D-DCT is applied on transform coefficients in the spatial domain. Both algorithms were presented for multispectral image coding and have a great computational complexity (due to KLT matrices calculations, quadtree optimization and classification by VQ).

On the other hand, in [1, 2, 3, 4] algorithms of moderate coding complexity that were designed especially for hyperspectral imagery coding are presented. Canta and Poggi in [1] obtain a significant complexity reduction and low bit-rate coding by representing each reshaped three-dimensional (3D) block as a Kronecker-product of *gain* and *shape* vectors and by quantizing these component shape and gain vectors. This technique, also called *Kronecker-product Gain-Shape VQ* (KP-GS-VQ), can provide high compression ratio by applying a *shape-invariance* assumption, which is correct only in hyperspectral and is not true in multispectral cubes (since high spectral correlation enables using the same shape for a number of adjacent bands). Abousleman *et al.* in [2, 3, 4] use a 2D spatial DCT (or DWT-discrete wavelet transform) followed by interband predictive coding of the coefficients or, as an alternative, DCT on 3D blocks. *Entropy-constrained trellis coded quantization* (ECTCQ) is utilized to better encode transform coefficients.

In the following sections we describe in more details two of the existing algorithms that were chosen to be implemented as a preparation stage for the research. The main purpose of this task is to check the relative performances of these algorithms under the same performance measure and encoding the same image scenes (or hyperspectral cubes). Moreover, these two algorithms are supposed to serve us as benchmarks for performance comparison with the techniques proposed in this thesis. Another goal is to show that these reported algorithms can achieve much better performance by utilizing spectral (interband) pixel dependencies. So, it is worthwhile to compare encoding results of these techniques to existing standard algorithms for still image coding, like JPEG.

The best candidates for this purpose were chosen to be the algorithm of [1] and a variation on the algorithms presented in [2, 3, 4] for their best relevance to hyperspectral

imagery, moderate coding complexity and very good performance reported.

2.2 Kronecker-product Gain-Shape VQ Algorithm

2.2.1 Motivation

Most of the algorithms for hyperspectral image coding presented recently ([1, 2, 3, 4, 5, 6, 7]) utilize *transform coding*, while performing KLT in spectral (interband) and then 2D-DCT (or 2D-DWT) in spatial (intraband) domains. Although transform coding techniques may assume that images are stationary and are characterized only by linear dependencies, both assumptions are grossly inaccurate for natural images. As a result, those techniques will not achieve optimal performance. It is a known fact that the *vector quantization* technique (VQ) is optimal among all block based coding techniques. Unfortunately, the computational complexity of VQ grows exponentially with block size, consequently enforcing the usage of VQ with small blocks and restricting compression performance by this technique. To overcome such a limitation, one can resort to some form of constrained VQ, trading strict optimality for reduced complexity. In this way, one can use larger blocks, thereby exploiting the statistical dependence among many more pixels, with the goal of overcompensating the lack of optimality. The actual success of such an approach depends on the constrained model considered, which should simplify the encoding task without destroying the statistical dependencies in the block.

In paper [1], Canta and Poggi propose a *generalized gain-shape* model that fits well to the case of multispectral images, thus allowing for a huge computational saving with negligible performance impairment. The authors build upon the working hypothesis that the shape do not vary significantly from band to band and is well approximated by a constant vector, while all variations are concentrated in the gain term. In such a hypothesis, encoding the input block involves selecting only a small-size shape codevector, common to all bands in this block, plus selecting a gain codevector that encodes the highly dependent gains. Further developing the encoding scheme the authors suggest an additional step in reducing complexity, in which prior to encoding, each input block is *represented* in (vector) gain-shape form and then the gain and shape codevectors are selected with reference to such a synthetic representation. It is obvious that such an approximation will impair the quality of the reconstructed image, but, if the *shape-invariance* assumption is

reasonably accurate, which is to say, if the spectral resolution is high, this additional quality impairment turns out to be totally negligible. On the contrary, this latter scheme reduces the complexity dramatically comparing to unconstrained VQ and even to simple GSVQ.

2.2.2 Background

It is known, [16], that in VQ, for encoding K- dimensional input vector X with rate $R = \log_2 N/K$, one needs to perform $N = 2^{R \cdot K}$ distortion calculations, while N is the number of vectors in the codebook. It is obvious that increasing block size on the one hand will result in exponential increase in computational complexity, but, on the other hand, will improve compression rate, so VQ is inherently limited in its performance. It turns out that, for a given computational complexity, suboptimal techniques can often outperform VQ by simply using larger blocks.

One of the most popular such techniques is *Gain-Shape Vector Quantization* (GSVQ). In GSVQ, for each input vector (block) $\mathbf{x}$ the gain $g = \|\mathbf{x}\|$ is extracted and scalar quantized using a codebook of gains $\mathcal{G} \equiv \{\hat{g}_1, \hat{g}_2, \ldots, \hat{g}_{N_G}\}$, while the unit-norm shape vector $\mathbf{s} = \mathbf{x}/g$ is quantized separately using a codebook of shapes $\mathcal{S} \equiv \{\hat{s}_1, \hat{s}_2, \ldots, \hat{s}_{N_S}\}$. Now the input vector is represented as the product of optimal gain and shape, and an optimal encoder will choose the indices i and j that will minimize the squared error between the input vector and its reproduction:

$$\|\mathbf{x} - \hat{g}_i \cdot \hat{\mathbf{s}}_j\|^2 = \|\mathbf{x}\|^2 + \hat{g}_i^2 \cdot \|\hat{\mathbf{s}}_j\|^2 - 2 \cdot \hat{g}_i \cdot \langle \mathbf{x}, \hat{\mathbf{s}}_j \rangle \tag{2.1}$$

where $\langle \cdot, \cdot \rangle$ indicates inner product. Since $\|\mathbf{x}\|$ is independent of i and j, and $\|\hat{\mathbf{s}}_j\| = 1$ the minimization problem reduces to finding first index j that maximizes the inner product $\langle \mathbf{x}, \hat{\mathbf{s}}_j \rangle$, and subsequently the index i that minimizes $\hat{g}_i^2 - 2 \cdot \hat{g}_i \cdot \langle \mathbf{x}, \hat{\mathbf{s}}_{opt} \rangle$, as shown in Fig. 2.1.

An algorithm, that closely resembles the *generalized Lloyd algorithm* (GLA) [16], is available for jointly optimal gain and shape codebook design. But, as experiments showed, due to the weak dependence of gain and shape, it is reasonable and much less complicated to design optimal codebooks for gain and shape independently of one another. It must be obvious now, that the computational complexity was reduced from N distortion calculations in standard VQ, to N_S (which is the number of vectors in the shape codebook,

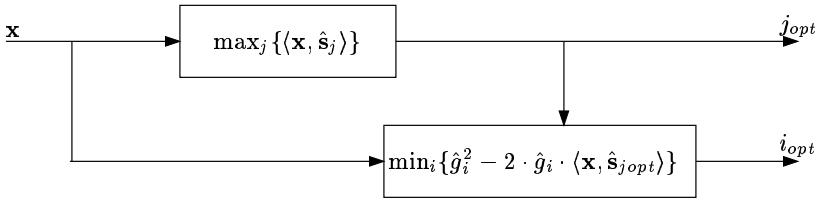


Figure 2.1: Encoding scheme for GSVQ

N_G is negligible) in GSVQ. Namely, $N_S \ll N$ because shape vectors, normalized by the gain, have smaller variability, so less representation vectors in the codebook are needed.

2.2.3 The Reported Algorithm

As mentioned above, a hyperspectral image is a set of images of exactly the same scene in different spectral windows. So, the basic assumption is that the shape of the specific block in the specific position in the image does not vary significantly from one spectral band to another and only the gain will represent the reflectance differences between the adjacent bands.

The first step in the model is GSVQ. Let's define a multispectral input vector of length $K \cdot B$ as a collection of K pixels from the same spatial location, gathered from B adjacent spectral bands

$$\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_b, \ldots, \mathbf{x}_B) \quad \forall i,\ \mathbf{x}_i \in \mathcal{R}^K \tag{2.2}$$

Then:

- Define a *gain* as $g_b = \|\mathbf{x}_b\|$ and calculate the gain vector

$$\mathbf{g} = (g_1, g_2, \ldots, g_B) \tag{2.3}$$

- Define a *shape* as $\mathbf{s}_b = \mathbf{x}_b / g_b$ and calculate the shape vectors

$$\mathbf{S} = (\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_B) \tag{2.4}$$

- Codebooks for gain and shape are given by

$$\mathcal{G} \equiv \{\hat{\mathbf{g}}_1, \hat{\mathbf{g}}_2, \ldots, \hat{\mathbf{g}}_{N_G}\} \qquad \mathcal{S}^B \equiv \{\hat{\mathbf{S}}_1, \hat{\mathbf{S}}_2, \ldots, \hat{\mathbf{S}}_{N_S}\} \tag{2.5}$$

  where $\hat{\mathbf{S}}_j = (\hat{\mathbf{s}}_{j1}, \hat{\mathbf{s}}_{j2}, \ldots, \hat{\mathbf{s}}_{jB})$ and $\|\hat{\mathbf{s}}_{jb}\| = 1$.

- The optimal encoder, if so, will find the best pair of codevectors $\hat{\mathbf{g}}_i$, $\hat{\mathbf{S}}_j$ that will achieve minimum squared error

$$\|\mathbf{X} - \hat{\mathbf{X}}_{ij}\|^2 = \sum_{b=1}^{B} \|\mathbf{x}_b - \hat{g}_{ib} \cdot \hat{\mathbf{s}}_{jb}\|^2 = \sum_{b=1}^{B} \|\mathbf{x}_b\|^2 + \sum_{b=1}^{B} \hat{g}_{ib}^2 \cdot \|\hat{\mathbf{s}}_{jb}\|^2 -$$

$$- \sum_{b=1}^{B} 2 \cdot \hat{g}_{ib} \cdot \langle \mathbf{x}_b, \hat{\mathbf{s}}_{jb} \rangle = \|\mathbf{X}\|^2 + \|\hat{\mathbf{g}}_i\|^2 - 2 \cdot \langle \mathbf{g}_i, \hat{\mathbf{z}}_j \rangle \tag{2.6}$$

where

$$\hat{\mathbf{z}}_j = (\langle \mathbf{x}_1, \hat{\mathbf{s}}_{j1} \rangle, \langle \mathbf{x}_2, \hat{\mathbf{s}}_{j2} \rangle, \ldots, \langle \mathbf{x}_B, \hat{\mathbf{s}}_{jB} \rangle) \tag{2.7}$$

To encode a size $K \cdot B$ multispectral input vector by GSVQ technique, the encoder has to minimize the quantity $\|\hat{\mathbf{g}}_i^2\| - 2 \cdot \langle \mathbf{g}_i, \hat{\mathbf{z}}_j \rangle$ over i and j, and therefore has to:
- evaluate N_S vectors $\hat{\mathbf{z}}_j$ ($N_S K B$ products);
- maximize over j the scalar product $\langle \mathbf{g}_i, \hat{\mathbf{z}}_j \rangle$ for all i's ($N_G N_S B$ products);
- minimize over i the sum $\|\hat{\mathbf{g}}_i^2\| - 2 \cdot \langle \mathbf{g}_i, \hat{\mathbf{z}}_{iopt} \rangle$ (negligible complexity).

The overall per-pixel complexity is therefore proportional to $N_S N_G / K + N_S$ in GSVQ as opposed to $N_S N_G$ in VQ for an unstructured codebook at the same rate.

Despite the significant saving, the computational complexity still remains heavy when large codebooks are used. This is mainly due to the fact that optimal shape and gain cannot be chosen independently of one another and, moreover, quite large multispectral shape vectors are generated even for small blocks and few spectral bands.

At this stage, the authors propose to take advantage of the high spectral resolution of hyperspectral images in order to assume the **shape-invariance principle** for further computational burden reduction.

As the first step let's assume that the shape multispectral codevectors are composed of a single-band shape replicated B times, namely

$$\hat{\mathbf{S}}_j = (\hat{\mathbf{s}}_j, \hat{\mathbf{s}}_j, \ldots, \hat{\mathbf{s}}_j), \qquad j = 1, \ldots, N_S \tag{2.8}$$

and the shape codebook from (2.5) turns to be

$$\mathcal{S} \equiv \{\hat{\mathbf{s}}_1, \hat{\mathbf{s}}_2, \ldots, \hat{\mathbf{s}}_{N_S}\}. \tag{2.9}$$

The generic input vector $\mathbf{X}$ is therefore encoded by a codevector, $\hat{\mathbf{X}}_{ij}$, obtained as the **Kronecker product** of a gain vector $\hat{\mathbf{g}}_i$ and a (single-band) shape vector $\hat{\mathbf{s}}_j$

$$\hat{\mathbf{X}}_{ij} = \hat{\mathbf{g}}_i \otimes \hat{\mathbf{s}}_j = (\hat{g}_{i1} \cdot \hat{\mathbf{s}}_j, \hat{g}_{i2} \cdot \hat{\mathbf{s}}_j, \ldots, \hat{g}_{iB} \cdot \hat{\mathbf{s}}_j) \tag{2.10}$$

In the next step of the algorithm the authors decided to represent the input vector $\mathbf{X}$ in a Kronecker product form, $\tilde{\mathbf{X}} = \mathbf{g} \otimes \mathbf{s}$, prior to encoding it, with the usual condition $\|\mathbf{s}\| = 1$. It is clear now that since the former assumption and the approximation of the input vector will lead to codevectors that are no longer optimal for the given input, they will impair the quality of the reconstructed image. The loss in performance will of course depend on the character of the image and can be determined only a posteriori by numerical experiments. On the other hand, the encoding procedure had been significantly simplified, since from the squared error expression

$$\begin{aligned} \|\tilde{\mathbf{X}} - \hat{\mathbf{X}}_{ij}\|^2 &= \|\mathbf{g} \otimes \mathbf{s} - \hat{\mathbf{g}}_i \otimes \hat{\mathbf{s}}_j\|^2 = \|\mathbf{g} \otimes \mathbf{s}\|^2 + \|\hat{\mathbf{g}}_i \otimes \hat{\mathbf{s}}_j\|^2 - 2 \cdot \langle \mathbf{g} \otimes \mathbf{s}, \hat{\mathbf{g}}_i \otimes \hat{\mathbf{s}}_j \rangle = \\ &= \|\mathbf{g}\|^2 \cdot \|\mathbf{s}\|^2 + \|\hat{\mathbf{g}}_i\|^2 \cdot \|\hat{\mathbf{s}}_j\|^2 - 2 \cdot \sum_{b=1}^{B} g_b \cdot \hat{g}_{ib} \cdot \langle \mathbf{s}, \hat{\mathbf{s}}_j \rangle = \\ &= \|\mathbf{g}\|^2 + \|\hat{\mathbf{g}}_i\|^2 - 2 \cdot \langle \mathbf{g}, \hat{\mathbf{g}}_i \rangle \cdot \langle \mathbf{s}, \hat{\mathbf{s}}_j \rangle \end{aligned} \tag{2.11}$$

it appears that the minimization over gain and shape is almost completely decoupled.

Now, to encode a size $K \cdot B$ multispectral input vector by KP-GSVQ technique, the encoder has to minimize the quantity $\|\hat{\mathbf{g}}_i\|^2 - 2 \cdot \langle \mathbf{g}, \hat{\mathbf{g}}_i \rangle \cdot \langle \mathbf{s}, \hat{\mathbf{s}}_j \rangle$ over i and j, and therefore has to:

- evaluate N_S inner products $\langle \mathbf{s}, \hat{\mathbf{s}}_j \rangle$, which means to evaluate $N_S K$ products and to find j that satisfies the maximum inner product;
- evaluate N_G inner products $\langle \mathbf{g}, \hat{\mathbf{g}}_i \rangle$, which means to evaluate $N_G B$ products and to find i that minimizes $\|\hat{\mathbf{g}}_i\|^2 - 2 \cdot \langle \mathbf{g}, \hat{\mathbf{g}}_i \rangle \cdot \langle \mathbf{s}, \hat{\mathbf{s}}_j \rangle$, which has a negligible complexity.

The overall per-pixel complexity is reduced to $N_S/B + N_G/K$ only in KP-GSVQ!

In Appendix A it is proven that the optimal representation of $\mathbf{X}$ by $\tilde{\mathbf{X}}$ requires the optimal shape $\mathbf{s}$ to be the dominant eigenvector (corresponding to the maximum eigenvalue) of the matrix

$$\mathbf{R}_X = \sum_{b=1}^{B} \mathbf{x}_b^T \cdot \mathbf{x}_b \tag{2.12}$$

and the optimal gain vector is the projection of $\mathbf{X}$ on $\mathbf{s}$

$$\mathbf{g} = (\langle \mathbf{x}_1, \hat{\mathbf{s}} \rangle, \langle \mathbf{x}_2, \hat{\mathbf{s}} \rangle, \ldots, \langle \mathbf{x}_B, \hat{\mathbf{s}} \rangle). \tag{2.13}$$

The eigenvectors are calculated by simple iterative algorithm and the additional computational overhead, due to Kronecker product representation of the input vector $\mathbf{X}$, is negligible compared to encoding.

The design of gain and shape codebooks $\mathcal{G}, \mathcal{S}$ can be done with joint optimality, but, like in ordinary GSVQ, the isolated design of the codebooks is also a viable alternative and often does not significantly impair performance.

The whole encoding and decoding procedures for GSVQ with Kronecker product representation are summarized here:

Encoding procedure

Given the input block $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_B)$ and the codebooks
$\mathcal{G} \equiv \{\hat{\mathbf{g}}_1, \hat{\mathbf{g}}_2, \ldots, \hat{\mathbf{g}}_{N_G}\}$ and $\mathcal{S} \equiv \{\hat{\mathbf{s}}_1, \hat{\mathbf{s}}_2, \ldots, \hat{\mathbf{s}}_{N_S}\}$

1) Form the matrix $\mathbf{R}_X = \sum_{b=1}^{B} \mathbf{x}_b^T \cdot \mathbf{x}_b$;
2) Find the optimal representation shape $\mathbf{s} : \mathbf{R}_X \mathbf{s}^T = \lambda_1 \mathbf{s}^T$, with λ_1 the maximum eigenvalue;
3) Find the optimal representation gain $\mathbf{g} = (\langle \mathbf{x}_1, \hat{\mathbf{s}} \rangle, \langle \mathbf{x}_2, \hat{\mathbf{s}} \rangle, \ldots, \langle \mathbf{x}_B, \hat{\mathbf{s}} \rangle)$;
4) Find the index of the optimal shape codevector

$$j_{opt} = arg \max_{j=1,\ldots,N_S} \langle \mathbf{s}, \hat{\mathbf{s}}_j \rangle;$$

5) Find the index of the optimal gain codevector

$$i_{opt} = arg \min_{i=1,\ldots,N_G} [\|\hat{\mathbf{g}}_i\|^2 - 2 \cdot \langle \mathbf{g}, \hat{\mathbf{g}}_i \rangle \cdot \langle \mathbf{s}, \hat{\mathbf{s}}_{jopt} \rangle].$$

Decoding procedure

Given the indices i_{opt} and j_{opt}, and the codebooks
$\mathcal{G} \equiv \{\hat{\mathbf{g}}_1, \hat{\mathbf{g}}_2, \ldots, \hat{\mathbf{g}}_{N_G}\}$ and $\mathcal{S} \equiv \{\hat{\mathbf{s}}_1, \hat{\mathbf{s}}_2, \ldots, \hat{\mathbf{s}}_{N_S}\}$

1) Reproduce the input block as

$$\hat{\mathbf{X}}_{ij} = \hat{\mathbf{g}}_{iopt} \otimes \hat{\mathbf{s}}_{jopt}.$$

2.2.4 Reported Results

The authors carried out a number of experiments to assess the performance of the proposed technique and to compare it with other known algorithms. For this purpose two kinds of

images were used:

(1) Hyperspectral image consisting of 63 spectral bands.

(2) Multispectral image consisting of only 6 spectral bands.

The distortion measure is defined as a mean squared error (MSE) or signal-to-noise ratio (SNR)

$$MSE = E[(x - \hat{x})^2] \quad ; \quad SNR = 10log_{10}\frac{E[x^2]}{MSE} = 10log_{10}\frac{E[x^2]}{E[(x - \hat{x})^2]} \tag{2.14}$$

where E denotes expectation, x and $\hat{x}$ are the original and distorted versions of the signal, respectively. In practice, since the statistical model is unknown, the arithmetic average is used instead of expectation. In this case of multispectral imagery it makes sense to define an average performance measurements over B spectral bands of the image (under the normalized-power condition)

$$\overline{R} = \frac{1}{B}\sum_{b=1}^{B} R_b \quad ; \quad \overline{MSE} = \frac{1}{B}\sum_{b=1}^{B} MSE_b \quad ; \quad \overline{SNR} = 10log_{10}\frac{1}{\overline{MSE}} \tag{2.15}$$

The experiments were carried out with the following parameters:

(1) $K = 16$ pixels (square 4×4 spatial blocks).

(2) $B = 4, B = 8$ and $B = 16$ corresponding to encoding rates of $R = 0.25, R = 0.125, R = 0.0625$ bpp and, therefore, compression ratios (CR) of $CR = 32 : 1, CR = 64 : 1, CR = 128 : 1$, respectively.

Hyperspectral Image

First of all, the **representation** quality of the original image as the Kronecker product of gain and shape was investigated. The average $\overline{SNR}_{rep}$ is between $25dB$ (for $B = 4$) and $24.2dB$ (for $B = 16$) . As expected, the representation is good for small values of B, since in that case the shape-invariance assumption is correct.

Secondly, the **encoding** performance was checked for gain and shape codebook sizes of $N_G = 64$ and $N_S = 1024$ codevectors, respectively. It is obvious, that as in the representation case, the better performance is achieved for smaller B, and that was shown by the encoding results. The average $\overline{SNR}_{cod}$ reported is between $23.24dB$ (for $B = 4$) and $22.59dB$ (for $B = 16$). The small variability of quality (SNR measure) means that for a small impairment in the quality of the reconstructed image, the significant compression ratio can be achieved.

The proposed algorithm reduces significantly the computational burden of the encoding. For example, with KP-GSVQ, it is possible to encode blocks of $4 \times 4 \times 16$ pixels at a computational cost of less than 100 products/pixel; in the same situation, unconstrained VQ would require more than 10000 products/pixel, ruling it out from consideration.

The authors also tried to consider the influence of the codebook sizes on the SNR. They revealed that for the given encoding rate (calculated from the given product $N_G N_S$) the performance depends on the bit allocation between gain and shape codebook. The optimal values were empirically set to be $N_G = 64, N_S = 1024$.

Multispectral Image (TM)

All the experiments were repeated for multispectral image, with the only difference $B = 6$. The representation and encoding average signal-to-noise ratios were measured for $N_G = 64, N_S = 1024$. Although the multispectral image visually is less noisy than the hyperspectral one, the average representation SNR is only $23dB$ and it changes dramatically from $28.6dB$ in band 2 to $18.9dB$ in band 6. Also, the encoding average SNR is only $19.5dB$ at the rate of 0.167 bpp.

The conclusion from these results is that the *shape-invariance assumption* has a cardinal role in the proposed algorithm, therefore, the algorithm fails in multispectral image coding, since high resolution in spectra (i.e. many spectral bands) is not available in this imagery.

2.3 Hybrid DPCM/DCT with ECTCQ Algorithm

2.3.1 Motivation

As mentioned in the previous section, many of the algorithms, proposed recently, utilize transform coding techniques. It is known that the Karhunen-Loeve transform (KLT) is optimal for completely decorrelating the input data (under the stationarity assumption). Although KLT is used in some algorithms [5, 6], optimality on one hand requires heavy computational complexity on the other hand, which is often a very important performance measure, like in the case of online applications. The solution of this problem is to use suboptimal transforms, that are much easier to implement and asymptotically converge

to KLT. Discrete cosine transform (DCT) is one of such transforms, with some attractive features: DCT is a real transform, it can be calculated efficiently, for example by the fast Fourier transform (FFT) algorithm or fast DCT algorithm, and it asymptotically converges to KLT.

Abousleman and collaborators in [2, 3, 4] present an algorithm for hyperspectral image compression, that uses pure transform coding and also hybrid differential pulse code modulation (DPCM) with DCT techniques. The former system performs 3D-DCT on the blocks (cubes) of the hyperspectral image in order to utilize both spectral and spatial dependencies inherently existing in such an imagery. Since the DCT needs to be evaluated in all the three dimensions, the system has quite a high computational complexity. However, the latter systems reduces the complexity to moderate level by using DPCM in the spectral domain (prediction between adjacent spectral bands) and 2D-DCT (or 2D-DWT) in the spatial domain (decorrelating transform on error band images). In both systems the resulting transform coefficient are then better encoded by entropy-constrained trellis coded quantizer (ECTCQ).

In this summary we will concentrate on the second type of systems presented above, namely hybrid DPCM/DCT system and this is mostly due to its moderate complexity and the very good results reported by the authors.

2.3.2 Encoding Scheme

The schematic diagram of the hybrid DPCM/DCT system with ECTCQ is shown in Fig. 2.2.

The utilization of DPCM to exploit the spectral correlations of hyperspectral imagery is straightforward. For B-band image of size $N \times N$, an ordinary DPCM loop can be employed to encode each one of N^2 pixel sequences of length B. In their implementation the authors use the optimum (in MMSE sense) first-order linear predictor for nonzero-mean sequences, which is given by

$$\tilde{x}_{n|n-1} = \rho \cdot x_{n-1} + \mu \cdot (1 - \rho) \tag{2.16}$$

where μ and ρ are the mean and correlation coefficient of the sequence, respectively, and $\tilde{x}_{n|n-1}$ is the predicted value of x_n. In this case N^2 pixel means must be calculated and transmitted as a side-information. After testing a hyperspectral data, it was revealed that the spectral correlation coefficient ρ, for any pixel in the image is approximately

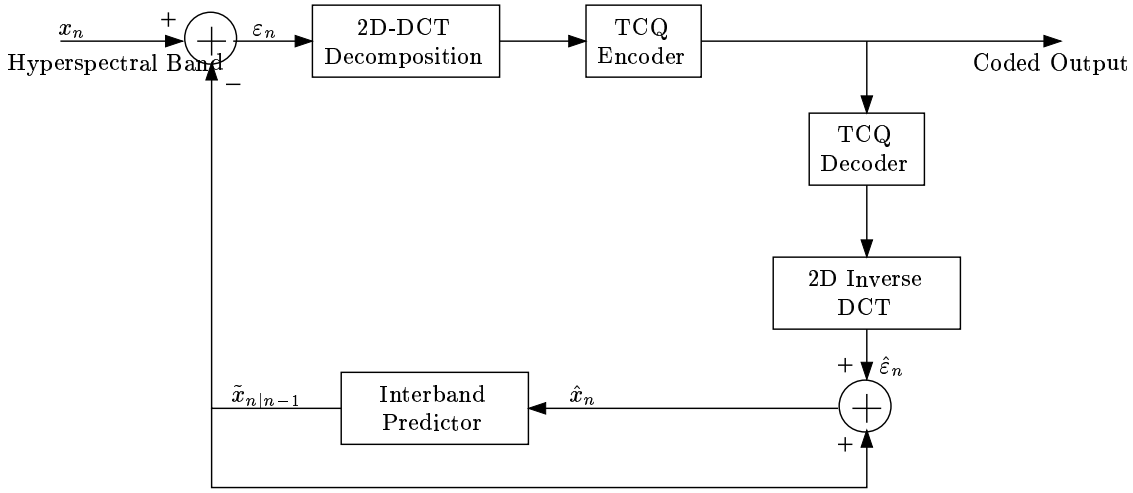


Figure 2.2: Hybrid DPCM/DCT hyperspectral image encoder

0.95. Accordingly, this value was used in the DPCM loop. If the prediction is good the difference, $\varepsilon_n = x_n - \tilde{x}_{n|n-1}$, is supposed to be significantly smaller in magnitude, on average, than x_n, therefore, fewer bits will be required to encode the error sequence. While the DPCM loop exploits the spectral correlations between the adjacent bands, it can not exploit the spatial correlation between the pixels in the same band. For this goal the *hybrid* system is used, which includes 2D-DCT inside the DPCM loop. The error images are partitioned into 8×8 blocks and the 2D-DCT is performed on each block. Transform coefficients, corresponding to the same position within each block (so-called *like-coefficients*) are collected into sequences, that are normalized. The DC sequence is normalized by subtracting its mean and dividing by its standard deviation, while the non-DC sequences are just divided by their standard deviations, since they are assumed to have zero mean. In case of 8×8 blocks, the 64 coefficient sequences (one DC and 63 non-DC) are further encoded by ECTCQ scheme. The means and standard deviations are transmitted as a side-information to the decoder. Note, that the error image must be decoded within the encoder loop (TCQ decoder and 2D Inverse DCT) so that the quantized image, $\hat{x}_n$, can be constructed and used to predict the next image.

Rate Allocation

The authors use the rate allocation algorithm described in [12]. That technique uses the rate-distortion performance of different quantizers to provide a near-optimal allocation of

bits, given an overall bit quota. In our implementation we used the sub-optimal, but less complicated algorithm, presented in [13]. This algorithm will be described in more detail in the following subsection.

ECTCQ Scheme

The main advantage of Trellis Coded Quantization encoding over VQ encoding is the reduced complexity and memory requirements ($O(NR)$-linear in TCQ instead of $O(2^{NR})$-exponential in VQ, for encoding a vector of length N with bit-rate R), and, therefore, quite long input vectors can be encoded by TCQ and cannot be encoded by VQ. More detailed comparison between these two techniques can be found in [19]. The TCQ scheme consists of a 4-state trellis encoder, based on [11, 10, 9, 8]. For encoding the memoryless source at R bits per sample, a codebook of size 2^{R+1} is partitioned into four subsets, each containing 2^{R-1} codewords (the partitioning is for each stage of trellis). These subsets are labeled D_0, D_1, D_2, D_3, and are used as labels for the branches of a suitably chosen trellis. An example is shown in Fig. 2.3 for $R = 2$.

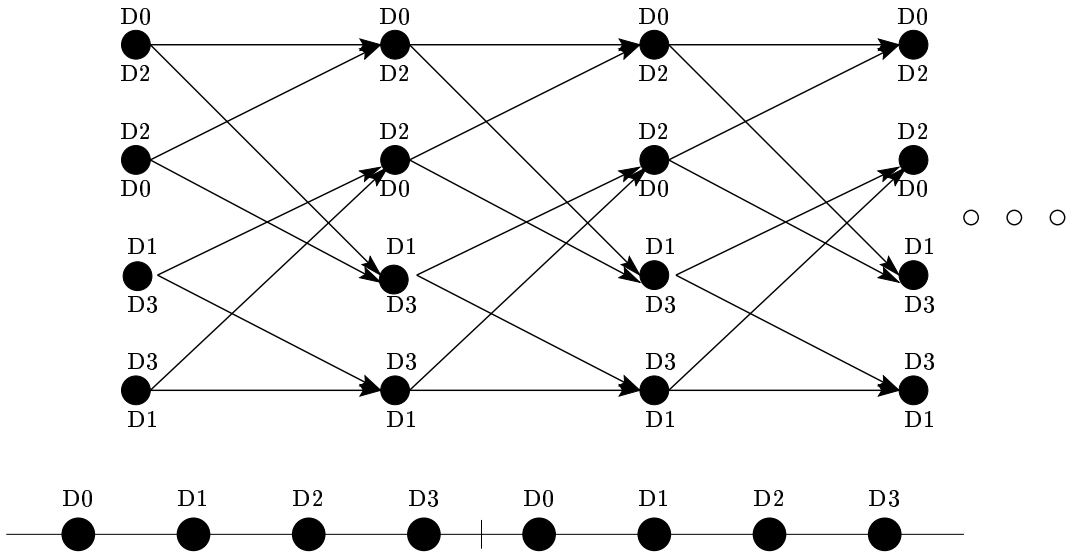


Figure 2.3: A 4-state trellis with subset labeling and codebook

Sequences of codewords that can be produced by the TCQ system are those that result from "walks" along the trellis from left to right. For example, if beginning in the top left state of the trellis in Figure 2.3, the first codeword must be chosen from either D_0 or D_2. If a codeword from D_2 is chosen, then we walk along the lower branch to the second state from the bottom, at which we must choose a codeword from either D_1 or D_3.

Given an input data sequence $x_1, x_2, \ldots$, the best (minimum MSE) allowable sequence of codewords is determined as follows. For the i^{th} stage in the trellis (corresponding to x_i), the best allowable codevector in the j^{th} subset ($j = 0, 1, 2, 3$), say c_j, is chosen and the associated cost $\rho_j = (x_i - c_j)^2$ is calculated. Each branch in the i^{th} stage of the trellis that is labeled with subset D_j is assigned the cost ρ_j. The Viterbi algorithm [14] is then used to find the minimum mean-squared error (MMSE) path through the trellis with the lowest overall cost. An entropy constraint is used in the design of codebook.

Codebook Design

The probability distribution of each normalized like-coefficient sequence can be modeled by the so-called *Generalized Gaussian Distribution* (GGD), whose probability density function (pdf) is given by

$$f_X(x) = \left[\frac{\alpha\eta(\alpha,\sigma)}{2\Gamma(1/\alpha)}\right] \exp\{-[\eta(\alpha,\sigma)|x|]^\alpha\} \tag{2.17}$$

where

$$\eta(\alpha,\sigma) \equiv \frac{1}{\sigma}\left[\frac{\Gamma(3/\alpha)}{\Gamma(1/\alpha)}\right]^{1/2} \tag{2.18}$$

and the gamma function $\Gamma(\cdot)$ is defined as

$$\Gamma(n) = \int_0^\infty e^{-x}x^{n-1}dx. \tag{2.19}$$

Distributions corresponding to $\alpha = 1$ and $\alpha = 2$ are Laplacian and Gaussian, respectively. The authors claim that from different codebooks optimized for various GGD, choosing $\alpha = 2$ and $\alpha = 0.75$ for the DC and non-DC sequences, respectively, results in excellent overall performance. Codebooks were designed for these distributions by employing a modified version of generalized Lloyd algorithm to minimize the cost function

$$J = E[\rho(x,c)] + \lambda E[l(c)]. \tag{2.20}$$

Here, x is the input data, c is the encoded version of x, $\rho(x,c)$ is the cost (MSE) of representing x by c, λ is a Lagrange multiplier, and $l(c) \approx -\log_2 P(x|A_i)$ is the number of bits used by the variable-length code to represent c. This algorithm chooses the "best" codeword by considering both MSE and the number of bits required to represent the particular codeword (entropy constraint). Training sequences for codebook design consisted of 100000 samples derived from generalized Gaussian pseudo random generators, each

tuned to the appropriate α value.

The algorithm implemented in this work (as a benchmark) differs from the algorithms presented by Abousleman and collaborators. Although it is mostly based on their scheme, it is slightly simplified in order to investigate the proposed concept rather than implementing very complex algorithms. The differences are described in the following subsection.

2.3.3 Encoding Algorithm as Used in This Work and Reported Results

The hyperspectral image is encoded in the following way, according to the Fig. 2.2. The first spectral band, x_1, is encoded and transmitted at a total rate of R_1 bits per pixel (bpp), as the initial condition for the DPCM loop. Each one of the following spectral bands, utilizing their predicted version, is used to produce the error images, which are encoded and transmitted at a total rate of R_s bpp. Note, that as more bands participate in the encoding process, the total bit rate of the hyperspectral image is more close to the asymptotic rate of R_s bpp.

Each one of the error images is partitioned to 8×8 blocks, and each block passes through the two-dimensional discrete cosine transform. The 64 transform coefficient sequences, which are combined from the like-coefficients of all the blocks in the image, are normalized and are passed to the trellis encoder.

Rate Allocation Algorithm

In this work we used a sub-optimal (but with much less complicated distortion model) rate allocation algorithm, described in [13]. It is a basic rate allocation algorithm widely used in various applications (for example sub-band coding) where there is a need to allocate a given amount of bits among the different sequences by satisfying the specified condition (for example, minimum MSE). The intuition of the technique implies to allocate more bits to the more "active" sequences in sense of some modeled criterion like variance of the sequence. We will use this algorithm in this research each time such a rate allocation problem arises. According to above, the variance of each non-normalized coefficient sequence is calculated and the MSE is modeled by

$$D(R) \approx \alpha \sigma^2 2^{-\beta R} \tag{2.21}$$

where D is the MSE, R is the encoding rate in bpp, σ^2 is the variance of the source, and α and β are parameters that depend on the particular encoder structure and the probability density function of the source. Let $M = 64$ be the number of coefficient sequences obtained by transforming a 8×8 block of error image data. The average distortion incurred by encoding the i^{th} coefficient sequence with R_i b/coefficient can be determined by (2.21). Since the DCT is a unitary transform, the overall average distortion introduced in the image by encoding all M coefficient sequences is given by

$$D = \frac{1}{M} \sum_{i=1}^{M} \alpha_i \sigma_i^2 2^{-\beta R_i} \tag{2.22}$$

The method of Lagrange multipliers can be used to determine the rates $R_i, i = 1, 2, \ldots, M$ that minimize (2.22), subject to an average rate constraint

$$R = \frac{1}{M} \sum_{i=1}^{M} R_i \tag{2.23}$$

Since this formulation does not guarantee that the resulting rate will be nonnegative, the rates, that are negative, are set to zero. Assuming that only the first M' coefficients have the nonzero rates, the resulting average distortion of the image is approximated by

$$D = \frac{1}{M} \left[\sum_{i=1}^{M'} \alpha_i \sigma_i^2 2^{-\beta R_i} + \sum_{i=M'+1}^{M'} \sigma_i^2 \right] \tag{2.24}$$

Using Lagrange multipliers to minimize (2.24) subject to (2.23) and arranging the coefficient sequence variances so that $\sigma_i^2 \geq \sigma_{i+1}^2$, yields the optimum rates

$$R_i = \begin{cases} \frac{1}{\beta} \log_2 \frac{\alpha_i \sigma_i^2}{\left(\prod_{j=1}^{M'} \alpha_j \sigma_j^2 \right)^{1/M'}} + \frac{RM}{M'} & i = 1, 2, \ldots, M' \\ 0 & i = M'+1, \ldots, M \end{cases} \tag{2.25}$$

The value of M' is chosen by setting $M' = M$ and computing the rates, $R_i, i = 1, 2, \ldots, M'$ from (2.25). If any rates are negative, M' is decreased by one and the rates are computed again. This process is repeated until all encoding rates are nonnegative.

The rate-distortion performance of the TCQ fixed-rate design (and not entropy constrained design), that was implemented in this work, was used to determine the parameters for the distortion model of (2.21). It was found that a value of $\beta = 1.88$ is appropriate for both DC and non-DC coefficients, while $\alpha = 1.54$ and $\alpha = 1.67$ should be chosen for the DC and non-DC coefficients, respectively.

After the rate allocation step in the algorithm, all of the M' nonzero rate normalized coefficient sequences are passed to TCQ encoder to be encoded at the appropriate bit-rate $R_i, i = 1, 2, \ldots, M'$.

For each rate R_i, the different trellis diagram for encoding/decoding was build. We concentrated on the full trellis diagrams with K states and L levels (or stages), which means that $q = K$ branches leave each state of the level j of trellis to each one of the states in the level $j + 1$. An example of full 4-state trellis diagram is shown in Fig. 2.4 below. The different trellis parameters are determined as follows. If an input vector of

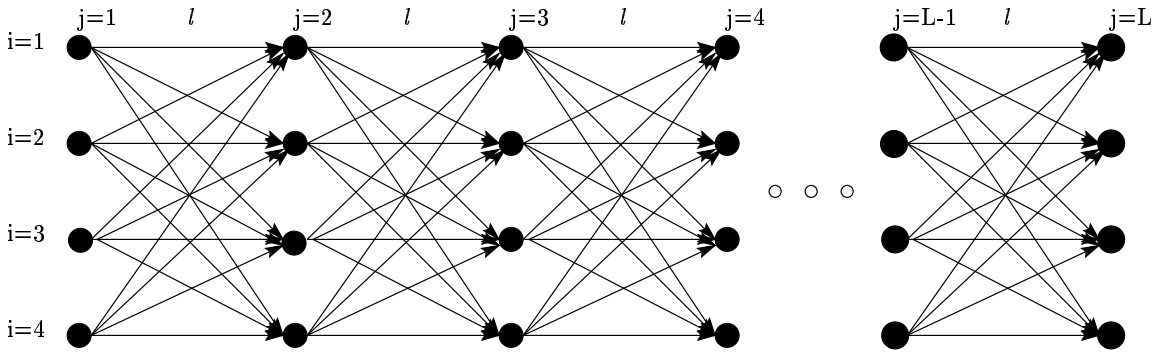


Figure 2.4: A 4-state full trellis example

length N have to be encoded by a K-state trellis encoder with q branches leaving each state at a rate of R bits/coefficient (bits per vector element), then

$$R = \frac{\log_2 q}{l} \quad \Longrightarrow \quad l = \left\lceil \frac{\log_2 q}{R} \right\rceil \tag{2.26}$$

and

$$L \cdot l = N \quad \Longrightarrow \quad L = \left\lceil \frac{N}{l} \right\rceil \tag{2.27}$$

where L is the number of levels (stages) of the trellis and l is the number of vector elements encoded by the single branch or the number of codevector elements populating each branch of the trellis.

The well-known Viterbi algorithm ([14, 16, 19, 20] etc.), is used for encoding the input vector with minimum MSE. The $\log_2 q$ bits for each stage of the trellis and the initial state information are transmitted from the encoder through the channel for the selected "best" path of each encoded coefficient sequence vector. This information, received at the

destination, is decoded in a similar way, inversely to encoding, by a TCQ decoder that reconstructs the coefficient sequences, and 2D-IDCT (inverse DCT) on the rearranged 8×8 blocks, producing the reconstructed error image. The DPCM loop finishes the reconstruction procedure by calculating the reconstructed band image, $\hat{x}$.

Random Populated Codebook Design

The last, but an important issue of the algorithm is how to populate the codebook with codevectors. Here, we used the generalized Gaussian distribution pseudo random generators to produce the long vectors of training data, tuned for appropriate value of the parameter α [1] in (2.17) and (2.18). First of all we concatenated the normalized coefficient sequences from all the bands of the particular hyperspectral image into long vectors (one for DC coefficients and one for all non-DC coefficients). Then, we tested two fitting techniques.

The first technique uses histogram presentation. The empirical probability distribution function (pdf) of the vectors was compared to the pdfs of the different GGDs with different values of α. The best value of α had to minimize the distance between the distributions, which was chosen to be the divergence of Kullback-Leibler, [17],

$$D(\underline{p}\|\underline{q}) = \sum_{i=1}^{n} p_i \log_2 \frac{p_i}{q_i} \tag{2.28}$$

where $\underline{p}$ and $\underline{q}$ are two random distribution functions (vectors of length n). The distribution $\underline{p}$ is a real data, while $\underline{q}$ is used as model estimation pdfs, with different parameters α, according to (2.17),(2.18) and (2.19).

The second technique is based on [36]. Here, given the real data vector X (DC or AC normalized coefficients), we determine estimate for the mean ($\hat{\mu}_X$) and variance ($\hat{\sigma}_X^2$) of the vector to be 0 and 1, respectively (due to the normalization). According to the following relationship

$$r(\alpha) = \frac{\hat{\sigma}_X^2}{\hat{E}^2[|X|]} = \frac{\Gamma(1/\alpha) \cdot \Gamma(3/\alpha)}{\Gamma^2(2/\alpha)} \tag{2.29}$$

where $\Gamma(\cdot)$ is defined in (2.19) and $\hat{E}[|X|] = \frac{1}{||X||} \sum_{i=1}^{||X||} |x_i - \hat{\mu}_X|$ is a modified mean of the absolute values of the vector, we can determine the best value of α by solving the equation

$$\alpha = r^{-1}\left(\frac{\hat{\sigma}_X^2}{\hat{E}^2[|X|]}\right) \tag{2.30}$$

[1] α is not the same parameter which appears in (2.21)

by using a lookup table.

The best fitting α values obtained from both techniques are quite close (about 0.6 for DC coefficients and about 0.9 for AC coefficients). After determining the optimal α values (one for DC and one for non-DC coefficients), the random vectors were generated by

$$\underline{x} = F_X^{-1}(\underline{u}) \tag{2.31}$$

where $\underline{x}$ has the "nearest" pdf to the real data distribution, $\underline{u}$ is independent uniformly distributed random variables and

$$F_T(t) = \int_{-\infty}^{t} f_T(t')dt' \tag{2.32}$$

is a cumulative distribution function of random variable T.

At the first step all the branches of each trellis for the different encoding rates were populated by this random population approach. At the next step the trellises were optimized by long training sequences derived from a random fitted GGD for finding the best representation codevectors in all of the possible paths. This was done utilizing a modified version of generalized Lloyd algorithm ([16]) for each trellis in the following way. In each iteration of the algorithm, firstly, all the training sequences are encoded using the existing trellis, and secondly, the trellis is updated by replacing each branch with the centroid calculated as a simple mean of all the sub-vectors being matched to this branch in the first step. The initial branches are populated from the random generated vectors with appropriate distributions and the iteration continues until the average distortion is below a specified threshold.

The above process achieves better performance with training sequences derived from the source data, but due to a very large number of training vectors needed to optimize trellis codebook (more than 100000 vectors), we derived training vectors from a fitted (with appropriate α) random generalized Gaussian distributions.

Reported Results

The authors performed coding simulations using a 140-band hyperspectral image obtained by the AVIRIS system, [21]. The bands were 256×256 pixels each and the performance of the coder is reported as the peak signal-to-noise ratio, PSNR,

$$PSNR = 10log_{10}\left(\frac{255^2}{(1/NM)\sum_{i=1}^{N}\sum_{j=1}^{M}(x_{ij} - \hat{x}_{ij})^2}\right) \tag{2.33}$$

where x and $\hat{x}$ are the original and distorted versions of the $N \times M$ image, respectively.

The first spectral band is encoded at $R_1 = 0.75$ bpp, and the asymptotic rate of rest of the images is $R_s = 0.1$ bpp (these encoding rates include the side-information, which remains constant regardless of sequence length). The hybrid system achieves a compression ration of 69:1 at an average PSNR of 40.29 dB for encoding of 40 bands. The same level of performance is attainable at a compression ratio of 76:1, if all 140 bands were encoded. Moreover, the authors claim that the subjective performance of the coder is excellent, with the reconstructed bands being virtually indistinguishable from the originals.

Such promising results, encouraged us to implement this algorithm as benchmark for the future research. But, the performance of the encoder highly depends on the image under the test, therefore, these very good reports must be dealt with a great caution.

Coding examples and overall performance for both benchmark coders are reported in Chapter 5 (subsection 5.3.1).

Chapter 3

Encoding of Hyperspectral Imagery Using 3D-DCT

This chapter presents the utilization of a Three-Dimensional Discrete Cosine Transform (3D-DCT) for encoding of hyperspectral imagery. Two different techniques for quantization of the transform coefficients that have not been used yet in the context of hyperspectral image compression are presented and investigated.

3.1 Introduction

It is obvious from the previous discussion that in order to achieve significant compression we must take advantage of the spatial and spectral redundancies inherently existing in hyperspectral imagery. The algorithms presented in the previous chapter utilize these redundancies as discussed while keeping relatively low computational complexity. If we want to overcome the limitations raised by the intermediate complexity requirement we may sacrifice the complexity in order to achieve deeper compression ratios (or better quality at the same compression ratio).

As mentioned above, the transform coding techniques are very popular and attractive due to their good energy compaction characteristics. The discrete cosine transform (DCT) is one of widely used transforms, and it is a building block of the JPEG algorithm which is a standard for still image compression. The main advantages of DCT: It is a real transform, it can be calculated efficiently, for example by the fast Fourier transform (FFT) algorithm, it asymptotically converges to KLT (which is optimal in transform

coding) and, unlike KLT, DCT is data independent. Moreover, the two-dimensional DCT (2D-DCT) used in JPEG, efficiently exploits the spatial redundancies in image domain by compacting the information to a small number of transform coefficients in the transform domain. For example, applying 2D-DCT on the homogeneous block in image domain will result in only one transform coefficient (also called DC coefficient).

In our application we would like to better exploit not only the spatial redundancies but also spectral ones, since the pixels in the same spatial location within the adjacent bands are not expected to vary much in their intensities. Therefore, if we stay with the transform coding techniques, the natural and straightforward extension of the 2D-DCT will be the three-dimensional DCT transform (3D-DCT).

Consider a basic equation of a 3D-DCT. The forward mathematical definition of the 3D discrete cosine transform is given by

$$\begin{aligned} I(u,v,w) &= \sqrt{\frac{8}{XYB}} E_u E_v E_w \sum_{x=0}^{X-1} \sum_{y=0}^{Y-1} \sum_{b=0}^{B-1} i(x,y,b) \\ & cos\left(\frac{\pi(2x+1)u}{2X}\right) cos\left(\frac{\pi(2y+1)v}{2Y}\right) cos\left(\frac{\pi(2b+1)w}{2B}\right) \end{aligned} \tag{3.1}$$

where

$$E_u = Ev = Ew = \begin{cases} \frac{1}{\sqrt{2}} & \text{u=v=w=0} \\ 1 & otherwise \end{cases}$$

and $i_{x,y,b}$ - is the 3D spatio-spectral intensity data element of the xth row, yth column and bth spectral band; $I_{u,v,w}$ - is the 3D transform domain data element at position u, v, w in the 3D transform space; X, Y, B are the dimensions of the hyperspectral data cube being transformed.

The inverse 3D-DCT (3D-IDCT) for the spatio-spectral data points is mathematically defined by

$$\begin{aligned} i(x,y,b) &= \sqrt{\frac{8}{XYB}} \sum_{u=0}^{X-1} \sum_{v=0}^{Y-1} \sum_{w=0}^{B-1} E_u E_v E_w I(u,v,w) \\ & cos\left(\frac{\pi(2x+1)u}{2X}\right) cos\left(\frac{\pi(2y+1)v}{2Y}\right) cos\left(\frac{\pi(2b+1)w}{2B}\right) \end{aligned} \tag{3.2}$$

The direct implementation of the 3D-DCT and 3D-IDCT is of high complexity. Fortunately, the DCT is a *separable* transform. This enabled to implement the 2D-DCT as first performing the 1D-DCT on rows of transformed block and then the 1D-DCT on columns of the resulting transformation. In a similar way, in the case of transforming

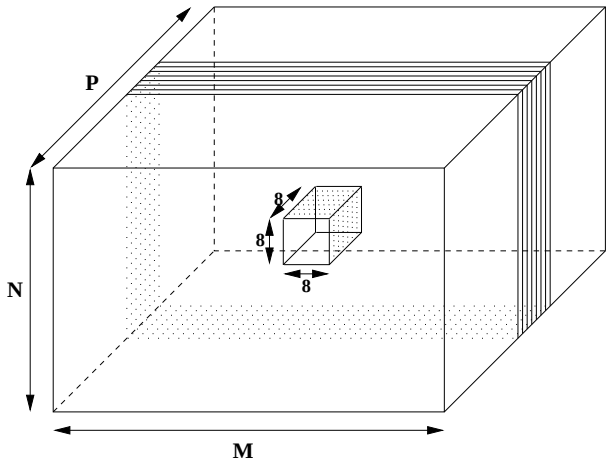


Figure 3.1: Partitioning of hyperspectral cube to 8x8x8 blocks

a three-dimensional blocks, we can perform first B 2D-DCTs of size $X \times Y$ in a spatial domain for each one of B bands as described above, and then XY 1D-DCTs of length B.

There is a tradeoff between the possible energy compaction (and therefore compression ratio or quality) and the size of 3D-blocks being transformed. If we transform a large 3D-block there is high probability that there exists a change in both spatial and spectral domains that will result in many transform coefficients with high amplitudes (bad energy compaction, which reduces compression ratio). On the other hand, transforming small 3D-blocks will certainly be applied to quite homogeneous spatio-spectral regions, but since the block is small the relative saving factor will be small too. These are the reasons for choosing intermediate 3D-block size of $8 \times 8 \times 8$ spatio-spectral pixels.

Thus X, Y, B in (3.1) and (3.2) turn to be equal to 8. The whole hyperspectral data cube of size $N \times M \times P$ have to be partitioned to 3D-blocks of size $8 \times 8 \times 8$ as shown in Figure 3.1. After applying the 3D-DCT on $8 \times 8 \times 8$ block we get 512 transform coefficient that can be regarded as the relative amount of the 3D-DCT spatio-spectral "frequencies" contained in the 512-point input signal. The coefficient with zero "frequency" in all the dimensions (i.e. $u = v = w = 0$) is called the DC coefficient and the remaining 511 coefficients are called the AC coefficients. Figure 3.2 depicts the relationship between the two domains. The DC coefficient is marked in the transform domain in black color, while the remaining 511 AC coefficient are white colored.

Assuming typical slow variations of sample values from point to point across the image and across the adjacent bands, after performing the 3D-DCT, data compression is achieved by utilizing the property that most of the 3D-block energy is concentrated

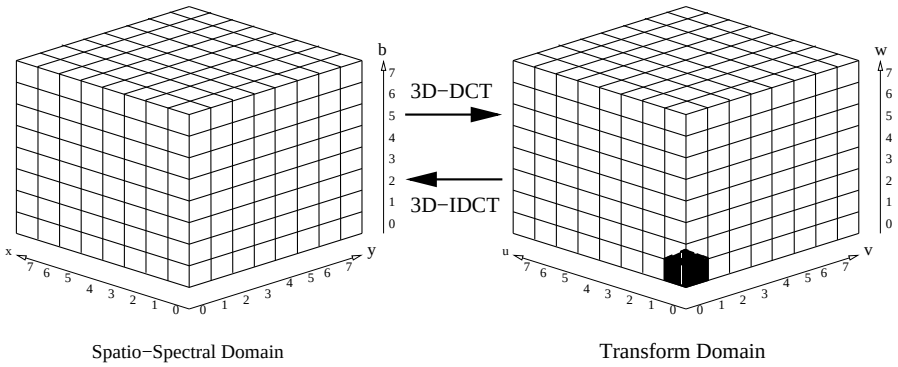


Figure 3.2: Spatio-Spectral and Transform Domains

in the low spatio-spectral "frequencies" (i.e. in the three dimensional corner near the DC coefficient). Most of the spatio-spectral "frequencies" will have zero or near-zero amplitude and need not be encoded.

In the following sections we will investigate two existing techniques for efficiently encoding the transform coefficients. The novelty of these approaches is implied from the fact that they have not been utilized on the hyperspectral imagery.

3.2 Quantization of 3D-DCT Coefficients Using 3D Quantization Table

As mentioned, Abousleman and collaborators showed in [3] that the performance of the encoder-decoder system based on applying 3D-DCT and further utilization of ECTCQ on the transform coefficients is comparable with the performance of the less complex hybrid DPCM/DCT system. Here, we concentrate on another technique for transform coefficient quantization, so-called *quantization table* technique.

In JPEG encoding algorithm the 64 transform coefficients (resulting from 2D-DCT performed on a 8x8 block) are quantized by a quantization table. This means that each coefficient is quantized by uniform quantizer with a quantization step specified in the quantization table, optimized for a large variety of images. The quantized coefficients are then zig-zag scanned and encoded by a lossless run-size Huffman table encoder. The purpose of the quantization and scanning is to reduce the energy of the transform coefficients and to order them in a sequence with as many running zero coefficients as possible in order to achieve better compression.

After transforming the hyperspectral cube by the 3D-DCT transform, we have a 8x8x8 mini-cubes with one DC and 511 AC transform coefficients that need to be quantized now with a three-dimensional quantization table, reordered and further losslessly encoded. The assumption leading to hope for efficient compression is based on the fact that the quantized coefficients will concentrate in a 3D corner near the DC coefficient and smart reordering will result in very long sequences of zero-valued AC coefficients, efficiently encoded by run-length encoder base on the Huffman tables.

We base the following sections on the algorithm presented by Lee and collaborators in [22], who proposed the technique for quantization of 3D-DCT coefficients and scan order for compression of video sequences. Here, we adjust their scheme for hyperspectral imagery.

3.2.1 AC Coefficients Statistics

Prior to applying the quantization table generation function proposed in [22] we had to be sure in the similar statistical behavior of the transform coefficients in both the video and the hyperspectral cases. The statistics of the AC coefficients, based on their dynamic range and empirical distribution in the video sequences, are modeled by the Laplacian (known as double-sided exponential) distribution. Consequently, we examined the dynamic range and the empirical distribution of the AC coefficients for the hyperspectral imagery. These parameters are presented for *"desert"*, *"glade"* and *"forest"* hyperspectral scenes in figures 3.3, 3.4 and 3.5, respectively.

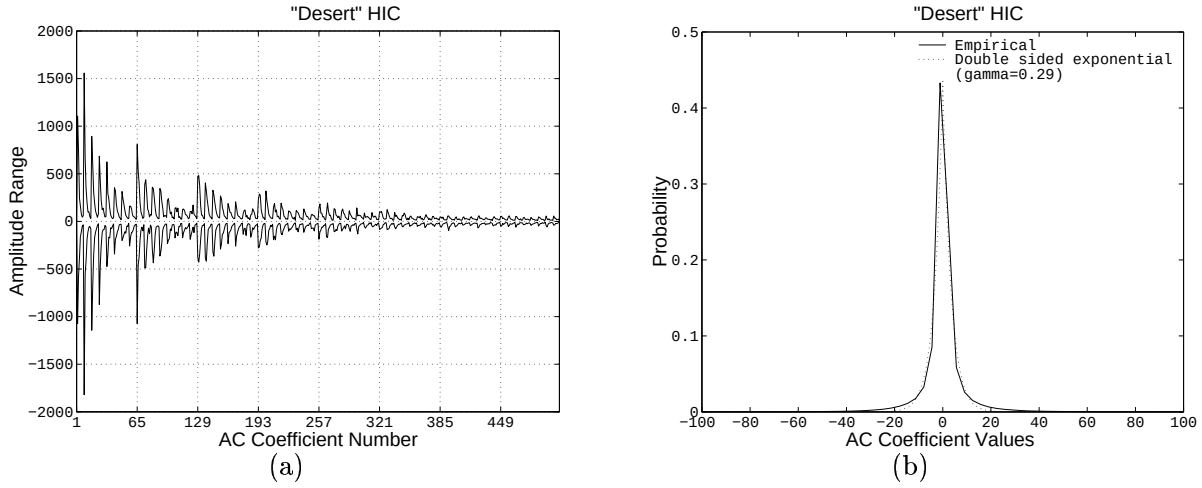


Figure 3.3: (a)-AC values dynamic range, (b)-AC coefficient distribution for "desert" cube.

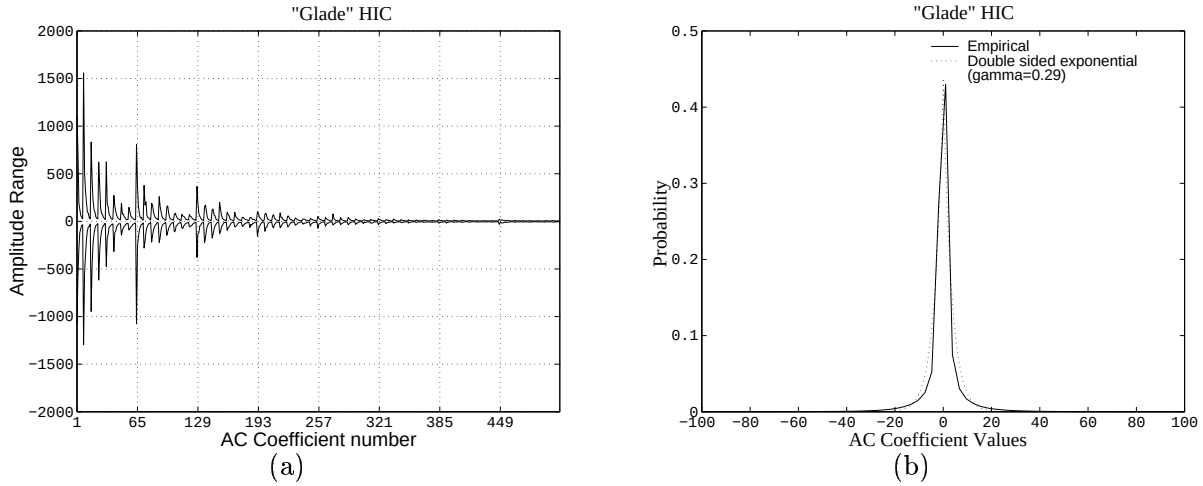


Figure 3.4: (a)-AC values dynamic range, (b)-AC coefficient distribution for "glade" cube.

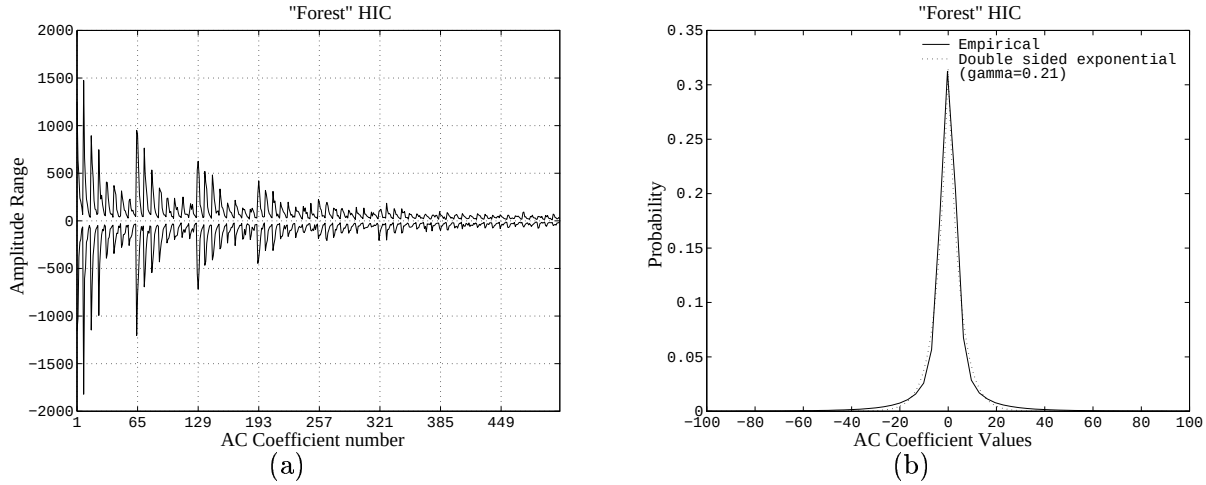


Figure 3.5: (a)-AC values dynamic range, (b)-AC coefficient distribution for "forest" cube.

The AC coefficients range from -1500 and +1500; most of them assume a small value, less than 100, and have a zero mean. In fact, many of the AC coefficients are close to zero. Thus compared with DC coefficients, AC coefficients contain little energy for the whole 3D-DCT cube. Periodic peaks are present for 511 coefficients for a 8x8x8 cube. The major period is 64, which corresponds to the position $u = v = 0$ for each plane of the cube. The minor period is 8, which corresponds to the dimension of the 3D-DCT cube in each axis. The figures show that more energy is concentrated on the major axes of the cube and the strength decreases as the locations concerned are further away from the DC location (0,0,0) along the u, v and w directions of the cube. Moreover, the distribution of 3D-DCT AC coefficients can be modeled by a general double sided exponential distribution

$$p(x) = \frac{\gamma}{2} e^{-\gamma|x|} \tag{3.3}$$

which is symmetric about zero, and can be adjusted to a particular hyperspectral sequence by choosing an appropriate parameter γ, as shown in the above figures. It can be seen that the empirical distribution closely follows the double sided exponential. The double sided exponential thus provides a method to model the distribution of 3D-DCT AC coefficients and can be used to limit the number of nonzero coefficients after quantization.

Since the empirical distribution and the dynamic range (the minimum and maximum values) of the AC coefficients in the case of hyperspectral imagery are very similar to those of the video we can apply the model proposed in the paper to our case.

3.2.2 The Design of the 3D Quantization Table

The quantization step plays an important role in the compression scheme since it reduces the magnitude of the coefficients, particularly those which contribute little or nothing to the quality of the image, and it produces more zero-valued coefficients. Our goal is to provide a 3D quantization table for the 3D-DCT coefficients as general as possible, according to the statistical behavior of the coefficients. It was observed that the dominant AC coefficients are spread along the major axes of the 8x8x8 coefficient cube, which typically contain a high percentage of the total energy. The magnitudes of the AC coefficients decrease exponentially as we move further away from the major axes. Therefore, the region of the coefficients cube, which captures such dominant coefficients corresponds to the *complement shifted hyperboloid*, as shown in Fig. **??**.

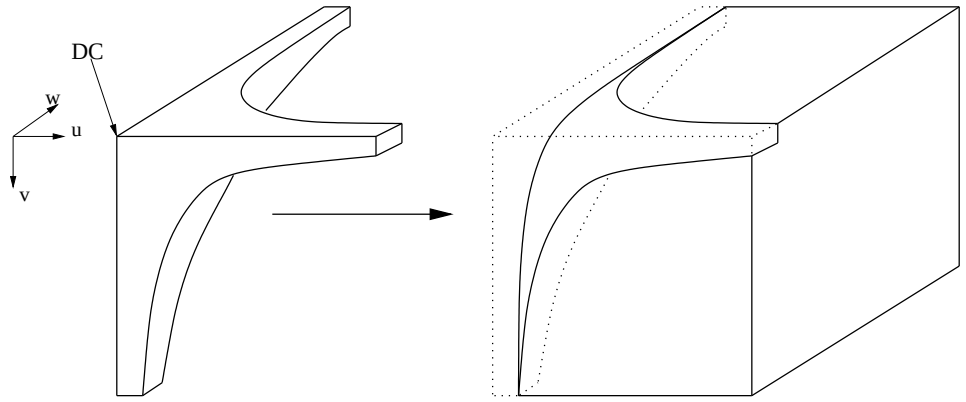


Figure 3.6:

-Ł3D-DCT . The 3D-DCT dominant coefficients are concentrated along the major axes, and are captured by a *shifted complement hyperboloid*.fig:fig8

In other words, as the *shifted hyperboloid* (on the right) defined by the curved surface

$$(u+1)(v+1)(w+1) > C \tag{3.4}$$

where C is a parameter, captures the less significant coefficients, the *shifted complement hyperboloid* (on the left) defined by the curved surface

$$(u+1)(v+1)(w+1) \leq C \tag{3.5}$$

is a complement of the shifted hyperboloid to the cube and aims to capture the most dominant coefficients. The size of the region for capturing the dominant coefficients can be varied by changing the parameter C. The shifted complement hyperboloid is also referred to as *inside region*, while the shifted hyperboloid is referred to as *outside region*.

The three-dimensional quantization table for a 8x8x8 coefficient cube should have an entry for each one of 512 coefficients, valued with a quantization step of the respective uniform quantizers. In the design of the quantization table we consider the following guide lines:

- The range of quantization values should be estimated based on the dynamic range of the coefficients.
- The quantization values should discard the high frequency components in the outside region.
- Since the dominant coefficients are spread along the major axes, the quantization values should have lower values along the major axes in order to preserve the high quality.

It follows from the above that the quantization values in the outside region have to be high tending to quantize the high frequency coefficients to zero. In the inside region, the quantization values have to be low thus achieving finer quantization and better quality. Considering this information and the observation of the exponential decay of the AC coefficients that can be modeled by a double sided exponential distribution, the following exponential function can be utilized to generate the quantization table:

$$q(u,v,w) = \begin{cases} A_{in}\left(1 - \frac{e^{-\beta_{in}(u+1)(v+1)(w+1)}}{e^{-\beta_{in}}}\right) + 1 & (u+1)(v+1)(w+1) \leq C \\ A_{out}\left(1 - e^{-\beta_{out}(u+1)(v+1)(w+1)}\right) & (u+1)(v+1)(w+1) > C \end{cases} \tag{3.6}$$

where (A_{in}, β_{in}) and (A_{out}, β_{out}) are parameters for the inside and outside regions, respectively, and $0 \leq u, v, w \leq 7$ for the 8x8x8 quantization table.

In order to build a generalized quantization table, we need to determine the values of the parameters $A_{in}, A_{out}, \beta_{in}, \beta_{out}$ and C for the specified type of imagery. Therefore, we optimize the above parameters for a set of the test hyperspectral images, while the appropriate tuning of the parameters results in the desired quality and compression ratio. The optimization procedure will be explained later in this section, after we present the scan order algorithm and the compression techniques of the quantized coefficients.

3.2.3 The Proposed Scan Order

Determining the scan order for the quantized 3D-DCT coefficients is an important issue, since in the case of quantization table the better compression is achieved by efficiently encoding the long sequences of zero-valued quantized coefficients. Consequently, the scan order algorithm should rearrange the 8x8x8 cube of quantized coefficients into a vector of length 512 while trying to collect as many consequtive zeros as possible.

The scan order proposed by the authors takes an advantage of the quantization values generated by (3.6). It is a reasonable assumption that the larger coefficients usually correspond to those dominant coefficients which are located in the inside region (the shifted complement hyperboloid). This implies that the coefficients from the inside region should be ordered before the coefficients from the outside region. The algorithm below generates the scan order using the quantization table values from (3.6):

Step 1. Set the parameter values $A_{in}, A_{out}, \beta_{in}, \beta_{out}, C$ in (3.6). These values are determined experimentally.

Step 2. Generate the coefficient numbers, defined by $w \cdot 64 + u \cdot 8 + v \quad (0 \leq u, v, w \leq 7)$, for an 8x8x8 cube and the corresponding quantization values by Eq. (3.6).

Step 3. Sort the resulting table with the quantization values from the inside region (whose coordinates satisfy $(u+1)(v+1)(w+1) \leq C$), as the primary key and the coefficient numbers as a secondary key (see Appendix B).

Step 4. Sort the resulting table with the quantization values from the inside region (whose coordinates satisfy $(u+1)(v+1)(w+1) > C$), as the primary key and the coefficient numbers as a secondary key (see Appendix B).

The column of the reordered coefficient numbers shows the scan order of the quantized coefficients. This ordering method gives a better result when compared with results from the 3D zigzag reported in [23] because it first orders the more dominant coefficients from the inside region, before those from the outside region. Thus, more coefficients that have been quantized to zero are packed together, resulting in an overall better compression ratio.

3.2.4 Lossy Compression

In the proposed scheme, two methods of compression are applied. Lossy compression, which is used in the quantization stage of the algorithm, and lossless compression. The implementation of the lossy compression is straightforward. Given a 8x8x8 cube of 3D-DCT coefficients, $I(u, v, w)$ and the adjusted 3D quantization table $q(u, v, w)$, the quantized coefficient cube $I'(u, v, w)$ is defined for each u, v and w by

$$I'(u, v, w) = round\left(\frac{I(u, v, w)}{q(u, v, w)}\right) \tag{3.7}$$

where *round* function rounds the result of division to the nearest integer.

While the division operation reduces the dynamic range of the coefficients which saves bits for encoding, the rounding operation introduces the distortion effect since the rounded number cannot be exactly reconstructed and some of the information is lost. The reconstructed coefficient values $\hat{I}(u, v, w)$ are defined for each u, v and w by

$$\hat{I}(u, v, w) = I'(u, v, w) \cdot q(u, v, w) \tag{3.8}$$

The quantization step regulates the compression ratio and the quality. If better compression is required the more coarse quantization must be performed, subsequently reducing the reconstruction quality. In the same manner, better quality implies fine quantization thus reducing the overall compression ratio. This is the only lossy part of the algorithm. After the quantization the coefficients are reordered according to scan order proposed and are forwarded to a lossless encoder for further compression.

3.2.5 Lossless Compression

After the quantization and reordering procedures described in the previous subsections, each cube of quantized coefficients is reordered into vector of length 512. The first element of the vector is a quantized DC coefficient and the rest 511 quantized coefficients are located afterwards while the dominant coefficients are closer to DC and many of the coefficients in the tail of the vector are zero valued.

The lossless encoding scheme is mainly based on the JPEG entropy coding technique. However, we made a slight changes for DC coefficient encoding and generated the new Huffman tables (for AC and DC cases) adjusted to the statistics of the hyperspectral imagery. More information about the Huffman tables can be found in [17]. The complete scheme is summarized in Figure **??**.

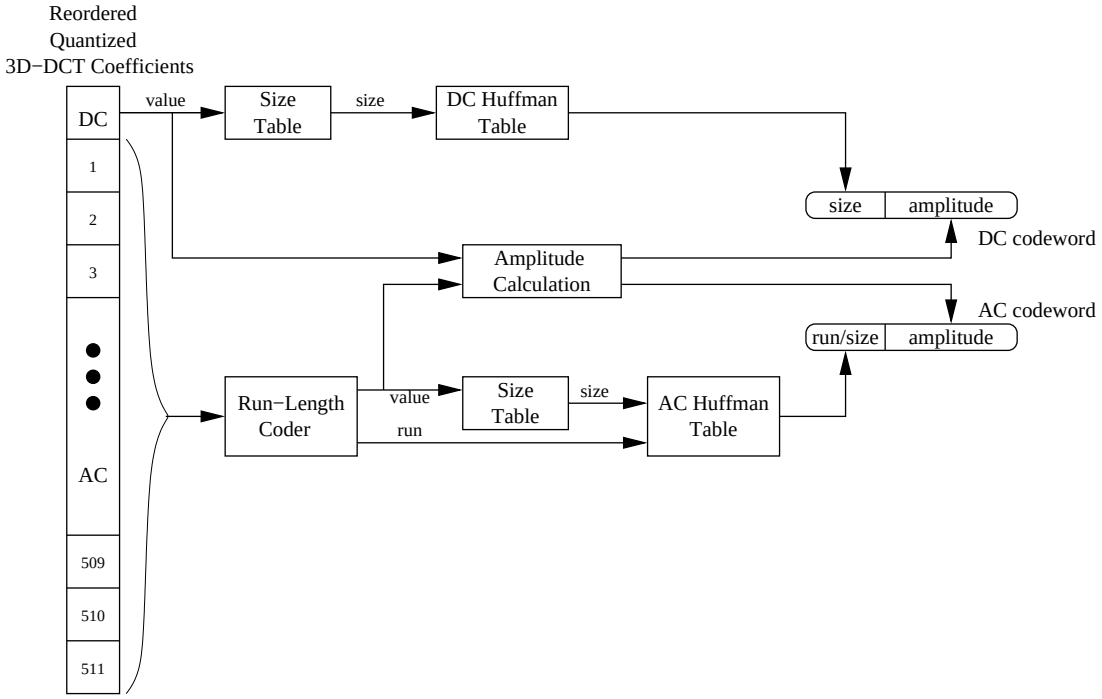


Figure 3.7: ŁHuffman

ŁHuffman Lossless encoding using Huffman tablesfig:fig9

Since the dynamic range is quite large in the case of 3D-DCT, unlike the case of 2D-DCT in JPEG, differential coding may not be appropriate for the DC coefficients. Instead, the DC coefficient of each vector is directly encoded by the size/amplitude pair. The *size* is the codeword from the DC Huffman table matching the appropriate range of amplitudes and the *amplitude* is a binary value of the actual amplitude value of the quantized DC coefficient. For example, if DC=13 is encoded, then from the lookup table 3.1 size is

determined to be 4. The amplitude is a binary value of 13 which is '1101'. Suppose Huffman code for symbol 'size=4' is '10', then the final codeword for DC=13 is (10,1101). For negative amplitudes one's compliment is performed. Thus, DC=-13 (the same size range, and therefore, Huffman table codeword) is represented by a code (10,0010).

The AC coefficients are encoded like in JPEG. The coefficients vector is first passed through run-length extractor that isolates groups of AC coefficients with a number of consequtive zeros preceding each coefficient (so-called *run*). Then each such a group is encoded by the run-size/amplitude pair. Here the Huffman codebook is a two-dimensional table, in which each entry with a codeword matches the appropriate event. The event is the mutual occurrence of the AC coefficient in some range of values (size) and a number of zeros preceding it (run), as depicted in Figure **??**. The run-size codeword is concatenated with the binary value of the amplitude of this quantized AC coefficient. It must be noted that the events with the biggest probability are located in the upper left corner of the table (not very large coefficient amplitude and a small number of zeros preceding such a coefficient). As we move to the lower right corner of the table (from the dotted line) the probability of such events (coefficient with large amplitudes and a large number of zeros) goes to zero. For instance, if we encode the event of 6 consecutive zeros (run=6) followed by the AC=-18, then according to table 3.1 size=5. Suppose Huffman code for symbol 'run/size=6/5' is '1110', thus the final codeword for this event is (1110,01101).

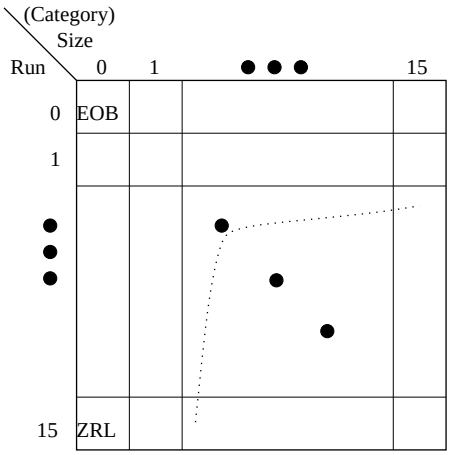


Figure 3.8: ŁHuffman ŁAC

ŁHuffman ŁAC Huffman table for encoding of AC coefficients.fig:fig10

There are two special events in the coding of AC coefficients, as follows:

1) The run-length value may be larger than 15. In that case, we use the symbol (15/0) to

denote a run-length of 15 zeros followed by a zero (signed as ZRL). Such symbols can be cascaded as needed, however the codeword for the last AC coefficient must have a nonzero amplitude.

2) If after a nonzero AC value all the remaining coefficients are zero, then the special symbol (0/0) denotes the end of block (signed as EOB).

The different amplitude ranges (sizes) are showed in Table 3.1. The same table is used to determine the sizes of both the DC and AC coefficients.

Size Range (Category)	Range of Values
0	0
1	-1,1
2	-3,-2,2,3
3	-7,...,-4,4,...,7
4	-15,...,-8,8,...,15
5	-31,...,-16,16,...,31
6	-63,...,-32,32,...,63
7	-127,...,-64,64,...,127
8	-255,...,-128,128,...,255
9	-511,...,-256,256,...,511
10	-1023,...,-512,512,...,1023
11	-2047,...,-1024,1024,...,2047
12	-4095,...,-2048,2048,...,4095
13	-8191,...,-4096,4096,...,8191
14	-16383,...,-8192,8192,...,16383
15	-32767,...,-16384,16384,...,32767

Table 3.1: Size Ranges (Categories) for lossless compression

It is important to note that since the DC coefficient is a sum of the gray levels of all the transformed pixels (512), it has always a positive value. In order to bring the DC value to required range of values, like in JPEG, a level shift is performed by subtracting from the 8x8x8 data cubes a uniform 128 valued cube in image domain or simply by subtracting a constant (equal to 2905, which is a DC coefficient of a 3D-DCT performed on a cube of 128s) from a DC coefficient (before the quantization) in the transform domain. The

decoder has to add the same constant when reconstructing the coded data.

The encoding here is lossless since the decoder can uniquely decode the bit stream it receives from the encoder through the lossless channel. For each received vector the first set of bits represent a size of the DC coefficient. Since *size* codewords are uniquely decoded, being a Huffman code, the decoder can now interpret the following number of bits, as pointed in *size*, as a binary value of the DC amplitude. A similar procedure is performed for the AC coefficients with the only difference that the decoding of run/size requires inserting a number of zeros into the reconstructed vector according to the matching entry in the Huffman table before the AC amplitude value.

3.2.6 Adjusting the Quantization Parameters

The adjusting of the quantization function parameters involves finding the appropriate values for the parameters $A_{in}, A_{out}, \beta_{in}, \beta_{out}$ and C in (3.6). Three hyperspectral images *"glade", "forest"* and *"desert"* were used for that purpose. To ensure the values in the quantization table would not require more than 8 bits for encoding, we fix the parameters A_{in} and A_{out} to 255, thus making the maximum value of in the 3D quantization table not greater than 255.

The parameter C is adjusted first to define the inside region (shifted complement hyperboloid) which captures the dominant coefficients. It was done for fixed $\beta_{in} = 0.00001$ and $\beta_{out} = 10$ to exclude the exponential decay effect. Since all the coefficients in the inside region are not quantized and all the coefficients in the outside region are discarded we determine this parameter according the acceptable visual quality and peak signal-to-noise ratio (PSNR) as defined in (2.33). In Figure 3.9 we show the average PSNR and compression ratio for three training images for different values of the parameter C.

$C = 10$ was chosen as a best fitting value since it gives a just-acceptable visual quality of the reconstructed images and quite a high average PSNR (about 32.2dB).

On the next step the parameter β_{in} was tested for fixed $C = 10$ and $\beta_{out} = 10$. Since β_{in} is responsible for the quantization steps in the inside region we can now use the compression ratio in addition to PSNR as a criterion. The compression ratio was calculated statistically in the following way. After quantization we perform lossless encoding utilizing DC and AC Huffman tables as explained earlier. The average bit-rate for a 8x8x8 block

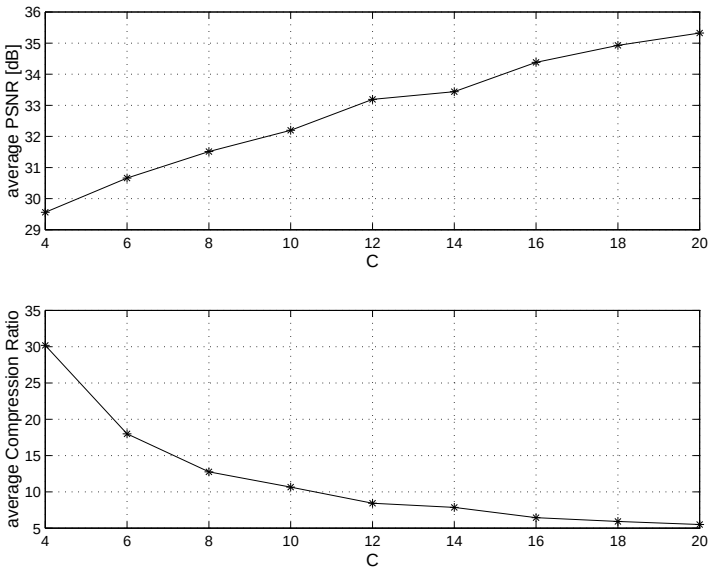


Figure 3.9: Average PSNR and compression ratio (CR) versus different values of the parameter C.

of pixels is $R = R_{DC} + R_{AC}$, while

$$R_{DC,AC} = \sum_{i \in T_{DC,AC}} P_{DC,AC}(i) \cdot [L_{DC,AC}(i) + SIZE_{DC,AC}(i)] \tag{3.9}$$

where $T_{DC,AC}$ symbolizes the DC and AC Huffman tables, respectively. For each table $P(i)$ is the empirical probability of each event, $L(i)$ is the number of bits in Huffman table's entry associated with each event and $SIZE(i)$ is the number of bits for binary representation for the amplitude of the appropriate quantized coefficient. The index i runs through all the possible events in each table. The compression ratio is therefore $CR = \frac{512 \cdot 8}{R}$.

Figure 3.10 shows the performance results for different values of the parameter β_{in} with fixed β_{out} and the fitted earlier $C = 10$.

$\beta_{in} = 0.15$ seems to achieve good performance with both a high average PSNR (29.5dB) and high compression ratio (about 80:1).

The last step is to adjust the parameter β_{out}, that controls the quantization steps in the outside region. For large values of β_{out} all the coefficients in the outside region are discarded. As β_{out} decreases more AC coefficients, less but still important are being treated thus reducing the CR but increasing the PSNR. The simulation results for this case are shown in Figure 3.11.

$\beta_{out} = 0.05$ brings us to PSNR=29.7dB with CR=75, so this value of β_{out} was chosen to be best fitting our data.

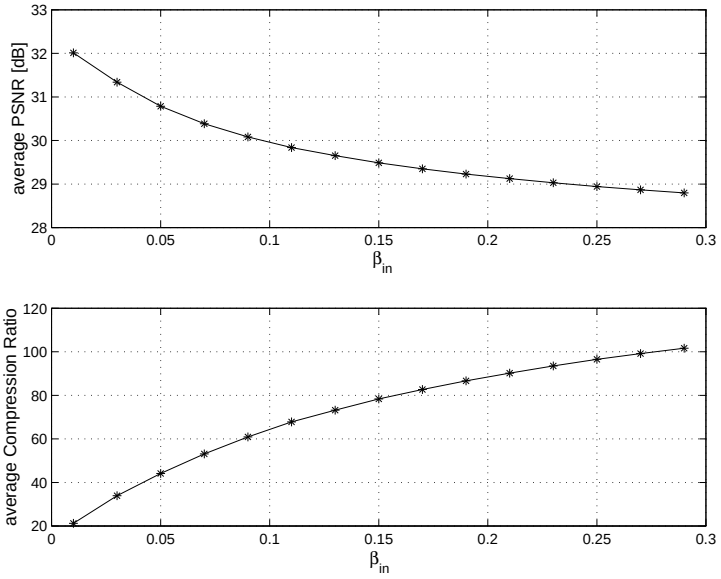


Figure 3.10: Average PSNR and CR versus different values of the parameter β_{in}.

It can be observed that determining the best fitting parameter values was done in quite a heuristic way. The parameters were chosen to give a good average overall performance (PSNR and CR). Of course, this choice of parameters can't be optimal for every set of images, but the main purpose of such an adjustment was to generate some standard 3D quantization table and scan order for the given set of hyperspectral images. Analogous to JPEG, we can better adjust the quantization table for a particular image by modifying the formulae (3.7) and (3.8) into

$$I'(u,v,w) = round\left(\frac{I(u,v,w)}{Q_f \cdot q(u,v,w)}\right) \tag{3.10}$$

and

$$\hat{I}(u,v,w) = I'(u,v,w) \cdot Q_f \cdot q(u,v,w) \tag{3.11}$$

where Q_f is a so-called *quality factor* that aims to regulate the compression ratio and the reconstructed image quality by increasing or decreasing the quantization step size for transform coefficients. The scan order nevertheless will remain unchanged since it was determined from the relative quantization step values.

3.2.7 Summary

We summarize this section by the complete encoding scheme for this algorithm, namely, quantization of 3D-DCT coefficients using a quantization table.

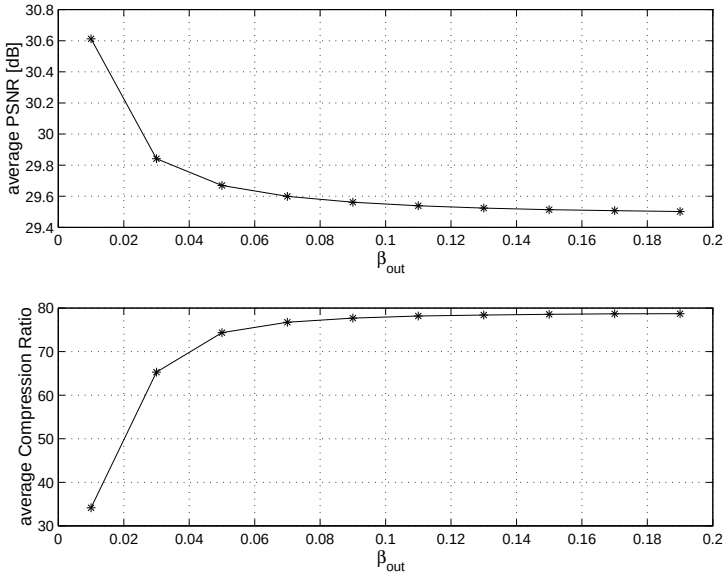


Figure 3.11: Average PSNR and CR versus different values of the parameter β_{out}.

As shown in Figure 3.12, every group of 8 adjacent input bands is partitioned into 8x8x8 cubes and transformed by a 3D-DCT. Then the encoder quantizes and reorders the quantized coefficients according to the predefined quantization table and scan order. The resulting vectors are further efficiently compressed by a lossless run-size Huffman tables encoder. Huffman tables can be predefined and used within an application as defaults, or computed specifically for a given image in an initial statistics-gathering pass prior to compression. Concerning our scheme, the Huffman tables are generated for each input image and are sent as a side-information. This side-information constitutes a negligible amount of bits (less than 0.1%) of the total amount of bits used for encoding.

Since the transmitted data is uniquely decodable, the decoder uncompresses the received sequences of bits into vectors. Then the 512 elements long vectors are back reordered into 8x8x8 cubes, multiplied by the quantization step and the reconstructed transform coefficient cubes are passed through 3D-IDCT thus resulting in the reconstructed hyperspectral image. Definitely, the same Huffman tables, scan order and quantization tables must be used in both the encoder and the decoder.

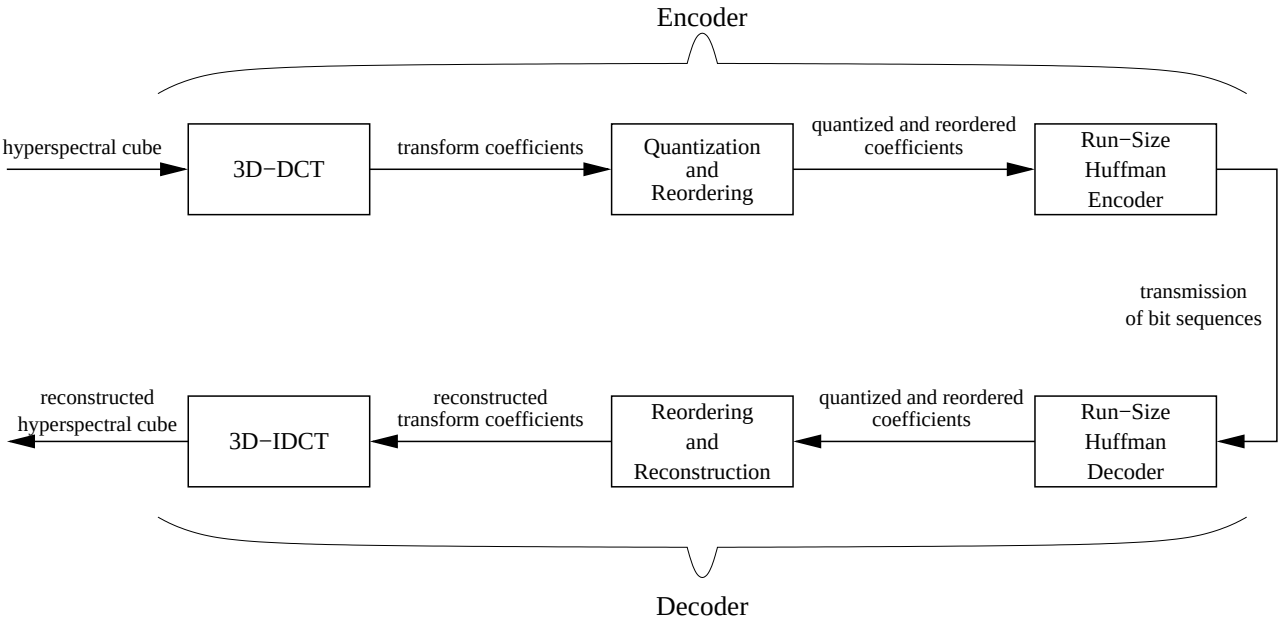


Figure 3.12: The complete scheme for QT approach.

3.3 Quantization of 3D-DCT Coefficients Using a Set of Vector Quantizers

After performing the 3D-DCT, one of the options for encoding the transform coefficients is by means of a quantization table with further entropy coding as discussed in the previous section. Another option is to use a trellis encoder as in [3]. Here we examine an additional way for encoding the transform coefficients. The proposed encoder quantizes the 3D-DCT coefficients using a set of *vector quantizers*. Vector quantization is a very popular technique (among block coding techniques) in image coding applications. Unfortunately, due to the exponentially growing computational complexity with enlarging block size, it is limited to using small blocks.

Here, the main idea is first to reshape the 8x8x8 block of transform coefficients into one long vector (of length 512) in some efficient way. Then, the reshaped vector is partitioned into sub-vectors and a bit allocation algorithm distributes the available number of bits along the sub-vectors according to their activity. Finally, the vector quantization codebooks are trained for each block (or sub-vector) of the vector, while more active blocks acquire richer codebooks according to the number of bits allocated to this block. Both the encoder and the decoder store this set of codebooks.

The efficiency of such a reshaping process improves if more high energy coefficients are

concentrated near the DC coefficient at the top of the vector and the low energy and near zero valued coefficients are grouped at the tail of the vector. We discuss in the following subsections the scan orders examined for efficient reshaping, the partitioning of the vector into the sub-vectors, the bit allocation algorithm and the training process of the set of codebooks for vector quantization.

3.3.1 Scan Orders

Two scan order techniques were examined. The first one is described in the previous section in context of the quantization table algorithm. Although it is based on the quantization table values generation, it performs a good reordering by concentrating the dominant coefficients (captured from the shifted complement hyperboloid) at the beginning of the vector near the DC coefficient and the rest low-energy coefficients at the end. We remind that the process starts with the generation of the quantization table values by (3.6) and proceeds with the sorting steps as pointed in subsection 3.2.3.

In JPEG the scan order used for better packing of the two-dimensional matrix of coefficients is a zigzag scanning. This method is based on the use of the isoplane, $u+v = C$, where C is a constant, covering the whole 2D matrix in a zigzag. In such a way, we first collect the DC and the dominant AC coefficients and the high frequency components tending to have near zero energy are concentrated in the groups at the end of the vector. The second scan order used is a direct extension of the 2D zigzag is a 3D zigzag scan using the isoplane $u + v + w = K$ where K is a constant. This idea was suggested by Yeo and Liu in [23].

The resulting vector of length 512 is passed further for quantization by a set of vector quantizers in either the first or second reordering scheme.

3.3.2 Partitioning, Bit Allocation and Design of Codebooks

Partitioning of coefficients vector

After the reordering process, a long 512-element vector of transformed coefficients is obtained. Our goal now is to partition it in some logical way into sub-vectors to be encoded by a set of vector quantizers. The reordering process assures the grouping of high-energy coefficients first and low-energy coefficients last. Therefore, it is reasonable to suppose that we partition the beginning of the vector into relatively short sub-vectors with quite

a large codebooks (due to high activity) and as we proceed to the end of the vector, the sub-vectors to be encoded will become longer, but with small codebooks.

From the observations of the reordered coefficient vectors and after testing a number of different partitionings it was decided to partition the vectors into eight sub-vectors containing 4, 4, 8, 16, 32, 64, 128, 256 reordered coefficients, as depicted in the Figure **??**. The last sub-vector consists of a near zero coefficients and will not require many codewords to be represented. Moreover, the coefficients grouped in it represent very high spatio-spectral "frequencies" which makes their contribution to the overall distortion negligible. That was the reason to group such a large number of coefficients together and this will increase much the compression ratio achieved.

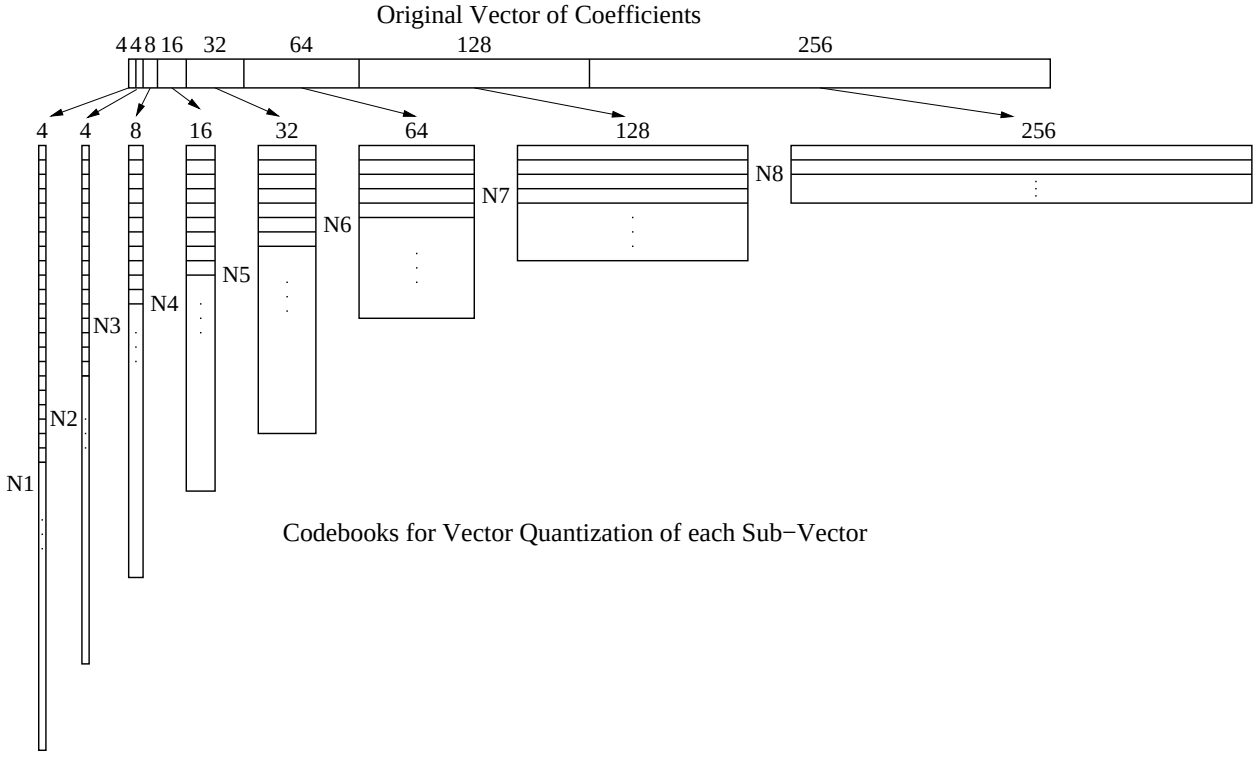


Figure 3.13: -

- ŁNi, $i = 1, .., 8$ Ł(3.14) -Ł(3.15). The partitioning of the original vector of coefficients into sub-vectors and the set of codebooks, where Ni, $i = 1, .., 8$ are as in (3.14) and (3.15).fig:fig11

Rate Allocation

Given the total number of bits for encoding a 8x8x8 spatio-spectral block of pixels, R (or compression ratio $CR = \frac{512 \cdot 8}{R}$), we need to spread these bits efficiently, after the packing of

the energy by means of transform coding (3D-DCT) and the reordering procedure. Thus, for a given compression ratio (or bit-rate) the codebooks for the different sub-vectors have to be trained while the number of vectors in each codebook is an exponential function of the number of bits allocated to the particular sub-vector. The more "active" sub-vectors should be allocated more bits and therefore more codevectors. The activity of the sub-vector is measured in terms of variance in the following way. Given V training vectors partitioned into the sub-vectors according to the previous discussion, we compute the variance of each coefficient in the vector $\sigma_j^2, j = 1, ..., 512$ by

$$\sigma_j^2 = \frac{1}{V}\sum_{k=1}^{V}(I_k^r - \mu_j)^2, \qquad \mu_j = \frac{1}{V}\sum_{k=1}^{V} I_k^r \tag{3.12}$$

where I_k^r is the reordered transform coefficient. Then, the sub-vector variance is calculated as the average variance of all coefficients belonging to this sub-vector:

$$\sigma_i^2 = \frac{1}{|M(i)|}\sum_{j\in M(i)}\sigma_j^2, \qquad M(i) \in \{\{4\},\{4\},\{8\},\{16\},\{32\},\{64\},\{128\},\{256\}\} \tag{3.13}$$

Now, given the total budget of bits R we allocate them to 8 sub-vectors R_i bits respectively according to the optimum bit allocation function from the rate-distortion theory which is given by

$$R_i = R_{av} + \frac{1}{2}log_2\frac{\sigma_i^2}{\left(\prod_{i=1}^{8}\sigma_i^2\right)^{\frac{1}{8}}} \tag{3.14}$$

where $R_{av} = \frac{1}{8}\sum_{i=1}^{8} R_i = \frac{1}{8}R$.

Fairly allocating the bits enables us to determine the number of codevectors that have to be trained for more accurate representation of the original vector of coefficients by the set of sub-codevectors. This number is set to be

$$N_i = 2^{R_i}, \qquad i = 1, ..., 8 \tag{3.15}$$

as shown in Figure **??**.

The Design of Codebooks

The codebook design is done according to the popular splitting method known as the LBG algorithm, which is based on the generalized Lloyd algorithm (GL) ([15, 16]). Let N be the desired size of the codebook (determined from rate allocation) and L be the length of the codevectors filling the codebook (which is also the length of the sub-vector to be encoded by this codebook). We start with generating n training vectors $\{\underline{\tilde{x}}_j\}_{j=1}^{n}$

from the reordered transform coefficients of the three training images *"glade", "forest"* and *"desert"*. The rule of thumb for a good quality codebook is that the number of the training vectors n is from $10N$ to $50N$. These are the steps of the LBG algorithm:

(0) Initialization: Set $M = 1$ and find the initial codebook $Y_0^M : \underline{y}_1 = cent\{\underline{\tilde{x}}_j\}_{j=1}^n$ where *cent* is a centroid of all the training vectors.
(1) Given $Y_0^M = \{\underline{y}_i\}_{i=1}^M$ split every codeword $\underline{y}_i$ into two codewords $\underline{y}_i \pm \underline{\delta}$ where $\underline{\delta}$ is a constant vector. We get as a result the codebook Y_0^{2M} with $2M$ codewords.
(2) Replace M with $2M$ ($M \leftarrow 2M$).
(3) Activate the GL algorithm for optimization of Y_0^M: $GL(Y_0^M) \rightarrow Y_0^M$.
(4) If $M = N$ stop. The codebook $Y = Y_0^M$ is the desired one. Else, return to (1).

The GL algorithm performs iteratively the optimization of a split codebook by matching the training vectors to the codevectors according to the "nearest neighbor" rule and then recalculates the centroids for each codevector until the distortion is less than a specified threshold.

Since the number of codevectors in each codebook N_i is a power of 2, the number of iterations of the LBG algorithm for each sub-vector is $log_2 N_i$ (while each iteration of the LBG algorithm may consist of many iterations of the GL algorithm).

Generally, the more bits assigned to the specific sub-vector, the more codevectors construct the codebook and the later encoding of this sub-vector will yield smaller distortion, but on the other hand the encoding involves more comparisons to be done, thus increasing computational complexity of the encoding.

3.3.3 Encoding and Decoding Summary

We summarize this section by describing the entire algorithm depicted in Figure **??**.

. The complete scheme for quantization by a set of vector quantizers approach.fig:fig12

The encoder receives the spectral bands of the hyperspectral image. Every group of 8 adjacent bands is partitioned into 8x8x8 cubes and transformed by a 3D-DCT. Then, the resulting transform coefficient cubes are reordered according to specified algorithm into long vectors and partitioned into 8 fixed size sub-vectors. Each sub-vector is encoded by

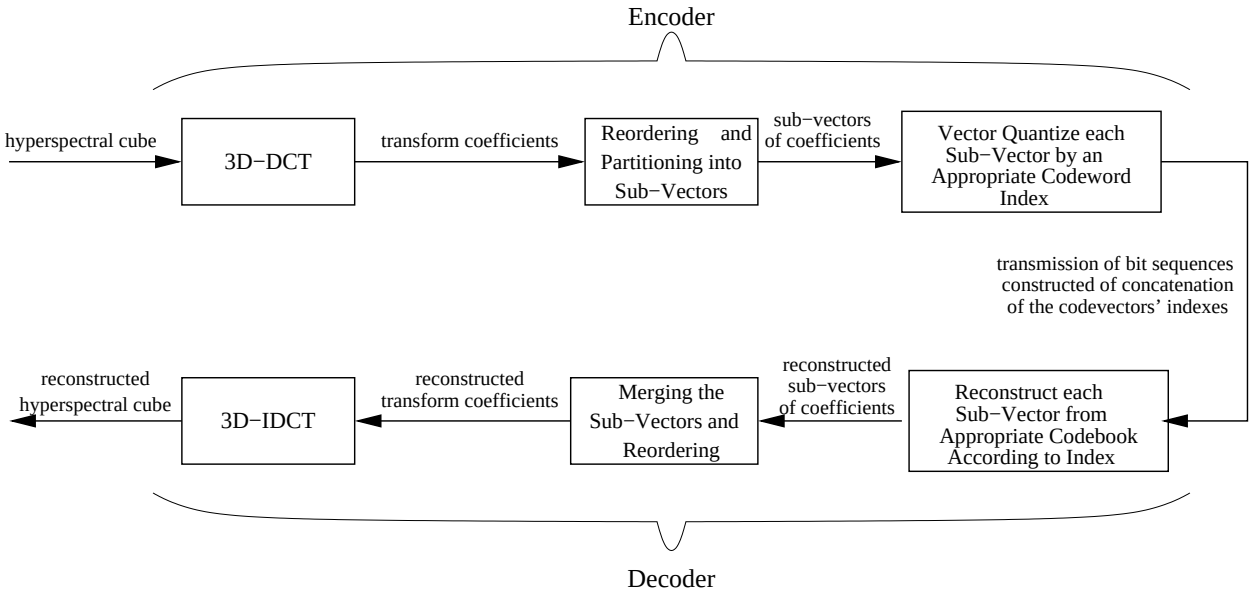


Figure 3.14:

an appropriate codebook (trained in advance) in terms of finding the index of the best matching codevector (Euclidean distance criteria). These indices (for all the sub-vectors) are concatenated and transmitted to the decoder or stored as the information about the particular spatio-spectral 8x8x8 block.

The decoder receives sequences of bits representing the indices of the codevectors in the respective codebooks for each 8x8x8 cube. It reconstructs the sub-vectors as the codevectors matching the received indices and merges the sub-vectors in a single 512 element vector of reconstructed transform coefficients. In order to return back from vector to cube, back reordering is performed with the same algorithm used in the encoder. Finally, the reconstructed hyperspectral image is produced by applying a 3D-IDCT on the reconstructed transform coefficients.

Coding examples and overall performance for both the QT and the VQ techniques are reported in Chapter 5 (subsection 5.3.2).

Chapter 4

Encoding of Hyperspectral Imagery Using 3D-Shape-Adaptive DCT

In this chapter we introduce a novel compression scheme for hyperspectral imagery. It is based on a *three-dimensional shape-adaptive DCT coding* technique. The detailed description of the proposed algorithm and the solution of associated problems (like side-information compression, determination of splitting criteria and devising a splitting algorithm) are discussed here.

4.1 Introduction and Motivation

In the previous chapter we showed techniques that can improve the algorithms presented recently in the literature. This was done applying some known compression techniques to an algorithm which uses a 3D-DCT approach.

Although the 3D-DCT approach has the very attractive feature of energy compaction and the encoding of the transformed blocks is usually efficient, it still fails in compacting the energy of high activity blocks. This is mainly due to the fact that high activity in a spatio-spectral block (i.e., high variations in pixel intensity values because, for example, edge in some area of the scene), results in relatively high AC coefficients even at high frequencies. These coefficients either decrease the compression ratio that can be achieved or, if we quantize them coarser, reduce the quality of the reconstructed bands.

Therefore, in order to improve the performance of the coding scheme we have to give a special treatment to those blocks. The main idea of the proposed scheme is to determine

in some way these high activity blocks, to encode the low activity blocks with a regular 3D-DCT scheme as explained in the previous section, and to encode in an efficient way the high activity blocks. Since these blocks have a major influence on the compression ratio and the quality of the reconstructed data, such a separation of the blocks into low and high activity blocks enables us to invest less bits in low activity blocks almost without impairing the quality, while high activity blocks deserve more bits for encoding due to their importance.

The first step in the algorithm is to determine the active blocks. Different criteria for splitting decision were investigated in this work. Then, based on the selected criterion, the appropriate blocks were split into two distinct regions using a vector quantization algorithm. A bitmap describing the partition was then generated. The two split regions were transformed with three-dimensional *shape-adaptive* DCT each, as will be explained later in this chapter. This technique proved to work well in cases of boundary block coding applications, and our application seems to be of that type. The encoding was performed applying both of the methods described in the previous chapter (quantization table and vector quantization of the transformed coefficients). The bitmap was losslessly encoded and transmitted as side-information to the receiver.

The proposed scheme tends to outperform the regular algorithm (which provides the same treatment for all the blocks in the hyperspectral image, regardless their activity). The main problem of the proposed scheme is the additional information that comes from two sources; the need to transmit the bitmap for correct reconstruction of the SA-DCT-transformed blocks, and the need to encode coefficients of the two transformed split regions. A good performance can be achieved only if we overcome this heavy overhead by a good choice of a splitting criterion and getting high compression ratios for unsplit low activity blocks.

The following sections explain in more detail all the issues raised here.

4.2 Splitting Criteria

As mentioned above, our first goal is to determine active blocks. High activity may come from two sources. Either the particular block has an edge (or a number of edges, which is less probable due to the small size of the block) or it constitutes a highly active texture. Since we are dealing with three-dimensional blocks, the variations in pixel values may

occur in all dimensions.

Generally, we are interested in blocks with sharp edges, since this is the case for which 3D-SA-DCT can outperform the regular 3D-DCT. That is why we tested a gradient criterion. But, if we can save bits or improve the performance of textured blocks also, we would like to split these blocks as well. Therefore, we also tested a variance criterion.

4.2.1 Variance Criterion

The main advantage of the variance criterion is its simplicity and low complexity which are very important factors in the algorithm implementation.

This criterion works as follows. For each input block of size 8x8x8 we calculate the variance of the pixels in the block by

$$\sigma^2 = \frac{1}{512}\sum_{k=1}^{512}(i_k - \mu)^2, \qquad \mu = \frac{1}{512}\sum_{k=1}^{512} i_k \tag{4.1}$$

where i_k are the intensity values for each of the 512 spatio-spectral pixels and μ is the mean intensity value in the 8x8x8 block.

Then we compare the variance of each tested block with a threshold value θ of the variance. Therefore, the splitting rule is: "split a block if its variance meets the requirement $\sigma^2 \geq \theta$, otherwise do not split the block."

The best value for θ could only be determined empirically as we do not have a model for the data. It is clear, that if we get an advantage from splitting the block, then there exists an optimum value of θ. Thus for small values of θ we split almost every block which is impractical and gives worse results because we invest bits in the wrong place and have a heavy overhead without improving the performance. On the other hand, for high values of θ we split a very small amount of blocks and therefore we return to the regular case of no splitting at all.

As already mentioned, the sharp edge usually results in high variance, but the high variance does not necessarily mean that we have an edge; it can be either an edge or coarse texture. Nevertheless, the shape-adaptive scheme is able to treat textures as well as edges, so that the variance criterion is a simple and convenient splitting criterion.

4.2.2 Gradient Criterion

This criterion takes into account an assumption of high spectral resolution. That is, since each 3D-block consists of only 8 adjacent spectral bands, large variations are not probable along the spectral axis. Therefore, high activity is expressed by spatial variations in each of the 8 spatial planes. Consequently, we can apply an algorithm for edge detection, like the one proposed by Canny [35], or a simpler technique like in [34], for determining the activity in each block. Both techniques, [35, 34], are based on gradient calculation, but since the latter one is much simpler and is applicable as a splitting criterion, we examined it.

The idea is to find the edges by calculating some type of spatial differentials. Since there may exist edges in four spatial directions (horizontal, vertical and two diagonal), we convolved each spatial plane with four types of masks as shown in Figure 4.1.

p1

1	0	−1
1	0	−1
1	0	−1

p2

1	1	1
0	0	0
−1	−1	−1

p3

1	1	0
1	0	−1
0	−1	−1

p4

0	−1	−1
1	0	−1
1	1	0

Figure 4.1: The four masks for determining local activity ([34]).

Since all elements are either equal to 0, +1 or -1, only additions are required for the calculation. Let $i(x, y)$ be the intensity value of a pixel at position (x, y), and let $p_k(i, j), \quad i, j = 1, 2, 3, \quad k = 1, 2, 3, 4$ be the elements of the mask operator p_k. The local activity, $A(x, y)$, at position (x, y) is then calculated by

$$A_k(x, y) = \sum_{i=1}^{3} \sum_{j=1}^{3} i(x - 2 + i, y - 2 + j) \cdot p_k(i, j) \tag{4.2}$$

and

$$A(x, y) = \max_{k=1,\ldots,4}(|A_k(x, y)|) \tag{4.3}$$

Now, if the value of $A(x, y)$ meets the requirement $A(x, y) \geq 128$ then the pixel at position (x,y) is defined as an edge pixel. Moreover, we say that the edge exists in the 8x8x8 block

if there are at least 4 edge pixels in the block, while they appear in at least 7 out of 8 spectral planes. The values of these parameters were determined empirically, so that the edges are sharp and continuous, and the number of edge blocks is of the same order as for variance criterion.

Nevertheless, the gradient criterion did not work well on our imagery (both using the above algorithm or the Canny edge detector). This may be mainly due to parameter sensitivity and to the fact that better performance can be achieved not only in blocks with edges, but also in textured blocks, that are not revealed by the gradient-based algorithms. Therefore, the variance criterion was used. The best value of the threshold parameter θ, was found experimentally.

4.3 The Splitting Algorithm by Vector Quantization

After determining a splitting criterion for high activity blocks we need to efficiently encode them. The high activity can result either from the presence of the edge in a 8x8x8 block or from the presence of coarse texture in this block. This means that if we could separate the highly variate block into a number of low activity regions with small variations in pixel values, we could efficiently compress this block by encoding a number of more homogeneous regions. In terms of transform coding this means that the 3D-DCT cannot compact efficiently the energy of the active block, i.e. many large amplitude AC coefficients exist. The *shape-adaptive* DCT can deal with this problem if the original block is partitioned into a number of low activity regions as explained later. The question is how many low activity regions are needed for achieving good compaction and how to perform such a partitioning in a successful way.

Since we encode the lower activity regions with 3D-SA-DCT choosing more than two regions for each block will increase drastically the information to be encoded (one SA block of transform coefficients for each region). Also the side-information increases since, for example, for three regions a ternary map have to be generated with additional 2 bits for each one of 64 pixels instead of 1 bit in the case of bitmap for two regions. Therefore, it was decided to partition the active blocks into only two lower activity regions.

Partitioning the block spatially in each one of spectral bands involves complex algorithms for edge detection or segmentation. Then, if we represent such a partition as two

regions separated by a contour in each plane, we have to encode the contour by a modified chain code which is at least 20 bits for each plane. This makes the side-information overhead unbearable.

Vector quantization (VQ) is a well known technique for coding. But, it can also be used as a clustering algorithm. Since we know that the hyperspectral imagery has a high spectral resolution, it is not likely to get high variations in data due to variations in intensity of a particular pixel through the 8 adjacent bands. On the other hand if two adjacent pixels are from different materials (have a different spectral signatures) they will have a different intensity values thus causing variations both in case of edge in the scene and in case of texture. Thus it is reasonable to decide that a pixel belongs to the one region or the other on the basis of the spectral data for this pixel in the specified block (i.e., a vector of length 8). This encourages the application of the VQ technique for clustering the 64 "spectral pixels" (64 vectors each of length 8) into two groups corresponding to different spectral signatures.

The VQ clustering proceeds using the LBG algorithm with the generalized Lloyd (GL) [15] algorithm already discussed in the previous chapter (with regard to codebooks generation). We briefly explain its implementation for the case of a 2-region clustering.

Given a 8x8x8 block of original data and assuming that according to the splitting criterion used it has to be partitioned into two lower activity regions, organize the 8x8x8 block as a set of 64 vectors (or "spectral pixels"), $\underline{x}_i$, each of length 8. In the first step an average of all vectors is calculated:

$$\underline{y}^0 = \frac{1}{64}\sum_{i=1}^{64}\underline{x}_i \quad , \tag{4.4}$$

which means that $\underline{y}^0$ is the centroid of the 64 input vectors. Then, this initial centroid is split into two new centroids by

$$\underline{y}_1^1 = \underline{y}^0 + \underline{\delta}; \qquad \underline{y}_2^1 = \underline{y}^0 - \underline{\delta} \tag{4.5}$$

where $\underline{\delta}$ is a small arbitrary constant vector.

The resulting centroids are then modified by the two following steps. The decision regions $\{S_j\}_{j=1}^2$ are determined by a "nearest neighbor condition" as follows

$$\{S_1\} = \{\underline{x} \mid d(\underline{x}, \underline{y}_1^1) < d(\underline{x}, \underline{y}_2^1)\} \tag{4.6}$$

where d is a squared-error distortion measure defined by

$$d(\underline{a}, \underline{b}) = ||\underline{a} - \underline{b}||^2 \tag{4.7}$$

and $\{S_2\}$ is the group of the vectors (out of all 64 vectors) that do not belong to $\{S_1\}$. After assigning each vector to one of two groups the centroids are recalculated by

$$\underline{\hat{y}}_i^1 = E\{\underline{x} \mid \underline{x} \in S_i\}, \qquad i = 1, 2 \quad , \tag{4.8}$$

(for the squared-error distortion measure) where E denotes expectation and it is calculated using a simple average of the vectors.

Finally, each one of 64 vectors is assigned to one of two groups, according to the minimum Euclidean distance from the two last centroids. This final partition provides us a 8x8 block segmentation bitmap in which entries corresponding to the spectral pixels belonging to one of groups are assigned '0' and the rest are assigned '1'. The concept of such a segmentation is illustrated in Figure **??**.

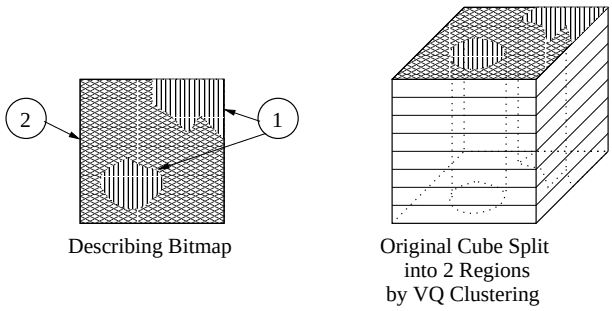


Figure 4.2: Ł8x8x8

Ł8x8x8 ŁVQ . Block of size 8x8x8 split by 2-class VQ clustering and its describing bitmap.fig:fig14a

The bitmap for each block to be split is forwarded to the 3D-SA-DCT and also to the lossless encoder in order to reduce the side-information (64 bits for each split block) to be transmitted to the receiver. This means that the side-information is only $\frac{1}{8}$ bits per pixel for the split blocks and can be further reduced by lossless compression as we show in the following section.

4.4 Lossless Coding of Side Information

Here we discuss different techniques that were examined for efficient bitmap coding for split blocks in order to reduce the side information overhead.

The input bitmap to be coded is a 8x8 binary image. Lossless compression is required since the decoder needs the exact partition map of "spectral pixels" in order to reconstruct

correctly the data and to perform the corresponding 3D-SA-IDCT for the two regions.

Many algorithms are available for encoding binary data [31]. The most popular and efficient algorithm for binary sequence encoding is *run-length* coding. We examined here a number of different techniques: Fixed run length coding as well as variable run length coding, followed by Huffman encoding tables, block-based Huffman coding, and Elias coding. Also, a number of different scanning techniques were used such as Hilbert scanning and "snake" scanning (an adaptive snake was also applied) in order to convert the 8x8 binary bitmaps to a 64 element vector of running zeros and ones. All of these are discussed here and the compression results are presented.

4.4.1 Space Filling Curves

There exist many techniques which reshape a 8x8 binary bitmap into a vector. The problem of good reshaping can be stated as follows: In order to achieve better compression, the reordering procedure has to group the binary elements of the bitmap into long sequences (or runs) of ones and zeros. With a reasonable assumption that the adjacent pixels are assigned to the same region we expect groups of ones or zeros in local areas of the bitmap. Thus the reshaping algorithm should walk through local areas in the 8x8 bitmap in order to obtain long runs of zeros and ones. The classical algorithm for such a walk is a Hilbert scan [25, 32]. As depicted in Figure 4.3 (left), this space filling curve crosses the whole 8x8 bitmap image while moving from one local area to another.

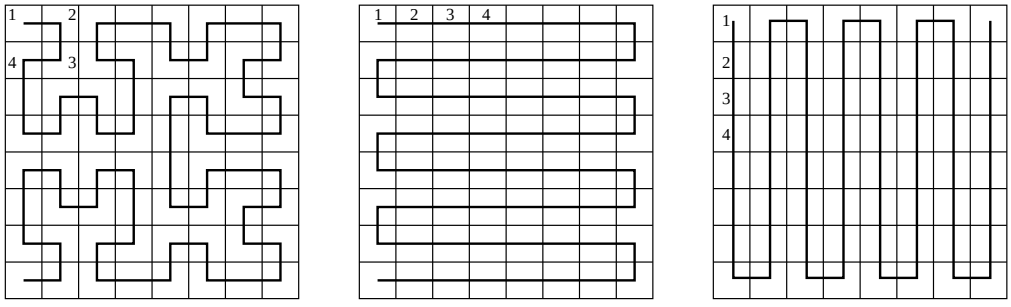


Figure 4.3: Hilbert, horizontal and vertical Snake curves.

Another such a walk is a so-called "snake" scan. It just crosses the whole bitmap in a snake's manner, while the snake can be performed horizontally (center image) or vertically (right image). Each one of the snakes can be preferable depending on the behavior of the particular bitmap. An adaptive snake was also examined. Adaptation is introduced by

choosing the best snake of the two (horizontal or vertical) for each 8x8 bitmap, while the best one achieves the minimum number of runs (long runs) of consecutive zeros or ones. The overhead for the adaptation is one additional bit for each bitmap that denotes which of the two possibilities was selected.

The reordering results are shown in Table 4.1 for the Hilbert, horizontal snake and adaptive snake reshaping algorithms. This experiment was carried out for about 2000 bitmaps of *"building"* image. The results show that an adaptive scheme works best in

scan type	zero/one runs	average zero/one run-length	runs per bitmap
Hilbert scan	9739/9658	6.75/6.54	9.6
Horizontal Snake scan	7377/7296	8.91/8.65	7.3
Vertical Snake scan	12348/12182	5.32/5.18	12.2
Adaptive Snake scan	6876/6857	9.56/9.21	6.8

Table 4.1: The number of zero/one runs; an average run-length and the number of runs per bitmap for Hilbert, horizontal and vertical snakes and adaptive snake reshaping algorithms.

the sense of creating the longest runs of consecutive zeros or ones. But, due to the 1 bit overhead in the adaptive scan, the simpler horizontal scan is a competitive candidate too. In addition, the final compression depends not only on the average zero/one run-length, but also on the further encoding algorithm that has to take an advantage of the long runs.

4.4.2 Bitmap Encoding Techniques

In this work we examined four different encoding techniques for compressing the binary bitmaps.

Block-Huffman Coding: Each 8x8 bitmap is partitioned into 2D blocks of P pixels each, which cover the entire image without overlap. Clearly we have 2^P different such blocks and we consider them as a source messages. Let p_i denote the probability (in practice, we measure frequency of occurrence) of the ith block. Then, the Pth-order joint entropy H_P (in bits per bitmap block) of the binary image source is equal to

$$H_P = -\sum_{i=1}^{2^P} p_i log_2 p_i \tag{4.9}$$

We can employ Huffman tables to code these 2^P different blocks (messages) at rate very close to H_P. Specifically, if R_{HUF} denotes the rate achieved by Huffman encoding of the blocks, then, according to Shannon's theory of discrete source coding, $H_P \leq R_{HUF} < H_P + 1$. All our results concerning block-Huffman coding refer to coding 2D blocks with sizes $P = 2 \times 2 = 4$ pixels (16 blocks per bitmap) and $P = 2 \times 4 = 8$ pixels (8 blocks per bitmap). For larger values of P, the number of Huffman codewords exceeds 256 and, hence, it becomes impractical for table lookup. The entropy for each case is $H_4 = 2.35$ and $H_8 = 4$ bits/block, and the respective Huffman tables achieve $R_{HUF4} = 2.45$ and $R_{HUF8} = 4.15$ bits per block. The average bit-rate for the bitmap is calculated by multiplying the average block bit-rate and the number of blocks for each case.

The following three techniques are for encoding the 64 element vectors of running zeros and ones, were also examined.

Elias Coding ([33]): This code is efficient for encoding long sequences of zeros separated by '1's or long sequences of ones separated by '0's. If we know all these run-lengths and separate them by some symbol marking the end of run, then we can exactly reconstruct the original sequence. Elias's idea was to code a run-length L of zeros, for instance, by representing it using m symbols of a m-ary arithmetic system and the $(m+1)$th symbol (referred as "comma") to represent the '1's separating the consecutive runs of zeros. Here, since we have both 'zero' and 'one' runs we modified this code by separating the runs of consecutive zeros or ones by '1' or '0' symbols. For $m = 3$, the run-lengths are represented in a ternary system and the Elias code requires $m + 1 = 4$ different symbols which can be obtained using a 2-bit code. Specifically, we chose the 4 required symbols "0, 1, 2, comma" to correspond to the 2-bit codes: 00="comma", 01="0", 10="1", 11="2". Consider, for example, the 40-bit binary sequence

0000000011111110000000001111000000011111

The sequence of run-lengths L is 8, 6, 8, 3, 6, 4 and the Elias code ($m = 3$) is

1111	00	1101	00	1111	00	1001	00	1101	00	1010
$2 \cdot 3^1 + 2 \cdot 3^0$	,	$2 \cdot 3^1 + 0 \cdot 3^0$	,	...	,	...	,	...	,	$1 \cdot 3^1 + 1 \cdot 3^0$

The coded sequence is shorter (34 bits) than the original, but the compression is not high. Actually, we found that it performs worse than the other examined coding techniques for our data. In fact, after Hilbert scan we have runs with about 6 bits length

('20' ternary) which is 4 bits/run. Since there are about 10 runs/bitmap (which means about 9 transitions (or commas) from zeros to ones and vice versa) we get a total number of $10 * 4 + 9 * 2 = 58$ bits/bitmap. Similarly, for the snake scan average runs are of length 8 bits ('22' ternary) or 4 bits/run. About 8 runs/bitmap require 7 commas and a total number of $8 * 4 + 7 * 2 = 46$ bits/bitmap. (Adaptive snake achieves only about $6 * 6 + 6 * 2 = 48$ bits/bitmap, while for representing the runs of length 9, 6 bits of code are needed).

Fixed-length Run-length Coding: This technique allocates a fixed number of bits for each run. Thus $log_2 L$ bits are required for encoding runs of length up to $L - 1$. Since the reordered sequence consists of the interchanging runs of zeros and ones, we need 1 bit for encoding the type of the first run (say, '0' for zeros run and '1' for ones run first). Then each run of length up to $L - 1$ is encoded by a fixed length code (which is simply a binary value of the run's length). If the run is longer than $L-1$, the "zero-state" symbol is inserted indicating that the next run is of the same type. The following example explains this concept. Consider the original 30-bit sequence

000000111111000000000111100000011111

If $L = 4$ (2-bit codewords, '00'-is a zero-state symbol) then the code

0 11 00 11 11 00 11 11 00 11 00 11 11 00 01 11 00 11 11 00 10

is a 41-bit sequence. If $L = 8$ (3-bit codewords, '000'-is a zero-state symbol) then the code

0 110 110 111 000 010 100 110 101

is a 25-bit sequence. Finally, $L = 16$ (4-bit codewords, '0000'-is a zero-state symbol) leaves us with a code

0 0110 0110 1001 0100 0110 0101

which is also of length 25 bit. The above example shows that it is possible to damage the performance by generating the code longer than the original sequence if we apply wrong length of codewords. Generally, the better codeword length should be about the average length of the runs. We examined this technique for both Hilbert and snake scans with 3 and 4 bit fixed-length codewords. The results are summarized in Table 4.2.

Run-length Huffman Coding: This technique is the most popular one and it achieves the best results among all the examined methods for the tested data set. First, it calculates the runs' statistics and then employs the Huffman coding algorithm (which is optimal among all variable-length codes) to encode the original sequences of bits. We trained here two different Huffman tables, one for zero runs and the other for one runs. The first bit of the encoded sequence denotes the type of the first run, while the rest of the code are uniquely decodable codewords representing the interchanging sequences of zero and one runs.

All the results are summarized in Table 4.2. As mentioned above, the preferred technique is the variable run-length Huffman coding method preceeded by a horizontal snake scan. It achieves an average compression ratio of bitmaps of 2 (32 bits of code instead of 64 original bits).

scan type	Elias code	fixed-length run-length code	run-length Huffman code	block-based Huffman code
Hilbert scan	58	45.6	37.8	-
Horizontal Snake scan	46	39.5 (3-bit/run) 36.6 (4-bit/run)	**32.1**	-
Adaptive Snake scan	48	37.4 (3-bit/run) 34.1 (4-bit/run)	34	-
- -	- -	- -	- -	39.2 (2×2) 33.2 (2×4)

Table 4.2: The bit-rate (in bits/bitmap) results for encoding 8x8 bitmaps with four different techniques.

4.5 Applying 3D Shape-Adaptive DCT on Split Cubes

The promising approach to boundary block coding is *shape-adaptive* DCT (SA-DCT) that was first proposed in [24]. This technique is generally used in image coding applications where the segmentation is applied for efficient encoding of boundary blocks (which are inherently highly active in our terms).

A three-dimensional SA-DCT used in this work is the extension of the 2D version

proposed in [24] and it can be applied on general three-dimensional shapes. However, in our case, because of the high spectral correlation and to reduce the side-information, the splitting of high activity cubes into two boundary regions is done by a vector quantization clustering algorithm. Thus, the classification into two classes implies uniform partitioning in the spectral axis. This means that the SA-DCT performed in the last (spectral) dimension reduces simply to a standard 1D-DCT of a fixed size (8).

A variable length DCT basis is used in SA-DCT. The length of the basis equals to the number of pixels involved in the transformation in each axis. A DCT transform matrix DCT_M,

$$DCT_M(p,k) = c_0 \cdot \cos\left[\frac{(2k+1)p\pi}{2M}\right], \qquad 0 \le p,k \le M-1 \tag{4.10}$$

containing a set of M basis vectors is generated, where

$$c_0 = \begin{cases} \frac{1}{\sqrt{2}} & \text{p=0} \\ 1 & otherwise \end{cases}$$

and p denotes the pth DCT basis vector. The M DCT_M coefficients $\underline{c}_j$ for each axis data $\underline{x}_j$ are calculated by

$$\underline{c}_j = \frac{2}{M} DCT_M \cdot \underline{x}_j \tag{4.11}$$

The DCT coefficients I(u,v,w) for all (u,v,w) in the block is calculated as explained above using successive 1D transformation for each axis. Figure **??** shows an example to emphasize the above process.

ŁDCT - . The 3D Shape-Adaptive DCT.fig:fig13

In this example we have a 3D cube of original data which was arbitrarily partitioned into two regions (only three 8x8 spectral bands are illustrated for the sake of simplicity). Let the pixels colored in black symbolize the region of interest needed to be transformed by a shape-adaptive DCT. We start the 3D transform with performing 1D DCT in row direction for each row in each plane or slice (also band). Thus as shown in the figure the first row in the upper slice contains only one black pixel and so it is transformed using DCT_1 basis vectors, while the fourth row is transformed using DCT_5 basis vectors. During this operation if the black pixels are contiguous in each row they are directly transformed, otherwise, the non-contiguous sequences of black pixels are merged into contiguous ones and transformed with 1D DCT of the appropriate length according to formulae (4.10) and (4.11). In both cases the resulting row transform coefficients are shifted towards the

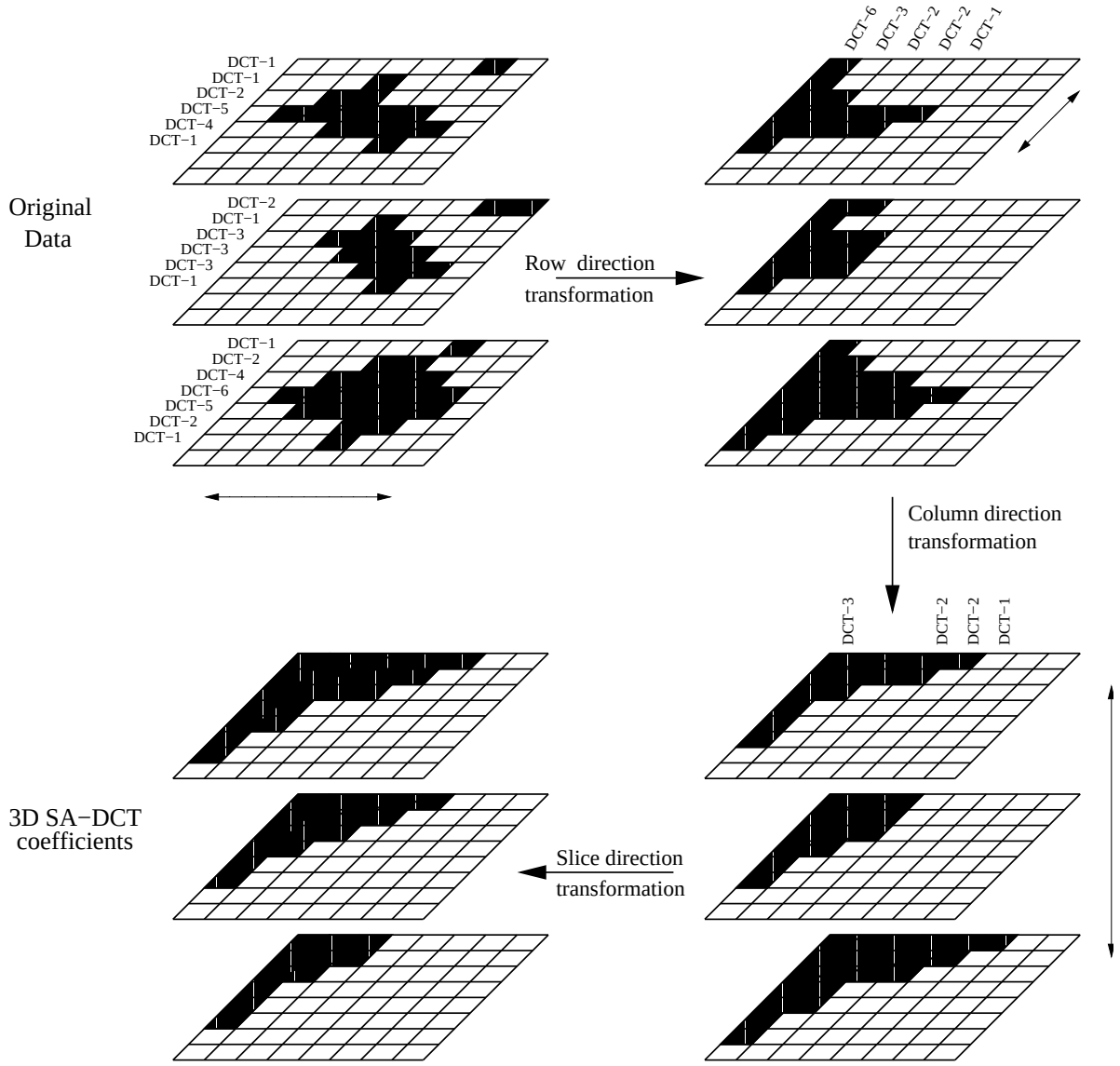


Figure 4.4: ŁDCT -

left border of each slice (see the upper right corner part of the figure). The transform in a row direction is also called *horizontal transform*.

The column direction transform (or *vertical transform*) is performed next. Now the 1D DCT is applied to the coefficients from the horizontal transform. Since the row transform coefficients were shifted to the left border of each slice, the vertical transform of the first column is a 1D DCT on all the row-DC coefficients, the vertical transform of the second column is a 1D DCT on all the first row-AC coefficients and so on. The M parameter of the 1D vertical transform depends on the number of the respective coefficients from the horizontal transform. For example, the row-DC coefficient was present in all 6 transformed rows in the first slice therefore a vertical DCT_6 is performed for the first column. The

discontiguous coefficients in each column are merged also here and the resulting transform coefficients (from formulae (4.10)and (4.11)) are shifted to the upper left corner of each slice as shown in lower right part of the illustration.

Finally, the *slice transform* is performed on the 2D-SA-DCT coefficients in the same way. The resulting 3D-SA-DCT transform coefficients are shifted towards the upper left three-dimensional corner as depicted in the lower left part of the illustration. Let us pay attention to the following important features of 3D-SA-DCT.

- The DC coefficient is always located at the upper left 3D corner of the transformed cube which corresponds to position $u = v = w = 0$ in the transform domain.

- Due to high spectral correlation and the side-information reduction we partitioned the high activity blocks by VQ resulting in 3D shapes as depicted in Figure **??**. By such a partitioning, only one bitmap for the whole 8x8x8 cube need to be losslessly encoded and transmitted as a side-information. We have to perform the 3D-SA-DCT twice; one time with the first region of interest (denoted 1) and one more time with the second region of interest (denoted 2).

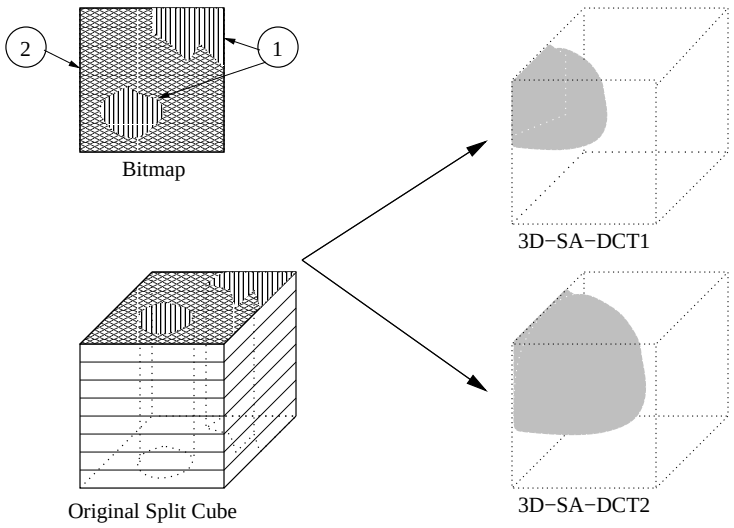


Figure 4.5: Ł3D-SA-DCTs Ł8x8x8

Ł3D-SA-DCTs Ł8x8x8. 3D-SA-DCTs of the split 8x8x8 block.fig:fig14

- The total number of transform coefficients remains 512 (as the number of pixels in the original cube) and they are concentrated in the top left 3D corner of each transformed cube (filled space in each cube in Figure **??**). The coefficients in the

non-filled space in both transform cubes are zero valued due to the special way we perform this transform (by shifting).

- Since the bitmap of the split block is transmitted to the receiver, the decoder can perform the 3D shape-adaptive inverse DCT in reverse order. The corresponding reconstruction formula is

$$\underline{x}_j = DCT_M^T \cdot \underline{c}_j \tag{4.12}$$

where DCT_M^T denotes an inverse one dimensional transform matrix (which is the transposed direct transform matrix defined in (4.10)). The output $\underline{x}_j$ are the appropriate (slice, vertical or horizontal) coefficients. The reverse coefficient shift operations are conducted, so that all the respective DCT coefficients are placed at the proper positions. We repeat this process in three different dimensions and the sum of the two reconstructed resulting cube provides a reconstructed data.

- A very important feature of SA-DCT is its ability to treat non-contiguous regions. This means that it can handle several non-connected regions, like in the illustration, or many regions as in textured blocks. Thus, this type of transform is capable of concentrating the energy of the data in the two 3D-corners of the transform cubes of the corresponding regions.

- It should be noted that obtaining good coefficient energy compaction, highly depends on the partitioned regions nature. An efficient representation of the original 3D signal is obtained if the vertical SA-DCT is performed along horizontal SA-DCT coefficients that are highly correlated and the slice direction transform is performed on the highly correlated vertical SA-DCT coefficients. While this assumption is usually correct for slice transformation (due to high spectral resolution), it is not always true for horizontal and vertical SA-DCTs. This is mainly due to the fact that non-contiguous partitions may damage the spatial correlation between the adjacent pixels thus reducing the encoder's performance.

Summarizing this section, the attractive features of the shape-adaptive DCT and the relevance of its three-dimensional extension to efficient encoding of high activity 8x8x8 blocks encouraged us to apply this technique. Moreover, 3D-SA-DCT enables us to take an advantage not only of high activities resulting from edges, but also of coarse texture

regions, due to the transform's ability to deal with the arbitrary shapes of the encoded region.

4.6 Proposed Coding Scheme

In this section we summarize the proposed encoding and decoding procedures that a hyperspectral image passes through. The heart of this scheme relies on the encoders/decoders introduced in the previous chapter. Namely, the quantization table (QT) approach and the VQ approach utilizing a set of vector quantizers. The full scheme is presented in Figure 4.6.

4.6.1 Encoding Procedure

The encoder is illustrated in the upper part of Figure 4.6.

The preprocessing block aims to partition the whole hyperspectral image into blocks (or cubes) of size 8x8x8. It calculates the activity of each block by measuring its variance. One bit for each block is transmitted as side-information ('1' for the high activity blocks and '0' for low activity blocks).

According to splitting decision each block passes through the proper track in the scheme: It can be encoded by a regular 3D-DCT approach, if it is of low activity or, if it has high activity, it is split into 2 lower activity regions and encoded using the 3D-SA-DCT approach.

The unsplit blocks are transformed by a 3D-DCT described in the previous chapter; the resulting transform coefficients are encoded by one of the two techniques (quantization table or a set of vector quantizers). It is important to mention, that since these blocks are of lower activity, all the parameters of the encoders must be recalculated for this type of data (parameters of the quantization table and a set of vector codebooks). This adjustment is required for achieving better performance.

With regard to the blocks to be split, they are passed through the VQ clustering algorithm that partitions them into two lower activity regions. The output of the splitting algorithm is a bitmap, according to which the 3D-SA-DCT is performed.

Since 64 bit bitmaps are quite a heavy overhead for each split block, we compress the bitmaps losslessly and the compressed 32 bits (on average) representing each bitmap are

transmitted as side-information too.

The split regions are transformed utilizing 3D-SA-DCT (two transforms per original block) and the transform coefficients are quantized and encoded by one of the two mentioned techniques (QT or VQ). Also this type of data needs the tuning of the corresponding quantization tables and a set of vector quantizers.

The adjustment of the quantization parameters is performed off-line using a set of training hyperspectral images to determine the best parameters. However, the encoder could also generate the information for each input hyperspectral image (like Huffman tables for lossless encoding) and this information has to be transmitted to the receiver. Although there are many types of side-information in the proposed scheme, the significant one is that of transmitting the bitmaps. Huffman tables need to be transmitted only once for each encoded hyperspectral image cube and they require negligible amount of bits compared to the compressed data amount.

4.6.2 Decoding Procedure

The decoder is illustrated in the lower part of the Figure 4.6. Its purpose is to follow the steps of encoding, in reverse order, to reconstruct the compressed data using also the received side-information.

If the quantization table (QT) approach is used, the data arriving from the encoder is reconstructed employing the Huffman tables for each type of blocks (split or unsplit). If a set of vector quantizers was used for encoding, the same set is available in the decoder and the data is reconstructed employing the appropriate sets of vector quantizers for both types of blocks. An additional bit per block is transmitted to provide the decoder information about the type of the decoded block. If a split block is to be decoded the decoder knows that it has to receive a double amount of information (for the two compressed regions) before it deals with the next 8x8x8 cube.

The low activity block's reconstructed transform coefficients are transformed back by applying the inverse three-dimensional discrete cosine transform (3D-IDCT). The reconstructed coefficients of the two regions of the high activity blocks are transformed back using a 3D-SA-IDCT for each one, relying on the exactly reconstructed bitmaps, and are summed into a single 8x8x8 cube.

The reconstructed hyperspectral image consists of the reconstructed cubes placed each

in an appropriate spatio-spectral location.

Coding examples and overall performance for the shape-adaptive approach are reported in Chapter 5 (subsection 5.3.3).

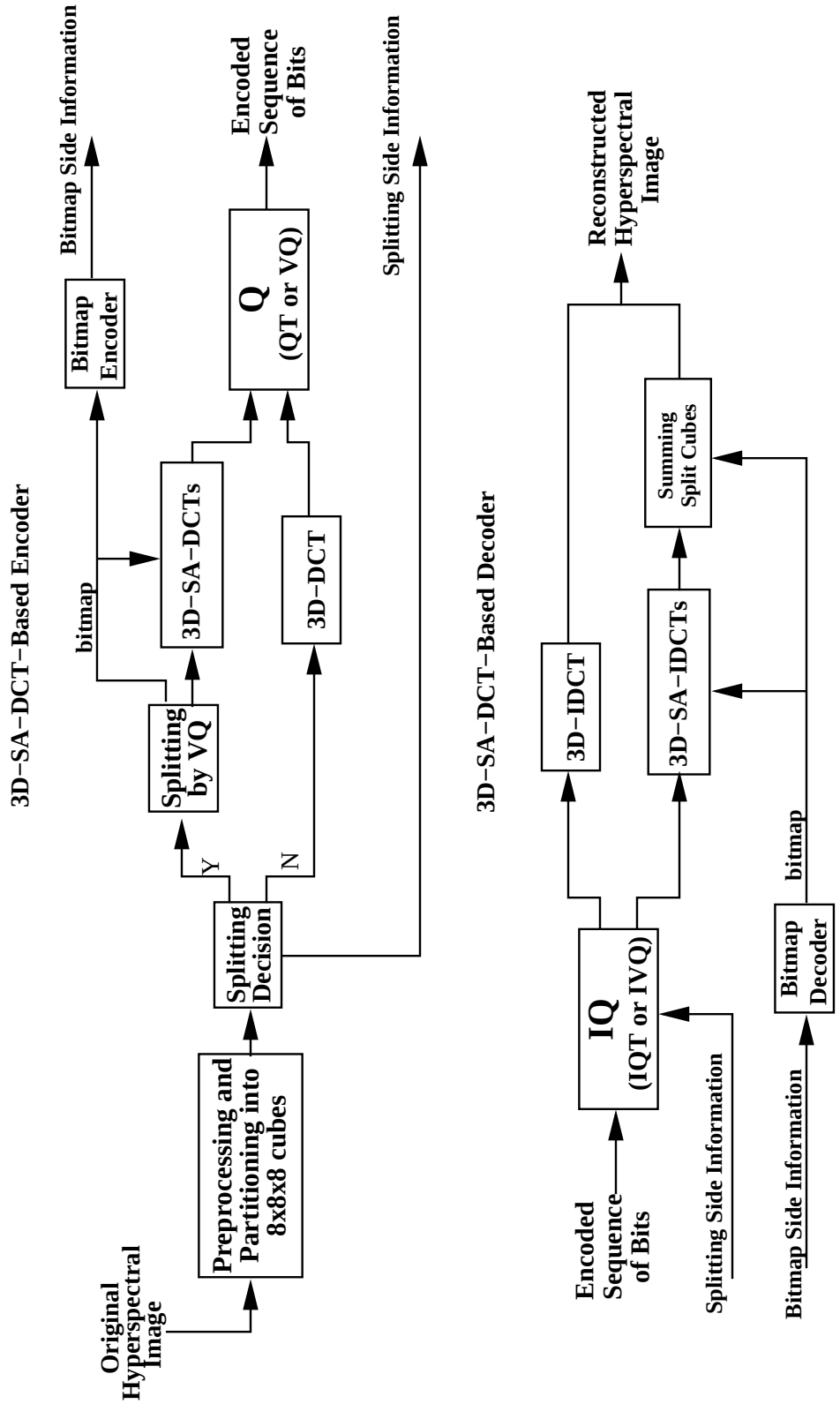


Figure 4.6: The Encoder-Decoder SA-based scheme.

Chapter 5

Implementation and Simulation Results

In this chapter we discuss implementation issues and present simulation results obtained with the described algorithms when applied on the hyperspectral imagery. We also discuss advantages and disadvantages of the different methods.

5.1 Introduction

The software implementation of all the algorithms was written in a *Matlab*© environment using *Matlab*© software. Although it is common to implement image compression algorithms in C language, we preferred using *Matlab*© software. This is mostly because of its built-in mathematical functions, that are difficult to implement in C (like *eig* function for calculation of eigen values and vectors of the matrix, *reshape* function and so on) and the convenient graphics. The main price for this convenience was the long run time for encoding the images and especially for the design of codebooks.

In order to compare coding performance of different algorithms, performance measures must be defined. In addition to standard performance measures, we also tested an application which this kind of imagery is destined for. A simple classification algorithm was tested on the original and the reconstructed data sets and the classification errors of the different coding techniques were compared. Moreover, fair comparison demands testing different techniques on the same input data.

Four HICs were used in our research. While the *"building"* image was utilized as

test data, we employed the remaining three image sets, *"desert"*, *"glade"* and *"forest"*, as training data for the different algorithms, to generate VQ codebooks, quantization tables, Huffman tables and so on. All of the scenes consist of 120 spectral bands covering the visible and near-infrared spectral window (wavelengths from 400[nm] to 1000[nm]). Band 60 of each scene is shown in Figure 5.1. The scenes are of different spatial sizes - 200x200, 200x280, 256x256 and 128x256 for "glade", "forest", "desert" and "building" images, respectively. Each pixel in each band has 8 bits of radiometric information.

5.2 Quality Measurement Definitions

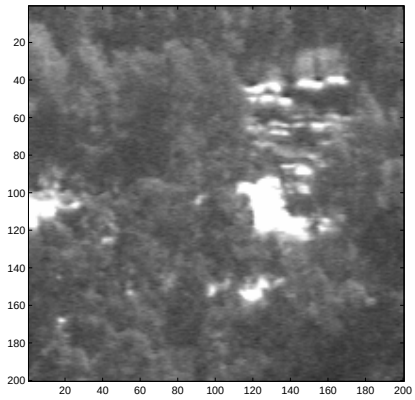
There exist different performance measures for verification of coding algorithms. In order to make a fair comparison between the techniques, the same performance measure must be used, preferably on the same hyperspectral data. It is known that this type of imagery is not necessarily viewed by a human visual system (HVS). Although the reconstructed cubes were examined also by a subjective quality criterion (visual quality, artifacts like blockiness, smoothness etc.), it is obvious that the true quality can be measured mainly according to the specific application the encoding is used for. The best example for hyperspectral imagery application, is the classification and material detection from the input data. Here, in order to check if at the given compression rate the quality of the reconstructed cube is good enough, the classification based on the original and reconstructed data must be performed and compared.

In this research it was decided to measure the performance with the following two performance measures:

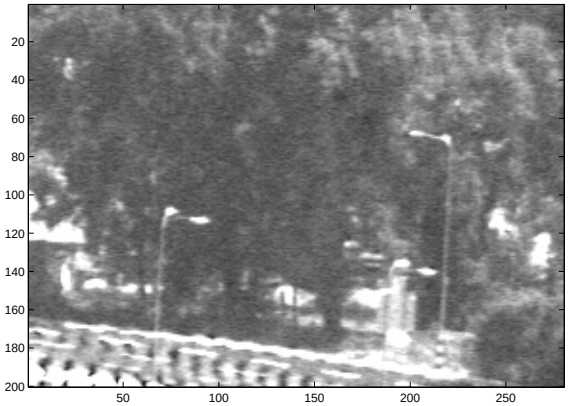
- **Peak signal-to-noise ratio (PSNR)**: This is a commonly used quantitative fidelity criteria (in image processing applications). Let $i_b(x,y)$ be the original pixel in spatial position (x,y) of the spectral band b (of size $N \times M$) and $\hat{i}_b(x,y)$ the respective reconstructed pixel, then, for each spectral band $1 \leq b \leq 120$, $PSNR_b$ is defined by

$$PSNR_b = 10log_{10}\left(\frac{255^2}{\frac{1}{NM}\sum_{x=1}^{N}\sum_{y=1}^{M}[i_b(x,y) - \hat{i}_b(x,y)]^2}\right) \tag{5.1}$$

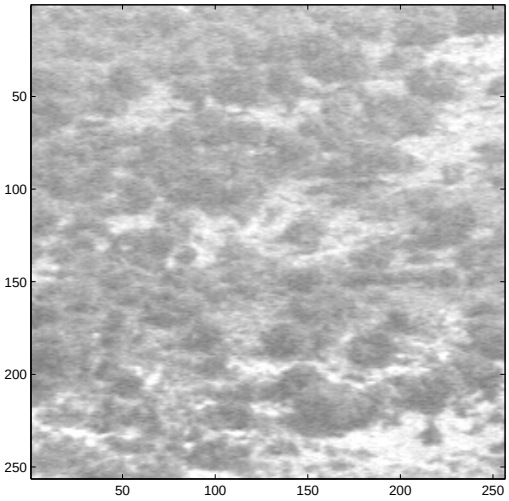
  We use an average PSNR as the quality measure, where the averaging is performed



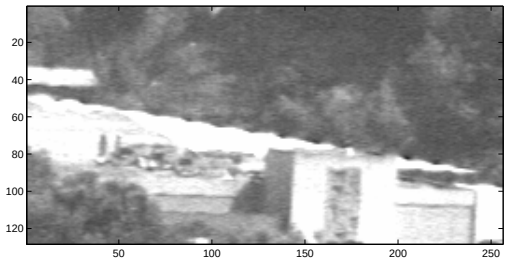
(a)-"glade" cube



(b)-"forest" cube



(c)-"desert" cube



(d)-"building" cube

Figure 5.1: Examples of different scenes (band 60).

over B spectral bands:

$$\overline{PSNR} = \frac{1}{B}\sum_{b=1}^{B} PSNR_b \tag{5.2}$$

(B=120 in our images).

- **Classification Error**: A classification algorithm is introduced in the following subsection as a target application for hyperspectral imagery. The performance measure here is the *classification error* obtained by applying the classifier on both the original and reconstructed cubes and comparing the results.

5.2.1 The Classifier

A simple maximum likelihood (ML) classifier ([37, 38]) was used for classification of each pixel in the scene into one of the three possible classes, according to its spectral characteristics. For the test cube "building" the two classes of interest were 'buildings' and 'trees' while the third class is called 'others' or 'unrecognized'.

The training stage of the classifier consisted of gathering known labeling information of the training pixels from the different training scenes. Partitioning of each class into a number of sub-classes (5 or 6) for more precise modeling of the data as a Gaussian distribution, was performed with a help of VQ clustering [16]. For each group of pixels $\{C_j\}$, belonging to a particular sub-class j, a mean vector and covariance matrix were calculated:

$$\underline{\mu}_j = \frac{1}{|C_j|}\sum_{k\in C_j}\underline{x}_k; \qquad \Sigma_j = \frac{1}{|C_j|}\sum_{k\in C_j}(\underline{x}_k - \underline{\mu}_j)\cdot(\underline{x}_k - \underline{\mu}_j)^T \tag{5.3}$$

where $\underline{x}$ is a training pixel vector containing N spectral values. To avoid the singularity of the covariance matrix (due to a small number of training pixels in some sub-classes) only N=45 bands were used for classification. Therefore, 45 most informative (in classification sense) bands were chosen and used both for training and classification. The band were chosen according to *discriminate analysis feature extraction* (DAFE) criterion, explained in detail in [38], while few bands at the edges of the sensed spectrum window were discarded due to camera's attenuation.

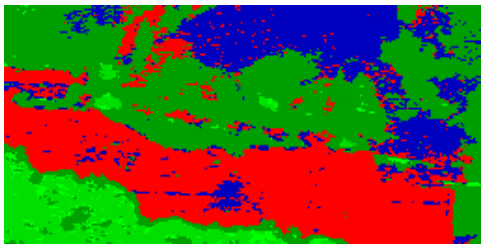
The classification of any pixel $\underline{z}$ (of dimension $N = 45$) in the scene is performed in a following way. A posteriori Gaussian probabilities for each given sub-class are calculated according to

$$P(\underline{z}|sub-class\ j) = \frac{1}{[2\pi\cdot det(\Sigma_j)]^{\frac{N}{2}}} e^{-\frac{1}{2}(\underline{z}-\underline{\mu}_j)^T\Sigma_j^{-1}(\underline{z}-\underline{\mu}_j)} \tag{5.4}$$

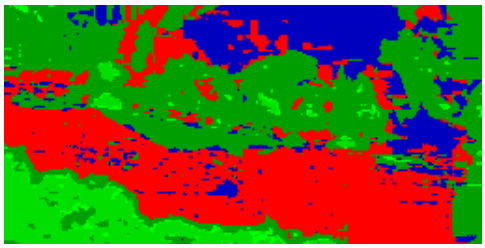
and are compared. Each pixel is classified to the sub-class that achieves maximum probability. If the maximum probability is less than a specified threshold the pixel is tagged as 'unrecognized'.

(a) Original band 60

(b) Reconstructed band 60 (QT, 80:1)



(c) Original classification map



(d) 'Reconstructed' classification map

(e) Post processing on original

(f) Post processing on 'reconstructed'

(g) Manual classification map

Figure 5.2: Examples of the classification algorithm.

The above classification algorithm is based only on the spectral signature of each pixel and does not take into account any spatial dependencies. As a result, in a region belonging to some material many pixel islands (holes) of another material may be generated. To reduce this phenomena we performed a post processing on the resulting classification map. All the sub-classes of a particular class were labeled with the same label. The morphological majority filter, [39], was applied to the classification map to eliminate

small regions. An example of this process is shown in Figure 5.2. The classification map of the original "building" cube (band 60, shown in (a)) is depicted in (c). Grouping of the sub-classes into three classes and the post processed map are illustrated in (e), where 'unrecognized' pixels are whitened. In (g) the result of manual pixel classification is shown; the classification is into 2 classes only - 'trees' and 'buildings'. The classification error is calculated according to percentage of the wrong classification (relative to the manual classification) of the 'recognized' pixels, which means that 'unrecognized' pixels are not taken into account. The same procedure is done for each one of the reconstructed cubes which are distorted by different compression schemes. An example of the "building" cube, coded and reconstructed by the QT algorithm (at CR=80:1) is shown in Figure 5.2 (b), (d) and (f).

5.3 Coding Simulation Results

In the following subsections we introduce the coding simulation results for different algorithms implemented in this work. The first two algorithms constitute benchmarks and the results of the proposed schemes will be compared to them later.

5.3.1 Benchmark Algorithms

First of all we summarize the coding simulation results of the two benchmark algorithms described in Chapter 2.

KP-GS-VQ Algorithm Results

The Kronecker-product gain-shape VQ algorithm was implemented as follows. 4×4 spatial blocks ($K = 16$) were collected from B adjacent spectral bands and encoded. The gain and shape codebooks for VQ were designed utilizing training sequences from the training image archives for each value of B. Note, that each 4x4xB spatio-spectral block (cube) was represented as an optimal gain and shape Kronecker-product form, prior to encoding. Such a representation involves a slight impairment, therefore we show the representation performance too.

Let us remind that the bit-rate (R) in bits per pixel (bpp) and the compression ratio (CR) are defined as $R = \frac{1}{B}$ (since we encode the block of 4x4xB pixels with 16 bits: 6

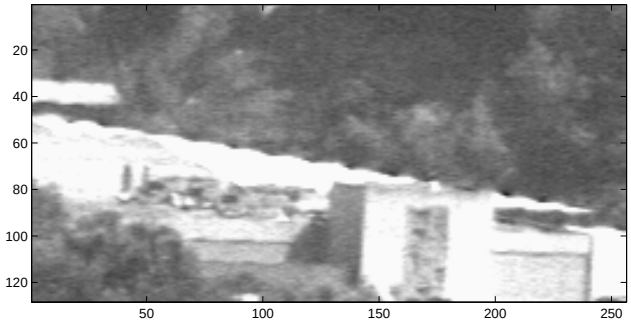
bits for the gain vector ($N_G = 64$) and 10 bits for the shape vector ($N_S = 1024$)) and $CR = \frac{8}{R}$, where B stands for a number of adjacent spectral bands included in a 3D block. The average representation and coding PSNR for different bit-rates (or compression ratios) are summarized in Table 5.1.

CR for "building" cube	32:1 (B=4)	64:1 (B=8)	80:1 (B=10)	128:1 (B=16)
$\overline{PSNR}$ for representation	36.68dB	32.64dB	31.84dB	30.68dB
$\overline{PSNR}$ for encoding	28.81dB	27.57dB	27.33dB	26.16dB

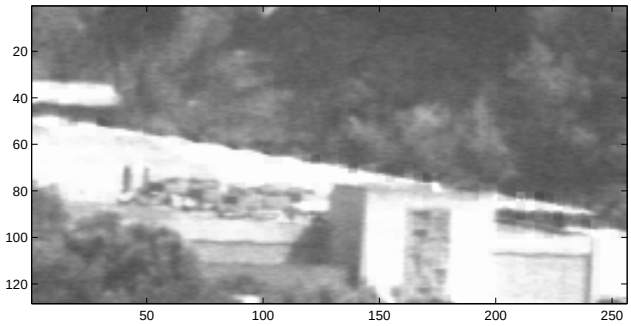
Table 5.1: Average representation and coding PSNR of the "building" cube for different compression ratios for the KP-GS-VQ coding algorithm.

It can be observed that this benchmark achieves quite a good performance (both quantitative and subjective - see the example below). This performance is achieved by taking advantage of spectral redundancies. The compression ratio may be doubled in this scheme with performance impairment (in terms of PSNR) which is not too high, as shown in Table 5.1. It is obvious (from the theoretical explanations) that this is due to the *shape-invariance* assumption which is less correct for larger number of bands being represented by the same shape.

An example of the representation and reconstruction of band 60 from the "building" archive, for $B = 10$ (CR=80:1), are shown in Fig. 5.3 (b) and (c), respectively.

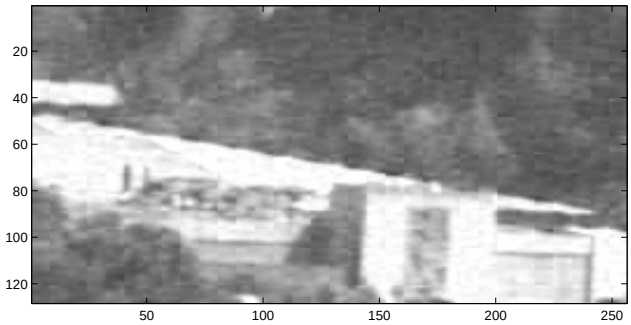


(a) Original band 60



(b) KP-GS-VQ-representation band 60 (B=10)

(c) KP-GS-VQ-reconstructed band 60 (CR=80:1)



(d) DPCM/DCT with TCQ-reconstructed band 60 (CR=76:1)

Figure 5.3: Coding results obtained for band 60 of "building" HIC using the KP-GS-VQ and hybrid DPCM/DCT (with TCQ) benchmark coders.

Hybrid DPCM/DCT with TCQ Algorithm Results

This algorithm was implemented according to the slight changes described earlier in Chapter 2. The first spectral band, which is the reference image in the DPCM loop, was encoded at a total rate (including side-information for the mean of the DC coefficient sequence and standard deviations of all 64 sequences) of $R_1 = 0.75$ bits per pixel (bpp). The error images for the remaining spectral bands were encoded at a total rate of R_s bpp. The total average encoding rate and compression ratio were calculated as follows

$$R_{total} = \frac{R_1 + (N-1) \cdot R_s}{N} \quad , \quad CR = \frac{8}{R_{total}} \tag{5.5}$$

where N is a number of spectral bands and all rates are in bpp. Therefore, the compression ratio achieved for the "building" cube (N=120) by allocating $R_s = 0.1$ bpp is CR=76:1. We see that if more bands participate in the encoding, the resulting compression ratio gets closer to the asymptotic ratio of 80:1. This is a good reason to call R_s an asymptotic encoding rate for this algorithm.

After the DCT transform, the coefficients were rearranged into the 'like-coefficient' sequences (by collecting all the coefficients from the same spatial position in each block), they were normalized and a rate allocation algorithm was invoked. The encoding rates for all the sequences were calculated as denoted in equation (2.25) and were forwarded to a fixed-rate TCQ encoder.

The fixed-rate TCQ encoder used has a full trellis diagram, which was built according to the specified encoding rate, and populated randomly by vectors of independent and identically distributed (i.i.d.) random numbers from an appropriate generalized Gaussian distributions (GGD). In order to determine the best fitting parameter α for GGD, we used two parameter fitting techniques (described in the theoretical subsection 2.3.3), one is based on the Kullback-Leibler divergence and the other on shape estimation. The best parameters are almost the same for both techniques and they were determined to be $\alpha = 0.5$ and $\alpha = 0.9$ for the DC and non-DC coefficients, respectively. Training sequences were generated from the appropriate GGDs with the respective α parameter. Each trellis was optimized utilizing the generalized Lloyd algorithm for each branch in the trellis, while the encoding was performed using the Viterbi algorithm.

The influence of different trellis diagram sizes was also investigated. These results are summarized in Table 5.2 for the "building" cube.

Trellis parameters	K=q=4	K=q=8	K=q=16
$\overline{PSNR}_{cod}$	26.91dB	27.14dB	27.39dB

Table 5.2: Encoding performance of the hybrid DPCM/DCT scheme for different trellis sizes (CR=76:1, "building" cube).

It can be understood from the above table that enlarging trellis diagram size improves the performance of the codebook (since more codevectors participate in the coding process), but the computational complexity increases exponentially because more comparisons have to be done for finding the best codevector for each input vector in the search procedure. As a compromise between computational complexity and performance we have chosen $K = q = 8$ as the best option.

The average PSNR for different bit-rates (or compression ratios) are summarized in Table 5.3, where the asymptotic bit-rate R_s in bpp controls the compression ratio.

CR (R_s)	59:1 (0.13)	76:1 (0.1)	97:1 (0.08)	120:1 (0.06)
$\overline{PSNR}$	27.5dB	27.3dB	26.4dB	25.9dB

Table 5.3: Average PSNR of the reconstructed "building" cube for different compression ratios (K=q=8), using the hybrid DPCM/DCT coding scheme with TCQ.

As an example, the reconstructed band 60 of the "building" cube obtained for a compression ratio of 76:1 ($K = q = 8$) is depicted in Figure 5.3-(d). Although slight, blocking effects can be observed in this technique as a result of performing quantization of the DCT coefficients. The image also looks noisier than the result shown in Figure 5.3-(c), due to the population of the trellis diagrams from a fitted distributions and not from the real training date (while the noise would increase if no optimization of trellis diagrams was performed).

Comparison with the H.261 standard

In addition, we performed simulations with the H.261 video coding standard [40]. The motivation for using the video compression standard is that the hyperspectral image cube can be viewed as the sequence of frames. Therefore, it may be useful to apply video compression techniques to hyperspectral imagery too. The H.261 encoding algorithm uses

motion estimation and compensation and DCT coding of the displaced frame difference (DFD), thus resembling the hybrid benchmark coder. However, since no motion exists in hyperspectral images (except for possible small jitter between frames) we do not expect this coder to outperform the examined algorithms that were designed and adjusted for hyperspectral images.

In order to encode the gray-level hyperspectral bands it was necessary to convert it to CIF format, consisting of Y, U and V components. Since only the luminance component (Y) is available, we generated the two chrominance components, U and V, by filling them with a constant value of 128, according to the standard. The chrominance components are decimated by a factor of two on both the horizontal and vertical directions. The resulting Y image for each band (frame) is of size 352x288. Since the original image occupies a space of 128x256 pixels only, the remaining pixels are filled with the value of 128. The UV components are of size 176x144 each, and are filled with a value of 128 (no chrominance information) as explained above.

As expected, the H.261 coding algorithm allocates only few bits per band for UV images, while almost all the bits are used for encoding Y. It it reasonable to assume that most of the bits are spent on about 1/3 of Y which contains the original hyperspectral frame (the rest of Y is uniformly filled with 128), but since we do not know the exact bit allocation, we show here only the best possible results for this coder ("best case"). The results are summarized in Table 5.4. The original "building" cube size is 3,932,160 Bytes, while the cube of Y images occupies 12,165,120 Bytes (due to the CIF standard size). The compression ratios were calculated according to Y bands and that is the reason for the "best case" results. Actually, the compression ratios are smaller if the original sequence only would have been encoded. Nevertheless, average PSNR calculations are performed with regard to the original bands and therefore are exact.

A comparison of the performance of H.261 to the performance of the other examined coders is shown in Figure 5.8. This comparison indicates that even in the "best case", the video encoder performs worse than the examined algorithms in almost all the range of bit rates. The H.261 coder enjoys the fact that after motion compensation the DFD is of low energy (the background remains unchanged, while the moving objects are compensated) thus leading to a good performance in coding video sequences. In hyperspectral cubes, on the contrary, there usually exist differences in the gray level for almost every pixel between the adjacent frames. That explains why the application of a standard video

Compressed file size	201KB	177KB	152KB	130KB	100KB
CR (R [bpp])	60:1 (0.13)	69:1 (0.11)	80:1 (0.1)	94:1 (0.09)	120:1 (0.06)
$\overline{PSNR}$	28.6dB	27.7dB	26.5dB	25.1dB	19.1dB

Table 5.4: Average PSNR of the reconstructed "building" cube for different compression ratios using the H.261 video standard coder.

coder to hyperspectral images has a relatively low performance.

5.3.2 Proposed 3D-DCT-Based Algorithms

Here we show the coding simulation results of two 3D-DCT-based algorithms described in Chapter 3.

Quantization Table (QT) Scheme Results

The complete scheme for the QT approach is shown in Figure 3.12 in sub-section 3.2.6. The 3D-DCT transform coefficient cubes (8x8x8) were quantized using a 3D quantization table, which is generated according to formula (3.6). The parameters of the quantization function were determined to be $C = 10, \beta_{in} = 0.15, \beta_{out} = 0.05$. This choice of parameters achieves the average PSNR of 29.7dB and an average CR of 75:1, where the average is taken upon the different image cubes in the training set.

According to the chosen parameters, a 3D "standard" quantization table (of size 8x8x8) was generated, in which each entry corresponds to a quantization step of a uniform scalar quantizer for each one of 512 transform coefficients.

The scan order, determined for the selected quantization table, was employed for the reordering of the quantized coefficients into a single vector of length 512. The vectors were further losslessly encoded utilizing the run-size Huffman tables for DC and AC quantized coefficients.

The quality of the reconstructed hyperspectral image is determined already in the quantization (lossy) stage of the algorithm, while for determining the compression ratio lossless coding must be also performed.

In order to achieve a specified average bit-rate (or compression ratio) for a tested hyperspectral image cube we have to adjust the quantization step. Therefore, we encoded

the "building" cube with different quantization steps, while the "standard" table was modified each time by multiplying each entry by a quality factor (Q_f). Thus, for larger values of Q_f the quantization steps increase (coarse quantization) and better compression is achieved (with sacrificed quality of the reconstructed image); smaller values of Q_f lead to small quantization steps (fine quantization), so the compression is decreased but quality is improved. The coding results of this experiment for the tested "building" cube are shown in Figure 5.4.

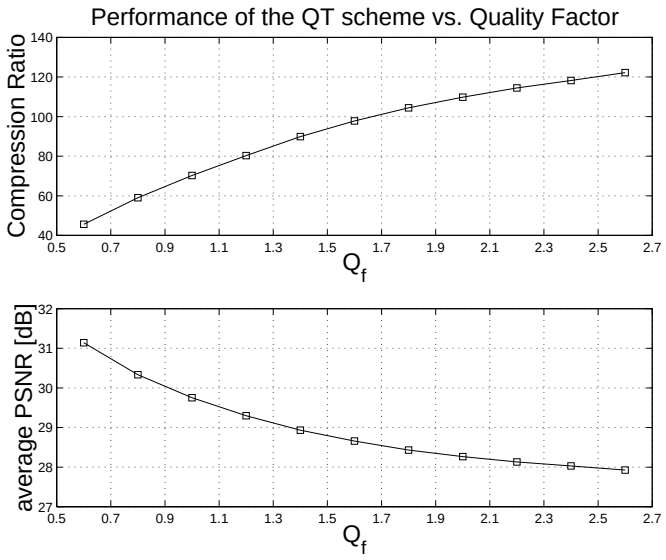


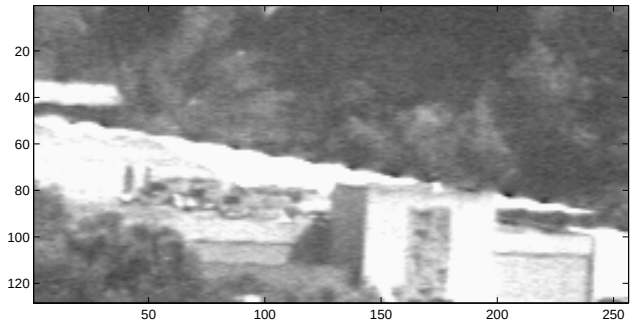
Figure 5.4: Performance of QT scheme for different quality factors (Q_f), "building" HIC.

We can also use these graphs for adjusting the coding scheme to the desired compression ratio (average bit-rate) or quality (in average PSNR sense) of the reconstructed HIC. A coding example is illustrated in Figure 5.5-(b) for compression ratio of 80:1.

Set of Vector Quantizers (VQ) Scheme Results

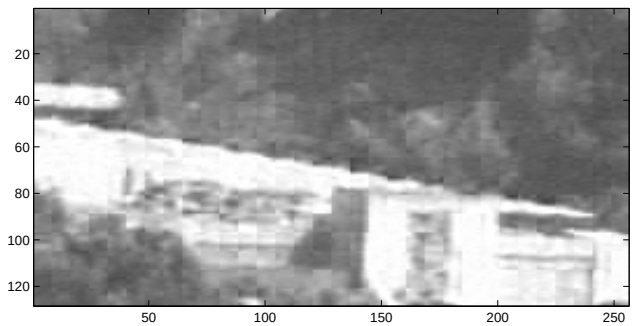
The complete scheme for the VQ approach is shown in Figure **??** in sub-section 3.3.3. Here, the 3D-DCT transform coefficients of each 8x8x8 block are reordered into a single vector using the same scan order as in QT approach. That scan order achieves the best concentration of the high energy coefficients at the head of the vector and the low energy coefficients at its tail.

Each vector is partitioned into 8 sub-vectors. A set of vector quantizers (codebook for each sub-vector) is used to encode the vector of coefficients. The desired average bit-rate or compression ratio determines a fixed number of bits per each 8x8x8 cube of coefficients, while this fixed amount of bits per cube is spread among a set of vector



(a) Original band 60

(b) QT-reconstructed band 60 (CR=80:1)



(c) VQ-reconstructed band 60 (CR=80:1)

Figure 5.5: Coding example of the proposed QT and VQ schemes for CR=80:1.

quantizers (codebooks). It should be noted that variable bit allocation for each block may also be used. But, it involves additional training of large number of codebooks for every possible rate assigned. Moreover, this method requires additional side-information to be available at the decoder for choosing appropriate set of codebooks. These drawbacks were the main reason for choosing a fixed bit-rate allocation per block. The number of codevectors in each codebook increases for the sub-vectors at the head of the vector ("rich" codebooks).

Table 5.5 demonstrates the performance of this techniques for different compression ratios (or bit-rate per 8x8x8 cube) and the coding example is shown in Figure 5.5-(c) for a compression ratio of 80:1.

CR (R_{cube})	60:1 (68)	80:1 (51)	100:1 (41)	120:1 (34)
$\overline{PSNR}$	28.9dB	27.8dB	26.6dB	26.2dB

Table 5.5: Average coding PSNR of the "building" cube for different compression ratios using the VQ approach.

It should be mentioned that as the bit-rate per cube increases, the number of vectors in the "rich" codebooks grows. Therefore the computational complexity (mostly due to the multiple comparisons performed in each codebook during coding procedure) may become very high for high bit-rates.

The blocking effect is visible here due to the fixed number of bits assigned to each cube.

5.3.3 Proposed 3D-SA-DCT-Based Algorithm

Coding simulation results of a 3D-SA-DCT-based algorithm which is described in Chapter 4 are presented here.

The complete scheme of this technique is shown in Figure 4.6. The main idea of shape-adaptive technique is to distinguish between the high and low activity blocks and treat them separately for more efficient coding. High activity blocks are split as described in Chapter 4. The splitting criterion was chosen to be based on variance thresholding.

According to this criterion, the block with a variance exceeding the threshold is considered to be highly active and therefore is split into two lower activity regions utilizing VQ clustering. Each one of the lower activity regions is transformed with a 3D-SA-DCT and the resulting two sets of transform coefficients are encoded by one of the two techniques described earlier (QT or VQ). If the particular block's variance is below the given threshold it is considered to have low activity and therefore it is transformed using a conventional 3D-DCT, whose coefficients can be efficiently encoded again by either the QT or the VQ approach.

The problematic issue here is the bitmap generated by the 2-class VQ clustering. It originally occupies 64 bits per each split block. This data is necessary for correct shape-adaptive transformation and therefore it must not be distorted. After lossless compression, using a "snake" scan followed by run-length Huffman coding, this side-information was reduced to 32 bits per split block (4 bits per spectral band in the block) on average. This

compressed data must be available to the decoder and has to be included in the total rate allocation.

Since coding of the transform coefficient is done according to one of the proposed schemes, we have to adjust the parameters for both types of blocks: those passing through the 3D-DCT and those passing through the 3D-SA-DCT. The training data was collected from the training HICs for two types of blocks and the parameters of the quantization function were determined empirically as described in subsection 3.2.6. The splitting threshold was chosen to be $\theta = 500$. The results are shown in table 5.6. Note, that the two sets of coefficients are treated the same way since we can not globally distinguish between them. Compression ratios for each method given in the table are for the QT approach and they

Method	C	β_{in}	β_{out}	CR
3D-DCT (all the blocks)	10	0.15	0.05	75:1
3D-DCT (low activity)	10	0.11	0.05	82:1
3D-SA-DCT	8	0.13	0.07	53:1

Table 5.6: The adjusted parameters of the quantization function defined in (3.6).

are given for blocks of the same type only. Following the setting of the parameters, scan orders for each type of blocks were also generated. The codebooks in the VQ approach, as well as Huffman tables in QT approach, for both types of blocks were trained.

As mentioned in the theoretical section, we tried to find the best threshold value of θ in the splitting criterion. It is intuitively reasonable to expect that there exist an optimal value because of the following. If the threshold is low, many block are split, but on average about a half of the bits allocated to the block are spent on the less important set of 3D-SA-DCT. This is mainly due to the fact that even non-active blocks (without an edge or texture) are forced to be split by the two-class VQ. Subsequently, such an allocation spends bits without improving performance and leads to a performance which is worse comparing to the non shape-adaptive scheme. On the other hand for a high threshold value we split only the blocks with high variance which are a minority, therefore we should asymptotically converge to the non shape-adaptive scheme. Finally, if the SA scheme succeeds to achieve better performance over the non-SA scheme, in spite of the high bit overhead (2 SA sets and the bitmap side-information), then we should observe a maximum performance for some optimal threshold value θ_{opt}.

We tested this approach for five different values of the threshold parameter θ: 500, 700, 1000, 1500 and 2000. Let N be the total number of 8x8x8 cubes in the tested HIC, N_1 the number of unsplit cubes and N_2 the number of split cubes. It is obvious that $N = N_1 + N_2$. In addition we denote by R_{cube} the desired average bit-rate allocated per 8x8x8 cube; R_1 is the bit-rate allocated for unsplit cubes and R_2 is the bit-rate allocated to each one of the split cubes. Then the following equation must hold:

$$N_1 \cdot R_1 + N_2 \cdot (2 \cdot R_2 + 32) = N \cdot R_{cube} \tag{5.6}$$

where 32 is the bitmap side-information overhead for the split cubes and the rate of the split cubes is doubled since two sets of the coefficients have to be coded. All the rates are in bits per 512 coefficient cubes and the total compression ratio is calculated by $CR = \frac{512 \cdot 8}{R_{cube}}$.

Since split blocks are encoded less efficiently than unsplit blocks, even after splitting them into two lower activity regions (see Table 5.6) and because these blocks mostly affect the reconstructed image quality, we allocated the bits using $R_2 = 2 \cdot R_1$. Thus, for a given value of the threshold parameter θ, the values of N_1 and N_2 are determined in the preprocessing stage of the algorithm, whereas the encoding rates R_1 and R_2 for unsplit and split blocks, respectively, are calculated from the former assumption and formula 5.6.

The bit-rates for both types of blocks are used for bit allocation for each sub-vector in the VQ approach. With regard to the QT scheme, we have to determine the quality factors Q_{f1} and Q_{f2} for the respective unsplit and split blocks which results in adjusting the quantization table for the desired block bit rates R_1 and R_2. Figure 5.6 illustrates a number of parametric curves of average block bit-rates versus quality factor for both cases. We use a simple lookup table for finding appropriate quality factors corresponding to the desired block bit-rates. It can be observed that the curves with higher θ are located higher on the graphs. The reason for this is that for the same quality factor higher values of θ result in more active blocks to be coded (with larger variations in pixel values within the block), thus on the average more bits are required for their encoding. In addition, the blocks that passed the 3D-SA-DCT are more active, therefore even after splitting into two lower energy regions they still require more bits (on average) than the lower activity regular blocks.

A coding example of the shape-adaptive approach is demonstrated in Figure 5.7. Band 60 of the "building" cube is encoded in all the cases at the average bit rate of $R = 51$ bit per cube (block), corresponding to a compression ratio of 80:1. Pictures on the left

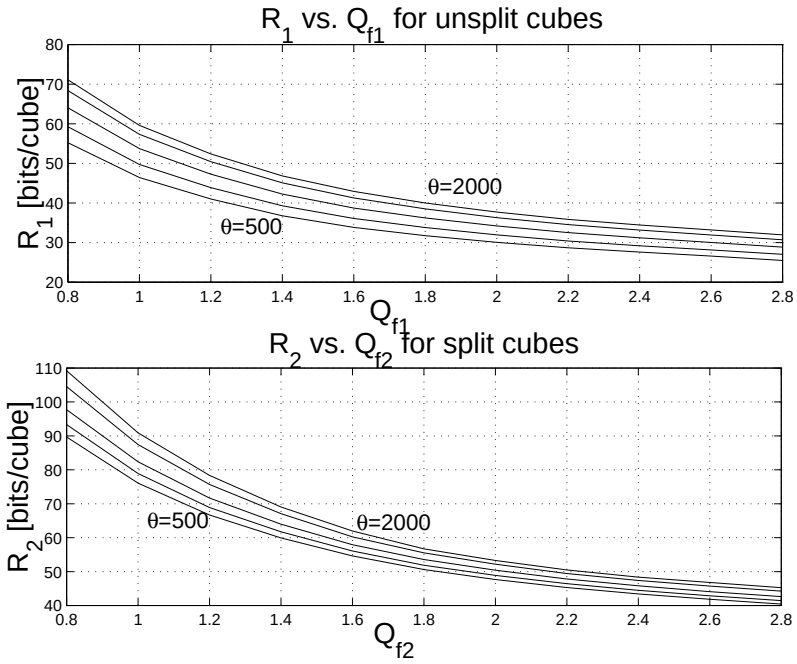


Figure 5.6: Parametric curves of average bit-rate vs. quality factor for both types of blocks for different threshold values θ.

depict the VQ SA technique for the different values of the threshold parameter θ, whereas pictures on the right correspond to the same values of the parameter but for the QT SA technique. Pictures (k) and (l) show the respective performance of both techniques for the regular (non shape-adaptive) approach.

We compare and summarize the simulation results as well as classification results in the following section.

(a) SA-VQ $\theta = 500$

(b) SA-QT $\theta = 500$

(c) SA-VQ $\theta = 700$

(d) SA-QT $\theta = 700$

(e) SA-VQ $\theta = 1000$

(f) SA-QT $\theta = 1000$

(g) SA-VQ $\theta = 1500$

(h) SA-QT $\theta = 1500$

(i) SA-VQ $\theta = 2000$

(j) SA-QT $\theta = 2000$

(k) VQ no splitting

(l) QT no splitting

Figure 5.7: Coding example of "building" HIC for different values of θ at a constant bit-rate $R = 51[bit/cube]$ (CR=80:1) for QT (right) and VQ (left).

5.4 Summary of Coding and Classification Results

All the results, coding and classification, for the examined algorithms are summarized and compared in this section.

Coding Summary

The coding performance of the different algorithms is summarized in Figure 5.8. Here, the proposed non shape-adaptive schemes (QT and VQ) are compared to the benchmarks. The graph shows the average PSNR obtained for each algorithm versus the average bit-

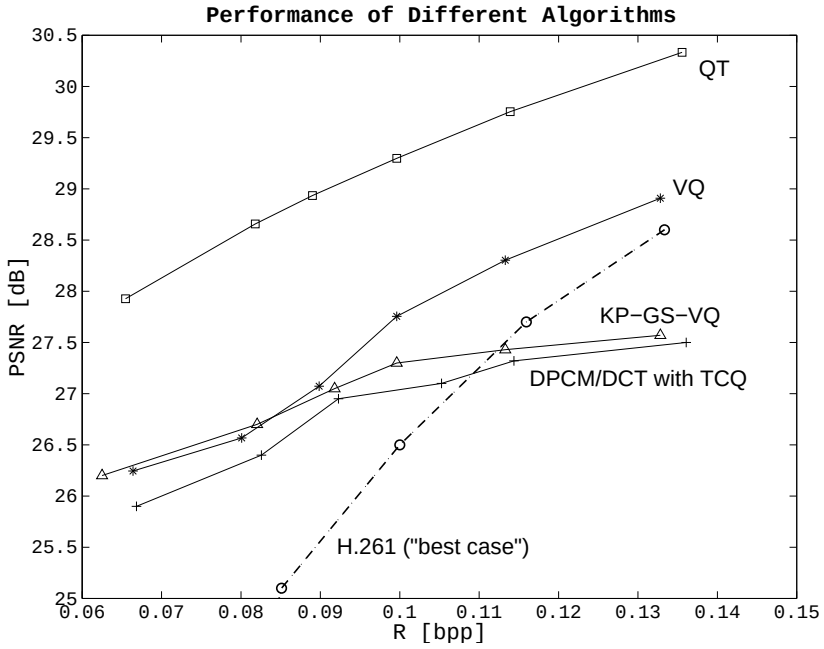


Figure 5.8: Coding performance summary (without shape-adaptive coding).

rate in bits per pixel. The rate axis corresponds to compression ratios between 60:1 (0.14[bpp]) and 130:1 (0.06[bpp]). The coding examples demonstrated earlier for different algorithms are for the compression ratio CR=80:1 that corresponds to bit-rate R=0.1[bpp] on the graph.

The graph obviously shows the advantage of the proposed techniques over both benchmarks. Both the VQ and the QT techniques achieve better PSNR for the same bit-rate than KP-GS-VQ and DPCM/DCT with TCQ algorithms in almost all the range of bit-rate values. The QT method is superior to others by about 1.5-2[dB] in PSNR. It should be noted that only in the QT scheme lossless compression was applied, since it is a built-in feature of the QT approach. But even when comparing the remaining algorithms (without lossless coding) the advantage of the proposed VQ method is visible.

The issue of encoding the indices losslessly for each sub-vector was also examined. For each sub-vector the occurrence of indices in the corresponding codebook was measured and the entropy of the indices was calculated. It was found that the resulting entropy is within less than 1 bit from the actual number of bits required for index representation or, in other words, the histogram of the indices is almost flat. This means that applying Huffman coding to indices will gain almost no advantage, except for complicating the algorithm and adding the side-information for Huffman tables. For example, if 10 bits were allocated to the first sub-vector of length 4, the entropy of the indices is 9.4 bits, therefore Huffman coding can achieve average bit-rate between 9.4 bits to 9.65 bits per codebook index. Not only the advantage of lossless coding is negligible, but also applying Huffman tables of size 1024 (in this example) is impractical. The similar results were obtained for the remaining sub-vectors.

As mentioned in the previous section, the performance of the H.261 coder (which includes the lossless coding) is relatively low even for the "best case" shown in the graph.

There are a number of reasons for this advantage of the QT scheme. First, 3D-DCT proved to be a good energy compaction transformation, taking advantage of both spatial and spectral redundancies existing in hyperspectral data. Second, a quantization table approach succeeded to take into account (by fine quantization) the dominant transform coefficients (near-DC 3D corner) that are responsible for the good quality of the reconstructed cube and discard the less or non-important coefficients. Coarse quantization of the non dominant coefficients and a smart reordering lets the encoder to achieve higher compression ratios with minimal quality impairment. The VQ scheme performs worse than the QT scheme due to the fixed rate allocation to each 8x8x8 block that spreads the same amount of bits among all the blocks, regardless of their compresability and quality of the representation by a set of vector quantizers. On the contrary, the QT scheme works according to a desired average bit-rate, allowing the lossless run-size Huffman table encoder to spread the bits efficiently among the different blocks. Therefore, less active block requires less bits for encoding quantized transform coefficient, whereas more active blocks require more bits.

As mentioned above, the next step was to apply a shape-adaptive approach for active blocks and to compare it with non shape-adaptive method. The results for 3D-SA-DCT with either VQ or QT techniques are shown in figures 5.9 and 5.10, respectively. The compression ratios include the side-information overhead. The performance of the non-

SA scheme for the same encoding rates (or CRs) is depicted by the dotted lines. Examples of the reconstructed "building" image were shown in Figure 5.7 for CR=80:1.

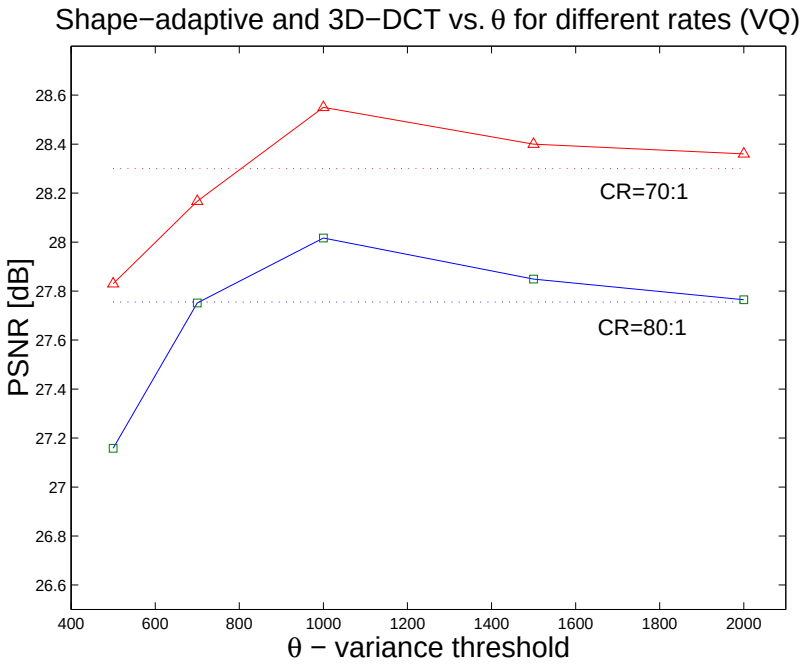


Figure 5.9: Coding performance with SA approach for VQ technique.

As desired, the VQ scheme combined with the shape-adaptive (3D-SA-DCT) approach is able to perform better than the same scheme without 3D-SA-DCT. The best performance is achieved for some optimal threshold value of the parameter θ, which is $\theta_{opt} = 1000$ in the shown examples (generally, different optimal values may be obtained for different compression ratios). This leaves about 14% of blocks (in the "building" HIC example) to be split and transformed by 3D-SA-DCT. The superiority of this approach is not high enough for utilizing it, due to additional complexity of the algorithm. The reason for this behavior is a high overhead characteristic to this approach, resulting from the need to code two sets of transform coefficients and the bitmap. The double encoding of two lower energy regions of a split block may be better than using the conventional 3D-DCT for coding the original high activity block. But, the bitmap side-information which consists of 32 bit per each split block (on average) ruins the advantage achieved by the splitting.

That's why the SA approach does not have any advantage at all in the QT scheme. It can be observed, however, that the performance of the SA method gets close to the non-SA one as the compression ratio decreases. This happens because of the smaller effect of the bitmap side-information on the total performance at higher bit-rates.

Nevertheless, there is in principle a case for using this approach, since there is evidence

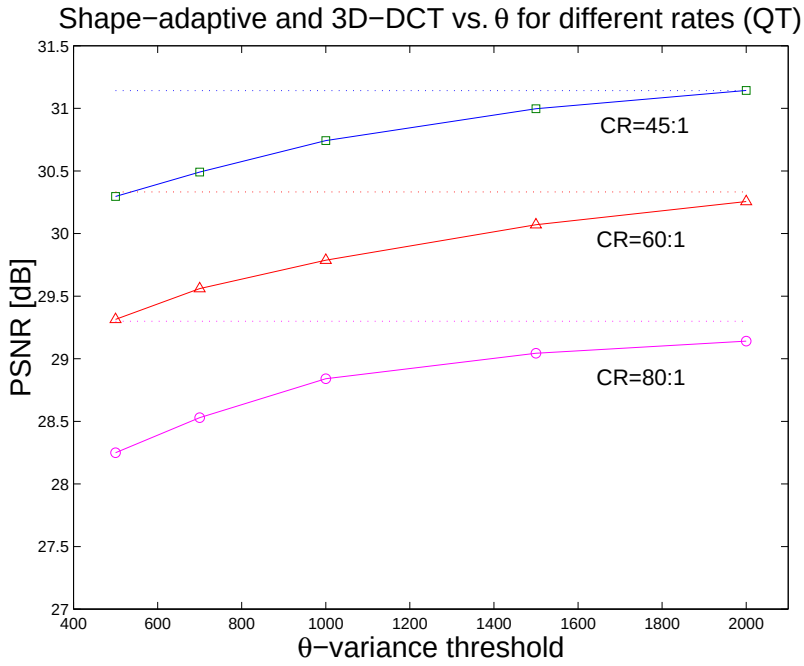


Figure 5.10: Coding performance with SA approach for QT technique.

to the superiority of shape-adaptive DCT transform over the regular one for high activity blocks. Further investigation is needed to improve the splitting criterion and to reduce the side-information overhead of the bitmap.

Classification Summary

Here, we summarize the classification results obtained from the different algorithms for both SA and non-SA approaches. The process of extraction of the classification error of each reconstructed cube was explained in detail in subsection 5.2.1.

Figure 5.11 illustrates the classification performance of the different algorithms (for non-SA methods). The classification error of the ML classifier on the original "building" HIC is 10.3% and it is shown by the horizontal solid line. Although theoretically the coding distortion can improve the classification error of the classifier (by better separation of the classes from one another), all the tested algorithms impair the classification error. Here, the VQ technique is superior to the other (non-SA) algorithms, almost for all the range of bit-rate values. It can be also observed that the classifier retains consistent behavior since increased bit-rate reduces the error and converges to the original classification error.

We tested also the performance of the classifier on the reconstructed cubes obtained from the coding schemes employing the shape-adaptive approach for different bit-rates and for different values of the threshold parameter θ. The results are depicted in Figure 5.12.

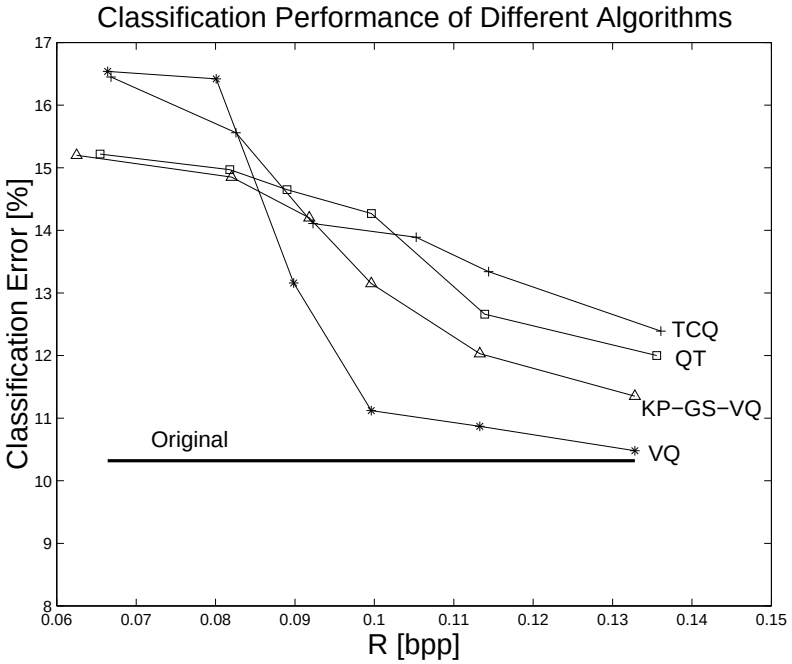


Figure 5.11: Classification performance summary (without shape-adaptive coding).

Also here the VQ technique performs better than QT. Two interesting phenomena can be observed from the graph. The first one shows that an optimal threshold parameter θ exists for every encoding bit rate for both techniques (VQ and QT); that is there exists a value of θ for which a minimum classification error is achieved. The optimal classification error is always smaller for the SA approach than for the non-SA schemes for the same encoding bit-rate. The second phenomenon is, as mentioned above, the ability of the coding system to improve the classification performance in comparison to the performance obtained for the original data. Although we can not predict or ensure this behavior, it is obvious that encoding active blocks with SA approach succeeds to improve the overall classification performance by better separating the different classes (as compared to the original HIC).

In spite of these promising results, they are definitely can not be considered as a fully reliable performance measure. This is due to a number of assumptions we applied (for example, manual labeling and post-processing morphological filtering) and also because of the small amount of training data available (about 20% of pixels are unrecognized by the classifier). Therefore, better training must be done and a better classification algorithm must be developed in order to obtain more reliable conclusions.

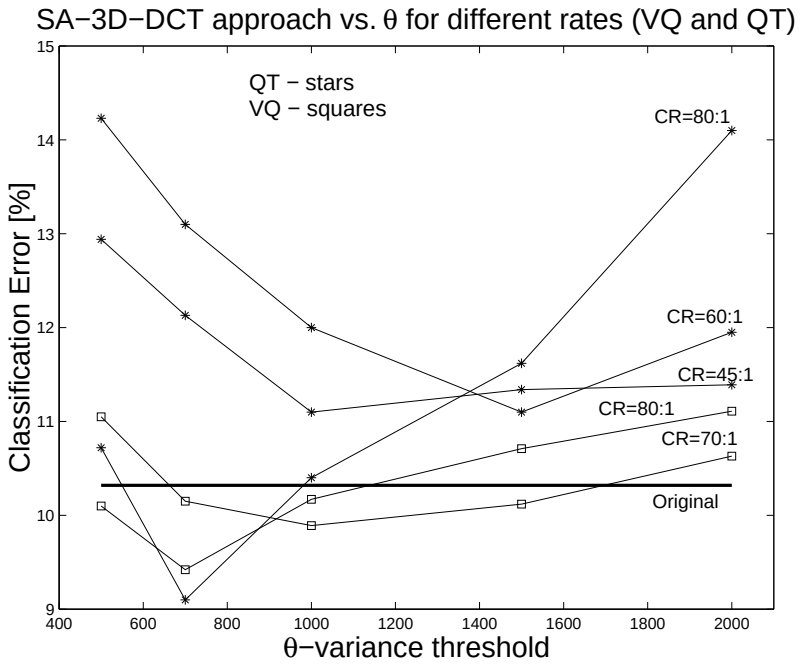


Figure 5.12: Classification performance with SA approach for both VQ and QT techniques.

Chapter 6

Summary, Conclusions and Future Studies

In this research, we investigated two approaches for hyperspectral image compression using 3D transforms. Two techniques for transform coefficients coding are proposed which can be used in each of the two approaches. The conclusions of this research are summarized below, followed by propositions for further studies.

6.1 Summary and Conclusions

At the first step of this research, two state of the art algorithms were implemented. These schemes, namely: the Kronecker-product gain-shape VQ algorithm and the hybrid DPCM/DCT scheme with TCQ, served as benchmarks. The H.261 standard video coder was also examined on the hyperspectral images. Then, an approach using a 3D-DCT was applied. Two techniques, the so-called QT and VQ, for transform coefficients coding, were proposed and implemented. Finally, a novel approach utilizing a SA-3D-DCT was proposed and investigated for more efficient coding of active blocks. An application for hyperspectral imagery was also introduced and applied as an additional performance measure for all the examined coding algorithms.

A simple calculation shows that a hyperspectral image with reasonable spatio-spectral resolution may contain a huge amount of data. Lossy coding of the data can compress the information with high compression ratios with reasonable distortion of the reconstructed image. The implemented benchmarks achieved compression ratios of about 80:1 (0.1 bpp)

with an average PSNR of about 27dB. We oriented our designs to this reference point, although all the algorithms were tested also at different bit-rates corresponding to different compression ratios.

As the simulation results show, the proposed 3D-DCT-based approach tends to exploit better both the spatial and the spectral correlation existing in the hyperspectral image cube. The three-dimensional transform concentrates the energy of the coefficients near the DC coefficient, while the quantization table algorithm (QT) or quantization with a set of vector quantizers (VQ) technique efficiently codes them. The VQ scheme outperforms the benchmarks by about 0.5dB (in PSNR), while the QT scheme is superior to the benchmarks by about 2dB, for the same average bit-rate (or compression ratio). The difference comes from the block bit allocation. The VQ algorithm is a fixed rate allocation scheme since the same desired bit-rate is allocated to each 8x8x8 block. Alternatively, the QT algorithm promises an average desired bit-rate, while the amount of bits that are allocated to each block differs from one block to another. Thus, more active blocks are encoded with more bits and vice versa. Such an adaptation is enabled by a run-size extractor followed by Huffman tables. The H.261 standard video coder did not succeed to achieve a performance better than the proposed schemes (even for the "best case"). Therefore, we can conclude that the models applied for video can not be used for the hyperspectral images in a straightforward way.

The idea of shape-adaptive (SA) coding ([24, 29]) implies that active blocks are treated separately. An active block is split into a number of lower activity regions. However, simulation results show only a slight advantage of SA coding with the VQ technique and no advantage at all of SA coding with the QT one, as compared to the standard 3D-DCT-based approach combined with these techniques. The main reason for the lack of success is a high side-information overhead due to the bitmap which describes the boundaries between the regions in the split blocks. The lossless 2:1 compression of the bitmap that we succeeded to achieve, still leaves us with 32 bits of side-information for each split block, which is a lot of data at low bit-rates. Therefore, the CR of the coefficients has to be increased, leading to a degradation in overall performance. In addition, the splitting criterion, chosen to be a variance threshold, is not good enough and does not detect all the blocks that will gain in performance from splitting them. Blocks for which the advantage is visible have a high variance, but not all the blocks with high variance show a coding advantage. Nevertheless, we conclude that this approach promises better results if the

drawbacks are overcome.

Since the hyperspectral imagery subsequently has to be analysed by a specific application, we employed a classification application. It was applied to the reconstructed cubes from the different coding schemes. The simulation results show that for the 3D-DCT-based approach, as expected, all the schemes impair the misclassification rate as compared to the original one, while the VQ method achieves the least impairment. Concerning the 3D-SA-DCT-based approach, the proposed schemes perform better than the respective ones without splitting, for the same bit-rate. Moreover, an interesting phenomenon takes place: The coding distortion may improve classification performance! But, due to the simplicity of the classifier, the small amount of data and a number of assumptions made, we cannot surely conclude on this issue. A more robust classification model should rather be developed.

Finally, concerning the computation complexity of the proposed schemes, it can be defined as moderate to high. Although the KP-GS-VQ benchmark tends to have a moderate complexity (despite the covariance matrix and eigen values calculations), the DPCM/DCT with TCQ benchmark has a high complexity, mostly due to the dynamic programming related to trellis coding. In the approaches proposed in this work 3D transforms are the main computational load, but efficient implementations exists for the DCT transform. Although the QT algorithm does not have a high additional computation load, the VQ method suffers from a high computational load at high bit-rates (low compression ratios) because of large codebooks. So, globally, the proposed schemes have a lower complexity than the benchmarks, at least at low bit-rates.

6.2 Propositions for Future Studies

The main directions for further studies can be summarized as follows:

- Concerning the 3D-DCT approach: we used the adjusted quantization function and the scan order derived from the generated quantization table as proposed by Lee in [22]. Other models for exponential functions and scan orders should be considered for more efficient quantization and reordering.

- Concerning the 3D-SA-DCT approach, we have two directions for improvements, which could utilize the potential of this approach. First of all, a better splitting

criterion should be defined. Although variance is a simple criterion it is not good enough. Since the standard gradient-based techniques worked even worse, we assume that a more sophisticated "spectral" edge detector should be developed. Secondly, the side-information need to be reduced. The fact that the VQ clustering algorithm, which is responsible for the bitmap generation, may produce many small holes and discontinuities, probably does not permit higher lossless compression of bitmaps. We suggest here to perform post- processing of the bitmaps for smoothing region boundaries and eliminate very small regions using, for example, morphological filters. Such an operation may slightly impair the performance of the SA-based encoder, but will definitely improve the lossless compression ratio.

- The last issue concerns the classification algorithm. More training data should be available for reducing the number of unrecognized pixels. Moreover, for exact error measuring, the real labeling for each pixel should be available. The better classifier should also take advantage of the spatial relations between the pixels.

Appendix A

Optimal Kronecker Product Representation Gain and Shape

This appendix refers to the discussion in Chapter 2 (subsection 2.2.3). Here we would like to minimize the representation squared error

$$\|\mathbf{X}-\tilde{\mathbf{X}}\|^2 = \|\mathbf{X}-\mathbf{g}\otimes\mathbf{s}\|^2 = \|\mathbf{X}\|^2 + \|\mathbf{g}\|^2\cdot\|\mathbf{s}\|^2 - 2\cdot\langle\mathbf{X},\mathbf{g}\otimes\mathbf{s}\rangle \tag{A.1}$$

By defining

$$\mathbf{z}(\mathbf{s}) = (\langle\mathbf{x}_1,\hat{\mathbf{s}}\rangle,\langle\mathbf{x}_2,\hat{\mathbf{s}}\rangle,\ldots,\langle\mathbf{x}_B,\hat{\mathbf{s}}\rangle) \tag{A.2}$$

and using the constraint $\|\mathbf{s}\|=1$ the expression in (A.1) can be rewritten as

$$\|\mathbf{X}-\mathbf{g}\otimes\mathbf{s}\|^2 = \|\mathbf{X}\|^2 + \|\mathbf{g}\|^2 - 2\cdot\langle\mathbf{z}(\mathbf{s}),\mathbf{g}\rangle \tag{A.3}$$

So the problem of minimization turns to be the following maximization problem - one has to maximize the expression

$$\langle\mathbf{z}(\mathbf{s}),\mathbf{g}\rangle - \frac{1}{2}\cdot\|\mathbf{g}\|^2 \le |\mathbf{z}(\mathbf{s})|\cdot|\mathbf{g}| - \frac{1}{2}\cdot\|\mathbf{g}\|^2 \tag{A.4}$$

while the last step was according to Cauchy-Schwartz inequality.

Now, for every optimal shape, $\mathbf{s}_{opt}$, the gain that will maximize (A.4) must be $\mathbf{g}_{opt} = \beta\cdot\mathbf{z}(\mathbf{s}_{opt})$, therefore one have to maximize the expression $(\beta - \frac{1}{2}\cdot\beta^2)\cdot\|\mathbf{z}(\mathbf{s})\|^2$. By the simple derivation by β we have $\beta_{opt}=1 \Longrightarrow \mathbf{g}_{opt} = \mathbf{z}(\mathbf{s}_{opt})$. It remains only to determine $\mathbf{s}_{opt}$ from maximization of

$$\|\mathbf{z}(\mathbf{s})\|^2 = \sum_{b=1}^{B}\langle\mathbf{s},\mathbf{x}_b\rangle^2 = \mathbf{s}\cdot(\sum_{b=1}^{B}\mathbf{x}_b^T\cdot\mathbf{x}_b)\cdot\mathbf{s}^T = \mathbf{s}\cdot\mathbf{R}_X\cdot\mathbf{s}^T \tag{A.5}$$

where $\mathbf{R}_X$ is a symmetric real matrix of size $K \times K$, having $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_K$ and $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_K$ as the eigenvalues and orthogonal eigenvectors respectively. A well known solution of the problem comes from the linear algebra and it states that the solution of (A.5) is the dominant eigenvector of $\mathbf{R}_X$. Summarizing these results the optimal representation gain and shape vectors, minimizing the squared error, are

$$\mathbf{s}_{opt} = \mathbf{u}_1 \qquad \mathbf{g}_{opt} = \mathbf{z}(\mathbf{u}_1)$$

Appendix B

Tables of Scan Order Indices and Quantization Values

This appendix refers to the discussion in Chapter 3 (subsection 3.2.3). We show here an example of scan order generation according to a quantization table values based on the parameter settings: $A_{in} = 255, A_{out} = 255, \beta_{in} = 0.15, \beta_{out} = 0.05$ and $C = 10$ in equation 3.6. The procedure to construct this table is described in detail in subection 3.2.3.

Coef. no.	Coord. no. (wuv)	Quant. val.	Inside region?	Reordered coef. no. (scan order)	Coord. no. (wuv)	Sorted quant. val.	Inside region?
0	000	1	Yes	0	000	1	Yes
1	001	37	Yes	1	001	37	Yes
2	002	67	Yes	8	010	37	Yes
3	003	93	Yes	64	100	37	Yes
4	004	116	Yes	2	002	67	Yes
5	005	136	Yes	16	020	67	Yes
6	006	152	Yes	128	200	67	Yes
7	007	167	Yes	3	003	93	Yes
8	010	37	Yes	9	011	93	Yes
9	011	93	Yes	24	030	93	Yes
10	012	136	Yes	65	101	93	Yes
11	013	167	Yes	72	110	93	Yes
12	014	190	Yes	.	.	.	.
13	015	115	No	.	.	.	.
14	016	128	No	96	040	190	Yes
15	017	140	No	257	401	190	Yes
				264	410	190	Yes
16	020	67	Yes				
17	021	136	Yes	13	015	115	No
18	022	179	Yes	19	023	115	No
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
510	776	255	No	510	776	255	No
511	777	255	No	511	777	255	No

Table B.1: Table of quantization values (left) and table of scan order indices (right).

References

[1] G. R. Canta and G. Poggi, "Kronecker-product gain-shape vector quantization for multispectral and hyperspectral image coding", IEEE Trans. on Image Processing, Vol. 7, No. 5, pp. 668-678, May 1998.

[2] G. P. Abousleman, "Compression of hyperspectral imagery using hybrid DPCM/DCT and entropy-constrained trellis coded quantization", DCC'95 Data Compression Conference, IEEE Comput. Soc. Press, Los Alamitos, CA, pp. 322-331, 28-30 March 1995.

[3] G. P. Abousleman, M. W. Marcellin and B. R. Hunt, "Compression of hyperspectral imagery using 3-d DCT and hybrid DPCM/DCT", IEEE Trans. Geosci. Remote Sensing, Vol. 33, pp. 26-34, Jan. 1995.

[4] G. P. Abousleman, "Hyperspectral image coding using wavelet transforms and trellis coded quantization", Proceedings of the SPIE, Vol. 2491, pp. 1096-1106, 1995.

[5] G. Gelli and G. Poggi, "Compression of multispectral images by spectral classification and transform coding", IEEE Trans. on Image Processing, Vol. 8, No. 4, pp. 446-489, April 1999.

[6] J. Lee, "Optimized quadtree for Karhunen-Loeve transform in multispectral image coding", IEEE Trans. on Image Processing, Vol. 8, No. 4, pp. 453-461, April 1999.

[7] M. J. Ryan and J. F. Arnold, "The lossless compression of AVIRIS images by vector quantization", IEEE Trans. Geosci. Remote Sensing, Vol. 35, No. 3, pp. 546-550, May 1997.

[8] M. W. Marcellin and T. R. Fischer, "Trellis coded quantization of memoryless and Gauss-Markov sources", IEEE Trans. Commun., Vol. 38, pp. 82-93, Jan. 1990.

[9] T. R. Fischer and M. Wang, "Entropy-constrained trellis coded quantization", IEEE Trans. Inform. Theory, Vol. 38, No. 2, pp. 415-425, Mar. 1992.

[10] P. A. Chou, T. Lookabauch and R. M. Gray, "Entropy-constrained vector quantization", IEEE Trans. on Acoust., Speech and Signal Processing, Vol. 37, No. 1, pp. 31-42, Jan. 1989.

[11] M. W. Marcellin, "On entropy-constrained trellis coded quantization", IEEE Trans. Commun., Vol. 42, No. 1, pp. 14-16, Jan. 1994.

[12] Y. Shoham and A. Gersho, "Efficient bit allocation for an arbitrary set of quantizers", IEEE Trans. on Acoust., Speech and Signal Processing, Vol. 36, No. 9, pp. 1445-1453, Sep. 1988.

[13] M. W. Marcellin, P. Sriram and K. L. Tong, "Transform coding of monochrome and color images using trellis coded quantization", IEEE Trans. Circuits Syst. Video Technol., Vol. 3, No. 4, pp. 270-276, Aug. 1993.

[14] G. D. Forney, Jr., "The Viterbi algorithm", Proc. IEEE, Vol. 61, pp. 268-278, Mar. 1973.

[15] Y. Linde, A. Buzo and R. Gray, "An algorithm for Vector Quantizer Design", IEEE Trans. Commun., Vol. COM-28, No. 1, pp. 84-95, Jan. 1980.

[16] A. Gersho and R. M. Gray, "Vector Quantization and Signal Compression", Boston, MA: Kluwer, 1992.

[17] T. M. Cover and J. A. Thomas, "Elements of Information Theory", J. Wiley & Sons, New York, 1991.

[18] A. K. Jain, "Fundamentals of Digital Image Processing", Prentis-Hall, Englewood Cliffs, NJ, 1989.

[19] S. Farkash, "Transform trellis coding of images", Research Thesis, Technion - Israel Institute of Technology, Haifa, Israel, Mar. 1988.

[20] E. Rotem, "Subband coding of images using trellis coder", Research Thesis, Technion - Israel Institute of Technology, Haifa, Israel, Aug. 1987.

[21] W. M. Porter and H. T. Enmark, "A system overview of the Airborne Visible/Infrared Imaging Spectrometer (AVIRIS)", Imaging Spectroscopy II, G. Vane, Editor, Proc. SPIE 834, pp. 22-29, 1987.

[22] M. C. Lee, R. K. W. Chan and D. A. Adjeroh, "Quantization of 3D-DCT Coefficients and Scan Order for Video Compression", Journal of Visual Communication and Image Representation, Vol. 8, No. 4, pp. 405-422, 1997.

[23] B. L. Yeo and B.Liu, "Volume rendering of DCT-based compressed 3D scalar data", IEEE Trans. on Visualization and Computer Graphics, Vol. 1, pp. 29-43, Mar. 1995.

[24] T. Sikora and B. Makai, "Shape-Adaptive DCT for generic coding of video", IEEE Trans. Circuits Syst. Video Technol., Vol. 5, No. 1, pp. 59-62, Feb. 1995.

[25] H. Sagan, "Space-Filling Curves", Springer-Verlag, New York, 1994.

[26] Editor C. H. Chen, "Information processing for remote sensing", World Scientific Co., Singapore, 1999.

[27] I.-T. R. T.81, "Information technology - Digital compression and coding of continuous-tone still images - Requirements and guidelines", tech. rep., ISO/IEC JTC1 10918-1, 1994.

[28] G. K. Wallace, "The JPEG still picture compression standard", IEEE Trans. on Consumer Electronics, Vol. 38, pp. 18-34, Feb. 1992.

[29] Y. Zhao, "Segmentation-based shape-adaptive image coding", Research Thesis, Technion - Israel Institute of Technology, Haifa, Israel, May. 1999.

[30] G. Fernandez, "Multispectral Imaging: Acquisition, Analisys, Compression and Applications", Research D.Sc. Thesis, The Polytechnique University of Catalone, Barcelone, Spain, 1997.

[31] H. S. Hou, "Digital document processing", J. Wiley, 1983.

[32] J. Alber and R. Niedermeier, "On Multi-dimensional Hilbert Indexing", Technical Report 98-392, KAM-DIMATIA Series, Faculty of Mathematics and Physics, Charles University, Praha, Czech Republic, May 1998.

[33] P. A. Maragos and R. W. Schafer, "Morphological Skeleton Representation and Coding of Binary Images", IEEE Trans. on Acoust., Speech and Signal Processing, Vol. 34, No. 5, Oct. 1986.

[34] J. Pandel, "Variable bit-rate image sequence coding with adaptive quantization", Image Communication, Vol. 3, No. 2-3, pp. 123-128, June 1991.

[35] J. Canny, "A Computational Approach to Edge Detection", IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. 8, No. 6, pp. 679-698, Nov. 1986.

[36] K. Sharifi and A. Leon-Garcia, "Estimation of Shape Parameter for Generalized Gaussian Distribution in Subband Decomposition of Video", IEEE Trans. Circuits Syst. Video Technol., Vol. 5, No. 1, pp. 52-56, Feb. 1995.

[37] R. O. Duda, P. E. Hart and D. Stork, "Pattern Classification", Second Edition, 2000.

[38] D. Landgrebe, "Some Fundamentals and Methods for Hyperspectral Image Data Analysis", SPIE Photonics West, San Jose CA, Jan. 1999.

[39] C. Gu and M. Kunt, "Contour simplification by a new nonlinear filter for region-based coding", in VCIP-94, Vol. 2308, pp. 1180-1191, Sept. 1994.

[40] V. Bhaskaran and K. Konstantinides, "Image and Video Compression Standards", Kluwer Academic Publishers, Boston, 1995.