DCT Based SPIHT Architecture for Hyperspectral Image Data Compression

A Thesis Presented
by
Jieming Xu
to
The Department of Electrical and Computer Engineering
in partial fulfillment of the requirements
for the degree of
Master of Science
in
Electrical and Computer Engineering

Northeastern University
Boston, Massachusetts

August 2016
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>List of Figures</td>
<td>iii</td>
</tr>
<tr>
<td>List of Tables</td>
<td>v</td>
</tr>
<tr>
<td>List of Acronyms</td>
<td>vi</td>
</tr>
<tr>
<td>Acknowledgments</td>
<td>vii</td>
</tr>
<tr>
<td>Abstract of the Thesis</td>
<td>viii</td>
</tr>
<tr>
<td>1 Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Hyperspectral image</td>
<td>1</td>
</tr>
<tr>
<td>1.1.1 AVIRIS Data</td>
<td>2</td>
</tr>
<tr>
<td>1.1.2 AIRS Data</td>
<td>3</td>
</tr>
<tr>
<td>1.1.3 Reflection Bands</td>
<td>3</td>
</tr>
<tr>
<td>1.1.4 Radiation Models</td>
<td>3</td>
</tr>
<tr>
<td>1.2 Hyperspectral Image Data Compression</td>
<td>6</td>
</tr>
<tr>
<td>1.2.1 Lossy &amp; Lossless Compression</td>
<td>6</td>
</tr>
<tr>
<td>1.2.2 Distortion Measure</td>
<td>7</td>
</tr>
<tr>
<td>1.2.3 Rate-Distortion Curve</td>
<td>9</td>
</tr>
<tr>
<td>1.2.4 Spectral &amp; Spatial Accessibility</td>
<td>10</td>
</tr>
<tr>
<td>1.2.5 Hyperspectral Compression System</td>
<td>10</td>
</tr>
<tr>
<td>1.3 Summary</td>
<td>11</td>
</tr>
<tr>
<td>2 Review of Background in Signal Coding</td>
<td>12</td>
</tr>
<tr>
<td>2.1 Coding Techniques</td>
<td>12</td>
</tr>
<tr>
<td>2.1.1 Entropy Coding</td>
<td>12</td>
</tr>
<tr>
<td>2.1.2 Linear-Predictive Coding</td>
<td>13</td>
</tr>
<tr>
<td>2.1.3 Vector Quantization</td>
<td>14</td>
</tr>
<tr>
<td>2.1.4 Bitplane Coding</td>
<td>15</td>
</tr>
<tr>
<td>2.2 Transform Coding</td>
<td>16</td>
</tr>
<tr>
<td>2.2.1 Discrete Cosine Transform</td>
<td>16</td>
</tr>
<tr>
<td>2.2.2 Discrete Wavelet Transform</td>
<td>17</td>
</tr>
<tr>
<td>2.2.3 KL Transform</td>
<td>19</td>
</tr>
</tbody>
</table>
## List of Figures

1.1 Hyperspectral Image Structure .................................................. 2  
1.2 Solar Radiation Model ............................................................... 4  
1.3 Spectral Signature of Different Objects ...................................... 5  
1.4 Atmosphere Transmittance .......................................................... 6  
1.5 Object Reflectance Curve ............................................................ 7  
1.6 Rate-Distortion Curve ................................................................. 9  
1.7 Transmitter .............................................................................. 10  
1.8 Receiver ............................................................................... 10  
2.1 Predictive Coding System ........................................................... 14  
2.2 Multi-level Decomposition .......................................................... 18  
2.3 Harr Wavelets ........................................................................... 18  
2.4 2-Level Wavelet Decomposition .................................................. 20  
2.5 8×8 Block Coefficients and MSB Map ......................................... 22  
3.1 3 Dimensional DWT .................................................................... 26  
3.2 3D-Hierarchical Tree Organization ............................................. 27  
3.3 Set patronization in DWT ............................................................. 30  
3.4 2-Level Filter Bank .................................................................... 34  
3.5 Implementation of 3D-DWT .......................................................... 34  
3.6 3D Filter Bank .......................................................................... 35  
3.7 Cube Tiling .............................................................................. 36  
3.8 Level Shifting Comparasion ....................................................... 37  
4.1 Large Code Cube Compression System ....................................... 41  
4.2 Small Code Cube Compression System ....................................... 42  
4.3 Cube Based Parallel Computing .................................................. 42  
4.4 bitstream1 .................................................................................. 43  
4.5 32×32×32 Division .................................................................... 46  
4.6 16×16×16 Division ................................................................... 47  
4.7 PCRD Control Unit .................................................................... 48  
4.8 Parallel Organization .................................................................. 49  
4.9 Trees Reorganization ................................................................. 49  
4.10 Bitstream .................................................................................. 50
5.1 Cuprite .......................................................... 52
5.2 Moffett Field ...................................................... 52
5.3 Sample .......................................................... 52
5.4 Bitrate-SNR of Moffett Field ................................. 53
5.5 Bitrate-SNR of Cuprite .......................................... 54
5.6 Bitrate-SNR of Sample .......................................... 55
5.7 Spectral Profile at 1.3 bpppb ................................. 54
5.8 Spectral Profile at 0.3 bpppb ................................. 55
5.9 Spectral Profile at 4.5 bpppb ................................. 56
5.10 DWT Coefficients .............................................. 57
5.11 Bitplanes of 16×16×16 DCT Coefficients ............... 59
5.12 Bitplanes of 16×16×16 DWT Coefficients .............. 60
List of Tables

1.1 Spectrum Table ................................. 4
List of Acronyms

AVIRS  Airborne Visible InfraRed Imaging Spectrometers
AR    Auto Regressive
bpppb  bit per pixel per band
DCT   Discrete Cosine Transform
DWT   Discrete Wavelet Transform
EZW   Embedded Zerotree Wavelet
EBCOT Embedded Block Coding with Optimal Truncation
HSI data Hyperspectral Imaging data
IR    Discrete Cosine Transform
LWIR  Long Wave Infrared
MAD   Maximum Absolute Difference
MSB   Most Significant Bit
MWIR  Mid Wave Infrared
PMAD  Percentage Maximum Absolute Difference
PSNR  Peak Signal-to-Noise Ratio
SNR   Signal-to-Noise Ratio
SPIHT Set Partitioning In Hierarchical Trees
SWIR  Short Wave Infrared
3D-SPIHT 3-Dimensional Set Partitioning In Hierarchical Trees
Acknowledgments

First of all, I would like to thank Prof. Vinay K. Ingle. As a thesis advisor, he gave me a lot of help and support as possible as he can through this thesis. Secondly, I would like to thank Prof. Bahram Shafai and Prof. Hanoch Lev-Ari for being my thesis committee and Prof. Lev-Ari gave many worthy suggestions on this thesis. Finally, I would like to thank my parents and friends who supported my study and research in Northeastern University.
Abstract of the Thesis

DCT Based SPIHT Architecture for Hyperspectral Image Data Compression

by

Jieming Xu

Master of Science in Electrical and Computer Engineering
Northeastern University, August 2016
Prof. Vinay K. Ingle, Adviser

The wavelet transformation is leading and widely used technology in transform coding. Many coding algorithms are designed for this transformation based on its unique structure, such as EZW, SPIHT and SPECK. The correlation between each subbands naturally generates a special tree structure in the whole image. With bitplane and entropy coding technique, the compression ratio can be achieved at a very high value. In this thesis, we focus on the traditional discrete cosine transform (DCT) to design our compression system. After analyzing the performance of the SPIHT algorithm, we found that, the coefficients’ arrangement in DCT still has the features similar to these in wavelet transform and these features are vital to maximize the performance of SPIHT algorithm.

For realistic implementation, a large hyperspectral data cube must be tiled into small size of code cubes for compression to achieve a fast compression speed. In JPEG standard, two applicable block sizes are given, which are $8 \times 8$ and $16 \times 16$ blocks. In our system, we extend these blocks into three-dimensional cube for hyperspectral image as $8 \times 8 \times 8$ and $16 \times 16 \times 16$ cubes for testing. To enhance the compression performance, PCRD algorithm is also applied in our system. Because the values in spectrum direction share very similar trajectory for each pixel, the power of several continuous bands is predictable. In this way, we optimized the PCRD algorithm for our system, and the truncation points can be chosen without calculation which saves time.

Three AVIRIS hyperspectral data sets are tested. For DCT based compression, each image cube was tiled into sizes of $8 \times 8 \times 8$ and $16 \times 16 \times 16$ cubes for transformation and compression independently. For DWT based compression in small code cube setting, each image cube was tiled into sizes of $32 \times 32 \times 32$ and $16 \times 16 \times 16$ with a DWT decomposition level of five and four. For DWT compression in large code cube setting, the transformation (five levels of DWT) and compression algorithm was performed on the whole image cube in a size of $448 \times 448 \times 224$. Results showed
that, the DCT based compression with $16 \times 16 \times 16$ code cube size has the best performance for lossy hyperspectral image compression and the bitplane arrangement is more effective for SPIHT algorithm.
Chapter 1

Introduction

In this thesis, we study and develop HSI data compression technique. Basic notion of HSI data, which includes the definition of hyperspectral image and some basic acronym of image compression is discussed in the first chapter. In the second chapter, a review of several coding techniques for image compression are presented and we outline our approach to the compression of hyperspectral image. In Chapter 3 and Chapter 4, the proposed algorithm and practical problems are introduced and solved. Chapter 5 provides a comparison of results between different scenarios and the reasons are analyzed. In the last chapter, conclusions and future work are given.

1.1 Hyperspectral image

The rapid development of remote sensing technique in different research areas accelerated the study in hyperspectral data, especially in phase of hyperspectral image compression. In our common lives, color images are consisting of three primary colors which are red (0.7µm), green (0.53µm) and blue (0.45µm). In human’s visual system, these colors can synthesize most colors that we can see in the real world. Very similar to color images, hyperspectral images also have multi-bands but the number of bands is much larger than those in color image.

Because of the implementation of highly sensitive sensors on airplanes or remote sensing satellites, the sensors can detect many invisible frequency bands to our eyes. Some typical hyperspectral images have several or hundreds of bands, and for ultraspectral images, they may have thousands of bands. The hyperspectral data is organized in a three-dimensional structure, and Figure 1.1 showed the features of this kind of structure.
Along with the spatial axes that reflects the spatial information of an image, the third dimension on spectra represents the spectral information at each pixel. Each value in a pixel vector in spectral dimension indicates the intensity of a given frequency’s radiation at pixel location. Because many more bands that we can have in hyperspectral images than in the common colored ones, each pixel provide richer information from its spectral signature.

### 1.1.1 AVIRIS Data

AVIRIS shorts for Airborne Visible InfraRed Imaging Spectrometers, which is widely used in remote sensing. The AVIRIS data can provide spectral information in a continuous 224 bands from 0.4\(\mu\)m to 2.5\(\mu\)m wavelength with a 10nm nominal bandwidth. The main purpose of AVIRIS is to survey the changes of earth atmosphere’s attribute change from its absorption, reflectance, and scattering features. From these atmosphere’s data, the scientist can track the environment and climate change. The AVIRIS is flown on two different heights, which are 20km height on ER-2 jet and 4km height on Twin Otter aircraft.

"The general shape of an AVIRIS spectrum is dominated by the light curve of the Sun and the absorption features of the atmosphere and the Sun has a "blackbody" curve, which in the case of the Sun peaks in the green wavelengths and diminishes at higher and lower wavelengths" [1]. The received data among different bands is correspond to the object’s reflectance attributes and
atmosphere absorption. At the bands with low transmittance, the spectra curve will be represented in deep valleys. For example, the valley of many spectra curves around 1.4\(\mu\)m is mainly caused by vapor and carbon dioxide. The peaks are often caused by the solar radiation, in many spectral curves, the highest peak is often around a wavelength of 0.53\(\mu\)m where the strongest radiation of sun light exists. The data is quantized into 10 or 12 bits which depends on the date. AVIRIS data covers a range of NIR and SWIR bands and the system has a 12Hz ”whisk broom” scanning rate with 76GB storage.

1.1.2 AIRS Data

AIRS stands for Atmospheric Infrared Sounder and is the standard reference in compression studies of ultraspectral data. It provides a number of 2378 spectral bands from 3.7 to 15.4 microns and the data is ranging from 12 bits to 14 bits which depending on the bands. The AIRS data covers the whole self-emitted IR bands. ”The mission of AIRS is to observe and characterize the entire atmospheric column from the surface to the top of the atmosphere in terms of surface emissivity and temperature, atmospheric temperature and humidity profiles, cloud amount and height, and the spectral outgoing infrared radiation” [2].

1.1.3 Reflection Bands

The spectral region in wavelength for remote sensing is shown in the Table 1.1. Sensors are designed for these bands since the atmosphere is almost transparent in these regions except for some isolated absorption peaks of vapor and carbon dioxide in 2.5 - 3\(\mu\)m and 5 – 8\(\mu\)m region. However, these absorption bands are not a main factor in the whole sensible bands. In passive remote sensing, the sensors measure radiations that are naturally reflected or emitted from the objects themselves. The visible, IR and SWIR bands belong to the reflectance regime since the reflectance of solar radiation in these bands takes a dominant part in a power sense than the that emitted by objects themselves. On the contrary, the MWIR and LWIR bands are dominated by self-emitted radiation.

1.1.4 Radiation Models

All materials on earth passively absorbs, reflects and transmits radiation at a wavelength from 0.4 to 3\(\mu\)m. The main power resource in radiation for passive remote sensing are the reflectance of solar radiation. The Sun can be approximately assumed as a black body, which only emits radiation at an effective temperature. From the literature, it is a good approximation of the solar
CHAPTER 1. INTRODUCTION

<table>
<thead>
<tr>
<th>name</th>
<th>Wavelength range</th>
<th>radiation source</th>
<th>surface property of interest</th>
</tr>
</thead>
<tbody>
<tr>
<td>Visible</td>
<td>0.4 - 0.7 µm</td>
<td>solar</td>
<td>reflectance</td>
</tr>
<tr>
<td>Near InfraRed</td>
<td>0.7-1.1 µm</td>
<td>solar</td>
<td>reflectance</td>
</tr>
<tr>
<td>Short Wave InfraRed</td>
<td>1.1-1.35 µm</td>
<td>solar</td>
<td>reflectance</td>
</tr>
<tr>
<td></td>
<td>1.4-1.8 µm</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>2-2.5 µm</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mid Wave InfraRed</td>
<td>3-4 µm</td>
<td>thermal</td>
<td>reflectance or temperature</td>
</tr>
<tr>
<td></td>
<td>4.5-5 µm</td>
<td>thermal</td>
<td>temperature</td>
</tr>
<tr>
<td>Long Wave InfraRed</td>
<td>8-9.5 µm</td>
<td>thermal</td>
<td>temperature</td>
</tr>
<tr>
<td></td>
<td>10-14 µm</td>
<td>thermal</td>
<td>temperature</td>
</tr>
</tbody>
</table>

Table 1.1

![Figure 1.2: Solar Radiation above Earth](image)

radiation model at a temperature of 5900 K (Kelvin). The solar radiation power curve outside the earth atmosphere from 0.4 to 2.4 µm is shown in Figure 1.2. In this figure, the solar radiation power is fading dramatically outside the visible light wavelength (0.38 to 0.75 µm).

After ignoring the factors caused by the sensor itself, the atmosphere model and the variability of material types are two main factors that caused the difference in spectral signature. Figure 1.3 showed the difference in spectral signatures of different type of materials. In this figure, the curves showed the radiation of artificial object, grass land and river. We can see that, the artificial object has a very strong radiation from 400 µm to 650 µm band when compared to the grass land. On the other side, river and grass land have high reflection bands between 700 µm to 900 µm. For
CHAPTER 1. INTRODUCTION

Figure 1.3: Spectral Signature of Different Objects

a recognition purpose, we can hardly judge objects from their shapes at a high-altitude, because of their low spatial resolution and special angle. However, from the spectral signature of different objects, their patterns can be easily defined. For example, the green dashed line in Figure 1.3 has an abrupt jump at about 0.7\(\mu\)m, which indicates this pixel is vegetation.

In NIR and SWIR remote sensing, the atmosphere is not a negligible factor in radiation measurement. Different components in earth atmosphere have different transmittance and absorption attribute in radiation. The transmittance curve of earth atmosphere is shown in Figure 1.4.

From this figure, the water vapor and carbon dioxide block almost all the energy in bands near 1.4\(\mu\)m and 1.9\(\mu\)m. The absorption features caused two deep valleys near these bands, which makes the values near zero in AVIRIS data. Near 0.8\(\mu\)m, 0.9\(\mu\)m and 1.1\(\mu\)m, there exist three absorption peaks of carbon dioxide and water vapor. In other word, the other bands will have a good transmittance, which have a lower effect on the radiation reflection from the earth atmosphere. The spectral curves in Figure 1.5 showed this phenomenon in AVIRIS data.

From the curve, there may be two highest peaks around 0.5\(\mu\)m and 0.8\(\mu\)m, since the solar radiation has the strongest power at these bands and the transmittance coefficient is approximately at 0.7 to 0.8 except for some isolated absorption peaks. The combination of these two factors caused the high power radiation region in these bands. For many NIR reflectance objects, four discrete radiation energy peaks appear around 0.9\(\mu\)m, 1.2\(\mu\)m, 1.6\(\mu\)m and 2\(\mu\)m in a decreasing order. The absorption bands of atmosphere caused this discreteness between these peaks and the decreasing
power of solar radiation is the reason of why solar-reflectance power of these peaks goes down as the wavelength becomes longer. In sum, the radiation power of different peaks always performs in a decreasing order, which is mainly correspond to the attribute of solar radiation.

1.2 Hyperspectral Image Data Compression

Since typical hyperspectral images have larger dynamic range and more bands than traditional images, the size of hyperspectral images become much larger than traditional ones. For example, a size of $943 \times 7465$ AVIRIS image is $3.53 \text{GB}$ (Giga Byte), which takes a lot of space and time for storage and transmission, especially for some band and storage limited areas, like satellite remote sensing. As a result, image compression plays a vital role in a remote sensing system.

1.2.1 Lossy & Lossless Compression

An image compression system consists of an encoder that sends an encoded image through a channel and a receiver that can decode the transmitted signal to recover the original image. There are two different kinds of way to compress images: lossy and lossless image compression. For lossless image compression, the recovered image should be identical to the original one. However, this way of compression may not achieve a very high compression ratio.
CHAPTER 1. INTRODUCTION

Figure 1.5: Object Reflectance Curve

The other way is lossy image compression. For a recovered image in lossy compression, the loss of some information from original image is allowed. On the other hand, any compression ratio can be achieved for a given information fidelity criterion. The goal of lossy image compression system is to minimize the difference between original images and the recovered ones under a given compression ratio. In our project, a lossy compression system is designed for HIS data compression.

1.2.2 Distortion Measure

In lossy image compression, a criterion is needed for evaluating the difference between original images and recovered ones. SNR (Signal to Noise Ratio), PSNR (Peak Signal to Noise Ratio), and their definitions are shown below. The SNR measure is defined in the following form:

$$SNR = 10 \log_{10} \left[ \frac{\sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(m,n)^2}{\sum_{x=0}^{M-1} \sum_{y=0}^{N-1} \left(f(x,y) - \hat{f}(m,n)\right)^2} \right]$$

(1.1)

Where $f(x, y)$ and $\hat{f}(x, y)$ represent the original and recovered value of each pixel in an image, and the value in this formula is in decibel. In some situations, especially for hyperspectral image, PSNR
CHAPTER 1. INTRODUCTION

is wider used, which is defined as:

\[
PSNR = 10 \log_{10} \left( \frac{P}{\frac{1}{mn} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} (f(x, y) - \hat{f}(x, y))^2} \right)
\]

(1.2)

where \( P \) is the highest value of each pixel that this image system can represent. For example, in a
gray scaled image, each pixel is coded in a 8-bit depth format, the value it can represent is ranging
from 0 to 255, therefore, \( P \) should be 255.

The value of SNR is decided by the error and the signal itself relatively. This works
very well for some low-power signal’s measure. However, for some peak-limited channels or
homogeneous images, the PSNR usually outperforms SNR.

Except for these mainly used criterion, two other measures are also introduced: MAD
(Maximum Absolute Distortion) and PMAD (Percentage Maximum Absolute Distortion).

MAD measures the maximum difference between recovered image and its original ones
in absolute value. In other words, MAD is independent from its original signal. In this way, the
MAD can make sure the error of the whole image is within an absolute bound. On the contrary, the
deficiency of this method is obvious, the low-power signals will always have a lower value of error
when compared to the high-power signals.

As a result, PMAD is designed for solving the deficiency of MAD measurement. PMAD
criterion measures the absolute difference between original and recovered signals versus the original
ones. This method guarantees all the values are restricted in some range when compared to its
original ones.

In the end, classification accuracy is another measurement for hyperspectral images. To
some degree, most of the hyperspectral images are not designed for human visual system, the rich
spectrum information of each object is difficult to be understood by human eyes. As a result, some
pattern recognition techniques are implemented to help classify different objects in an image by
machine. After classification, a fuzzy image will be segmented into several classes, which helps
human to decide what is in the image. Therefore, the classification accuracy is defined as the
percentage of difference between the original and compressed image. Because the classification
accuracy error is highly correlated to different classification algorithms, this criterion is only used for
some specific situation, or just as a complementary part.
1.2.3 Rate-Distortion Curve

In our thesis, we analyze the performance of compression in distortion-bitrate curve which is used in lossy image compression. Figure 1.6 shows a typical distortion-bitrate curve of a compressed hyperspectral image. The distortion is represented by SNR.

![Rate-Distortion Curve](image)

Figure 1.6: Rate-Distortion Curve

From this curve, the SNR value increases rapidly at the very beginning. That is because for a system to achieve the highest SNR value in a very low bitrate, it will choose some largest values to compress, which contains most energy of the whole image. Usually, these values represent the low frequency part of an image, which depicts the contour and the main objects in an image. Commonly, for a natural image, a large part of energy will be concentrated in low frequency bands. After transformation, all this energy in low frequency will be transformed into several large values, and the amount of these values is relatively small. As a result, just a very low bitrate is needed for compressing these values.

For some compressed image with high fidelity, most of the bits will be consumed to compress the details or texture of an image which are usually distributed in high frequency bands. These values contribute a little to the whole image in energy, however, greatly improve the visual effects and details’ representation. In Figure 1.6, as the bitrate grows faster, the increasing speed of the SNR becomes lower and lower. In the end, a large amount of bitrate increment won’t make a
CHAPTER 1. INTRODUCTION

great improvement to the noise reduction. Therefore, for high-fidelity lossy image compression, the main task is how to compress the scattered high-frequency values in a relatively low bitrate.

1.2.4 Spectral & Spatial Accessibility

Spatial accessibility is an ability that the compression system can access an arbitrary cropped image without decoding the whole one. In hyperspectral image compression, we generalize this definition to the spectral dimension, which means the ability to access any cropped image in any several continuous bands. With this kind of feature, a system can easily access any interested region in high resolution after getting the low resolution part of the whole image. This ability is very useful in multi-resolution image processing. For convenience, users can access any part of the image in a high resolution without decoding the whole image, which is time saving and coding efficiency.

1.2.5 Hyperspectral Compression System

The very basic Hyperspectral compression system is shown in Figure 1.7 and Figure 1.8. In

![Figure 1.7: Transmitter](image)

![Figure 1.8: Receiver](image)

Figure 1.7: Transmitter

Figure 1.8: Receiver

In the compression part, a variety of techniques can be applied and the compressed image will be sent to transmitter. In some literatures, the pre-processing will generate some side information for transmission independently. However, in practical system, the side information will
CHAPTER 1. INTRODUCTION

be embedded into the compressed image bitstream as control information. Therefore, we organize all this information together for compression.

At the receiver side, the image content will be decoded firstly and the post processing part is just the reversed operation of pre-processing part in transmitter side. Based on the quality of recovered image, it will be used in classification, detection or some other research areas.

1.3 Summary

In this chapter, we introduced the structure of hyperspectral data at the very beginning. In the following, we discussed spectral signature of hyperspectral images based on the reflectance attribute of different objects and the transmittance of atmosphere. By taking advantage of the spectral signature, some process for compression can be applied and this will be specified in the following chapter. At last, we introduced some criterion to decided the quality for lossy image compression and a compression system for hyperspectral data is also raised.
Chapter 2

Review of Background in Signal Coding

In this chapter we will briefly review coding techniques that are widely used in image or other media compression. These techniques are used in different application areas based on their specific features. Some of the wavelet based coding and entropy coding techniques are introduced in this chapter which are the basis of our proposed compression system.

2.1 Coding Techniques

In this section, spatial transform-domain coding techniques are discussed. Except for vector quantization, all the other techniques can be applied both for the original data coding and transformed data coding.

2.1.1 Entropy Coding

The notion of information entropy was firstly introduced by Claude Elwood Shannon, the founder of information theory. In his work: A Mathematical Theory of Communication [4], he introduced probability into a communication system. The information that a symbol contains is correlated to the probability it appears, which is measured by entropy. The formula is:

\[ H[s] = \sum_{i=1}^{m} p_i \log_2 \left( \frac{1}{p_i} \right) \]  

(2.1)

where \( H[s] \) represents the entropy of signal source, and there are \( m \) symbols with a probability of \( p_i \) to appear in signal source. As a result, the entropy coding is a approach to maximize the entropy of each symbol and each symbol in the transmission channel shall be efficiently used. In this section,
CHAPTER 2. REVIEW OF BACKGROUND IN SIGNAL CODING

we will briefly review three dominant entropy coding techniques.

- **Run-Length Coding**: Run-length Coding was introduced by Golomb in his work: *Run-Length Encodings* [5]. The long-continuously same symbols will be encoded into short encoded codewords, which is in a more generalized form: Golomb Coding. In this form, an integer $x$ will be mapped into different codewords. Large values will be mapped into long codeword, and vice versa. For common run-length coding, each symbol in the signal will be assumed as independent and identical distributed random variable. Therefore, the signal will be in a geometric distribution. To maximize the maximum entropy of basic Golomb code, the order $k$ of continuous symbol should satisfy [5]:

$$p^k + p^{k+1} \leq 1 < p^k + p^{k-1}$$  \hspace{1cm} (2.2)

- **Huffman Coding**: Huffman Coding is another entropy coding, which maps the symbol with high entropy to long codewords and with low entropy to short one [6]. Huffman code is a prefix code, which is very easy for the encoder design. However, one deficiency for Huffman coding is that, it can’t optimally match the entropy with different length of code words, especially for some symbols with non-integer entropy.

- **Arithmetic Coding**: To enhance the performance of Huffman coding, Arithmetic coding was generalized. In this form of coding, the signal will be coded into a rational number between 0 to 1 [7]. To some degree, arithmetic coding can perfectly achieve the max entropy of each symbol.

All these techniques of entropy coding that we referred are belong to lossless coding, the information can be perfectly recovered from the encoded signal.

2.1.2 Linear-Predictive Coding

A hyperspectral image is spatial and spectral correlated and the current value can be predicted using other values from spatial and spectral in two directions. In linear-predictive coding, the signal is assumed to be an AR (Auto regressive) process, and the prediction error is calculated after compression. The prediction error is defined as the difference between original and predicted signals. This error signal is called innovation, which contains all the information that we need to recover the original signal. If a signal is an AR signal or performs very similar to an AR signal and
CHAPTER 2. REVIEW OF BACKGROUND IN SIGNAL CODING

The prediction process is well-designed, the original signal can be transformed into innovation with a low dynamic range, which is very easy to be compressed.

A typical one-dimensional Linear-Predictive Coding system is shown in Figure 2.1:

In general case, we assume the signal is in a one-dimensional form. The signal needs compression is a white noise passed through a colored filter. In this model, what the compression system needs to do is just calculating the colored filter and whitening the original signal to remove the redundancy and reduce the dynamic range of original signal. Therefore, the only basic problem for this system design is to calculate the coefficients of the colored filter. This can be solved by calculating Weiner-Hoff equation:

\[
a = R^{-1}r
\]  

(2.3)

where

\[
a = \begin{bmatrix} a_1 \\ \vdots \\ a_n \end{bmatrix}, \quad r = \begin{bmatrix} r_1 \\ \vdots \\ r_n \end{bmatrix}, \quad R = \begin{bmatrix} r_0 & \cdots & r_{n-1} \\ \vdots & \ddots & \vdots \\ r_{n-1} & \cdots & r_0 \end{bmatrix}
\]  

(2.4)

In this equation, \( a_n \) is the \( n \)th order coefficient of colored filter. Symbol \( r_n \) represents the \( n \)th order autocorrelation of original signal. The matrix \( R \) is a Toeplitz matrix.

2.1.3 Vector Quantization

Vector quantization (VQ) is a wide-used technique which has already been applied in hyperspectral image compression. The traditional way of scaler quantization can be generalized to
vector quantization. From the work of G.Motton and F.Rizzon [3], we can define the system for vector quantization:

A kind of transform $T$ is needed for mapping each vector to another one. In a compression sense, the total number of transformed vector should be far less than the original ones. So that, the transform $T$ should be a low rank matrix.

From above, VQ belongs to lossy image compression, and after quantization, the original signal can’t be perfectly recovered. Some information will be lost. Such that, the quantized vectors should be carefully selected to achieve the least information loss with competitive compression ratio.

### 2.1.4 Bitplane Coding

Bit-plane coding is a source-encoding technique designed and tested for use with certain kinds of data telemetered by scientific space probes [9]. A number of data is buffered and transmitted from most significant bit to least significant bit. This technique was applied in image transmission which encodes images bitplane by bitplane. In this way, the gray level of transmitted image will grow larger as the transmission process goes further. At a pixel level, as the transmission goes, the resolution of each pixel’s value is progressively becoming higher. Therefore, the traditional bitplane coding of transmission is SNR scalable.

Bitplane coding has also been applied in wavelet transform coding. In a transformed image process, the first bitplane is defined as the highest coefficient’s MSB. While comparing the bitplane coding in wavelet transform with that in traditional ones, the difference is that, the bitplane coding is only applied to MSB of each coefficient. In many wavelet transform coding technique, like SPIHT, EZW and SPECK, the process of coding MSB of each bitplane is called dominant pass. Except for the MSB, all or some of the other bits of each bitplane are transmitted directly and this process is called refinement (or subordinate) pass. The reason that caused this difference is the side information’s generation of some wavelet encoding techniques. As the bitplane approaching LSB, the bits in the whole image become very fussy and perform like an iid binomial process. This kind of source code will generate a longer compressed codeword than the original signal. As the transformed image has negative values, the sign should also be encoded into the embedded stream.


CHAPTER 2. REVIEW OF BACKGROUND IN SIGNAL CODING

2.2 Transform Coding

To take advantage of the correlation in a picture’s pixels, transform a image to another basis will be much easier for compression. Transform coding is nowadays the most wide-used way of compression. In most cases, the transform matrix \( T \) is a full-rank matrix. In other word, the transform is reversible. After transform, we can make the power of an image to be more concentrated in some basis, which can’t be better for compression. In this section, several common image transform will be introduced.

2.2.1 Discrete Cosine Transform

In digital image processing, a very common way to view property of an image is using Discrete Fourier transform. However, two deficiencies this approach will introduce:

- An image is consisted of real values, as a result, the transformed results will contain complex-values. This will burden compression system for calculating, representing and transmitting these kind of complex numbers.

- Implicit n-point periodicity of the DFT introduces boundary discontinuities which result in additional high-frequency components added to the original signal. ”After quantization, the Gibbs phenomenon will cause obvious block artifact” [10].

Therefore, DCT (Discrete Cosine Transform) was introduced which can overcome the deficiencies of DFT. The basis of DCT is to given in the following form:

\[
t_\omega = \alpha(\omega) \cos \left[ \frac{(2x+1)\omega \pi}{2N} \right]
\]  

\[\alpha = \begin{cases} \sqrt{\frac{1}{N}} & \omega = 0 \\ \sqrt{\frac{2}{N}} & \omega = 1, 2, 3..., N - 1 \end{cases}\]

From the formula above, the transform basis is shifted to the real domain, and the calculation will be simplified. Another change is that, the 2N-point periodic tapping method will make the boundary of each block smoother, which will reduce the block artifact. By doing projections, a function \( f(x) \) can be represented as the coefficients of a set of orthogonal discrete cosine functions, which is:

\[
C(\omega) = \langle f(x), t^*_\omega \rangle
\]
CHAPTER 2. REVIEW OF BACKGROUND IN SIGNAL CODING

and the form of inversed transform is:

\[ f(x) = \sum_{\omega=0}^{N-1} C(\omega)t_{\omega} \]  

(2.8)

The basis of DCT has a very strong correlation to the KL decomposition in the first-order Markov chain. The covariance matrix of the first-order Markov process is given by [11]:

\[
C = \begin{bmatrix}
1 & \rho & \rho^2 & \ldots & \rho^{n-1} \\
\rho & 1 & \rho & \ldots & \rho^{n-2} \\
\rho^2 & \rho & 1 & \ldots & \rho^{n-3} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
\rho^{n-1} & \rho^{n-2} & \rho^{n-3} & \ldots & \ldots
\end{bmatrix}
\]

(2.9)

where \( \rho \) is the correlation coefficient. For this matrix, when \( \rho \) is approximately 1, the eigenvectors of this matrix will be very close to the discrete cosine functions. Besides, most natural images perform like a first-order Markov process and their correlation is near linear. As a result, the DCT will be a suboptimal transform in de-correlation and compression.

In sum, DCT is still widely used in our common lives, the daily used standard JPEG is mainly based this technique. Though some flaws of this technique exist, like block effect in low SNR, it still performs well in many scenarios.

2.2.2 Discrete Wavelet Transform

Wavelet transform and its theory matured in recent twenty years. Unlike the Fourier transform only having frequency localization, the wavelet transform has both time and frequency localization property. A wavelet function should be zero-mean valued and supported in a limited range. Mathematically speaking, wavelet transform is a process to approximating a function by double-indexed wavelet functions \( \psi_{a,b}(t) \) in \( L^2 \) space:

\[
C_{a,b} = \langle \psi_{a,b}(t), f^*(t) \rangle = \int \psi_{a,b}(t) f^*(t) \, dt
\]

(2.10)

where the mother wavelet function is:

\[
\psi_{a,b}(t) = |a|^{-\frac{1}{2}} \psi \left( \frac{t - b}{a} \right)
\]

(2.11)

In this way, the original signal \( f(t) \) can be represented as a linear combination of dilated and shifted mother wavelet function \( \psi_{a,b}(t) \) and both time and frequency localization are achieved.
In Mallat’s multi-resolution theory for discrete wavelet transform, the original function can be decomposed by doing projections on the scaling function $\phi_{a,b}(t)$ and the mother wavelet $\psi_{a,b}(t)$. Once the scaling function $\phi_{a,b}(t)$ is decided, its counterpart, $\psi_{a,b}(t)$ can be calculated by doing projections on the scaling function family. In multi-resolution analysis, a functional space is divided into several laddered parts, the function $\phi_{a,b}(t)$ represents the low-resolution part of the original signal in this functional space. On the contrary, function $\psi_{a,b}(t)$ is the high-resolution part. The whole discrete wavelet transform can be represented as a process that keep splitting the finer component of the signal in its low-resolution part. The Figure 2.2 illustrates this process [12].

where $c_n$ represents the low-resolution part of the signal and $d_n$ represents the high-resolution part. The integer $n$ represents the function $\psi_{n,k}$ and $\phi_{n,k}$ dilated at a scale of $n$. A very simple case is Harr wavelet which is shown in Figure 2.3. Harr wavelet is the only linear phase orthogonal wavelet, which is also the simplest wavelet. However, Harr wavelet is not suitable for image compression since its step shape will causes the low regularity of this kind of wavelet. This means it’s more difficult for us to decompose an image with smooth edges using Harr wavelet.
CHAPTER 2. REVIEW OF BACKGROUND IN SIGNAL CODING

To achieve both linear phase and regularity, the bi-orthogonal wavelets were invented for image compression and the wide-known bi-orthogonal wavelets are CDF 9/7 and CDF 5/3 wavelets, which have already been applied in lossy and lossless image compression in JPEG2000 standard. In the later chapter, we will clarify how DWT can be implemented using filter banks for hyperspectral image compression.

2.2.3 KL Transform

In signal processing, “A signal can be represented in a set of statistically uncorrelated basis functions on its property of second order random signal” [13]. In this way, the signals can be transformed into a diagonal matrix which represents each uncorrelated basis in a power sense. This transform was firstly introduced by Karhunen and Loeve as the Hotelling transform. Assume that, a zero-mean vector \( x \) is a sampled signal needs to be transform. This transform can be represented in this way:

\[
\begin{align*}
R_x &= KDK^T \\
w &= K^T x
\end{align*}
\] (2.12, 2.13)

where \( R_x \) is the autocorrelation matrix of vector \( x \) and the matrix \( K \) stands for its eigenmatrix which is unitary. The orthogonal transform in (2.13) generates an uncorrelated vector \( w \) who has a zero mean with autocorrelation \( D \). In practical, \( x \) is assumed to be the original signal and the compression can be achieved by encoding \( w \) to \( \hat{w} \). At the receiver side, the recovered signal \( \hat{x} \) can be recovered by:

\[
\hat{x} = K \hat{w}
\] (2.14)

KL transform is the optimal transform in signal compression, however, its deficiency is also obvious. Because the KL transform is a data-based transform and the basis should be transmitted for every image. Another problem is the high computational complexity caused by its basis computation every time. In sum, KL transform is the best transform theoretically, however, hard to be applied in practical use.

2.3 Wavelet-based Coding techniques

After performing isotropic DWT to an image, the transformed data will have a pyramidal structure. For natural images, the coefficients have correlation between each subbands, and this can
be used to compress these coefficients. The following techniques based on DWT exploited this kind of correlation, which showed how powerful the DWT is in image compression.

Two main techniques are introduced here, which are EZW and SPIHT. In EZW, the tree structure for compression was first raised. Later, SPIHT was invented to improve the compression ration of EZW which gives a more refined way to output compressed symbols.

### 2.3.1 Embedded Zerotree Wavelet (EZW)

EZW algorithm is firstly introduced by Shapiro in his epoch-marking paper [14]. This paper showed how powerful the DWT is in image compression. EZW stands for Embedded Zero Tree coding. By taking advantage of the multi-resolution structure in different level of wavelet transform, through bitplane coding, the transformed coefficients can be highly compressed.

This algorithm introduced tree structure in image coding. In a tree structure, the pixels in the similar orientation to represent same place in a picture will be grouped as a tree. The Figure 2.4 showed this relation:

![2-Level Wavelet Decomposition](image)

Figure 2.4: 2-Level Wavelet Decomposition

In an image, separable two-dimensional DWT is applied. As a result, after two times downsampling, each pixel in the lower frequency band will have four children in the higher band
CHAPTER 2. REVIEW OF BACKGROUND IN SIGNAL CODING

in the same frequency direction (except for the highest frequency subbands). In EZW algorithm, bitplane coding technique is applied. The coding order is from MSB (Most Significant Bit) to LSB (Least Significant Bit). For each bitplane, all the zeros are coded in a tree structure. The EZW algorithm takes advantage of the correlation between wavelet coefficients from different subbands who represent the same place in the original picture. It used the value of a tree root to predict its leaves. If in one typical bitplane, the root and its leaves are all 0, then, a bunch of all these coefficients in this tree will be coded into only one symbol. There are four symbols in EZW algorithm to represent the pixels in each bitplane. They are: \( P \) (Positive Significant), \( N \) (Negative Significant), \( I \) (Isolated Zero) and \( R \) (Zero Tree Root). After zero tree coding, all these symbols are compressed again in entropy coding for further transmission.

2.3.2 Wavelet Difference Reduction (WDR)

In this algorithm, each pixel are assigned a position number in a baseline scan order. In such way, the significant bits of each bitplane can be represented in position numbers and the compression process is performed on these numbers. For natural images, the large coefficients always concentrate in low frequency bands and are close to each other. Such that, to reduce the dynamic range of position number for a better compression result, only the difference of position numbers are encoded. WDR algorithm has a much lower computational complexity than EZW and achieves a better visual effect under the same compression ratio. This method is more commonly used in the areas like underwater communication where the transmission speed is limited.

2.3.3 Set Partitioning In Hierarchical Trees (SPIHT)

The SPIHT algorithm was introduced by Said and Pearlman [15]. It is very similar to the EZW algorithm, however, this algorithm discards the notion of Zero Tree, It uses a hierarchical tree structure. Because SPIHT is the algorithm that we used in our compression system, some more details of this algorithm are introduced here.

SPIHT coding is based on tree structure of each bitplane. A typical \( 8 \times 8 \) image wavelet coefficients matrix (left) and its MSB map (right) is shown in Figure 2.5.
CHAPTER 2. REVIEW OF BACKGROUND IN SIGNAL CODING

This 8 × 8 image is decomposed at the lowest frequency level of 3 and is divided into 3 independent groups, which are marked by the same color. In this matrix, the highest value is 63. Therefore, 6 bitplanes are needed to represent this matrix. In this bitplane, any coefficients whose absolute values between 32 and 64 are represented as 1 for positive values and −1 for negative values. These kind of coefficients are called significant coefficients. All other values are represented as 0. The SPIHT algorithm can be viewed as testing and splitting of the independent (colored) groups.

If all values in a group are 0 then the output is 0, otherwise it is ‘1’. If the output is 0, then the test of this group is finished, because this group can be represented using only one symbol labeled as 0, as shown for the yellow colored group in Figure 2.5. On the contrary, the group output is nonzero then that group will be split into 5 independent subgroups to be tested. The first four subgroups contain only one pixel located in the lower frequency subband. The last remaining subgroup is the union of pixels whose parents are the pixels in the first 4 subgroups.

For example, in the green group’s bit map, the group contains non-zero values (except R). Then, 1 is outputted and this group is split into 5 subgroups. These subgroups are:

- Pixel: $r_1$;
- Pixel: $r_2$;
- Pixel: $r_3$;
- Pixel: $r_4$;
- Pixels:
  Offspring of $r_1$: $d_1$, $d_2$, $d_5$, $d_6$;

Figure 2.5: 8 × 8 Block Coefficients and MSB Map
CHAPTER 2. REVIEW OF BACKGROUND IN SIGNAL CODING

Offspring of $r_2$: $d_3, d_4, d_7, d_8$;
Offspring of $r_3$: $d_9, d_{10}, d_{13}, d_{14}$;
Offspring of $r_4$: $d_{11}, d_{12}, d_{15}, d_{16}$;

Then a set of new tests will be applied to these subgroups. Here we found that, $d_4$ is 1. Hence, symbol 1 is the output of the last subgroup. Because the last subgroup is in the highest frequency level, this subgroup will only be divided into four subsubgroups based on the pixels in this subgroup of their parents and another test will be applied. These four divided subsubgroups are:

- Pixel: $d_1, d_2, d_5, d_6$ (Offspring of $r_1$);
- Pixel: $d_3, d_4, d_7, d_8$ (Offspring of $r_2$);
- Pixel: $d_9, d_{10}, d_{13}, d_{14}$ (Offspring of $r_3$);
- Pixel: $d_{11}, d_{12}, d_{15}, d_{16}$ (Offspring of $r_4$);

This procedure will repeat until the output is 0 or the groups can’t be divided anymore.

The procedure that we referred above is called dominant pass in SPIHT coding. Next, the significant coefficients will be sent to the refinement pass for a finer quantization in order to achieve a better recovery in which the quantization bits are adaptive.

2.3.4 EBCOT & JPEG2000

EBCOT stands for Embedded Block Coding with Optimal Truncation, which was firstly introduced by David Taubman in his work. He opined that, in a high performance scalable image compression system, an image can achieve scalability in both SNR and resolution by enabling efficient one-pass rate control and feature-rich bit-stream [16].

EBCOT tiles an image into relatively small code block (typically $32 \times 32$ or $64 \times 64$) for each subband [17]. Then each block is independently encoded into highly scalable bit-stream. For a compression purpose, almost every bit-stream of each block is truncated in a length of $L_i$ at a distortion $D_i$. To achieve the best Rate-Distortion compression ratio for a given size, each code-block is truncated at an optimal truncation point. In the end, the compressed data are series of bit-stream with a large number of truncated points. The rate distortion optimization algorithm is called PCRD (post-compression rate-distortion). It assumes that the Rate-Distortion function performs like a conventional convex hull function and finds the best truncation points by calculating the distortion-rate slopes, which should be strictly decreasing. The JPEG2000 standard is almost
CHAPTER 2. REVIEW OF BACKGROUND IN SIGNAL CODING

based on EBCOT. The difference is that, the JPEG2000 standard enhances the compression speed at a cost of relatively low SNR performance by using a fast but less optimal arithmetic encoder, which reduces the code block size by reducing the fractional bitplane path [17].

2.4 Proposed Algorithm

Recent works in hyperspectral image compression paid much attention to the modification of decomposition levels for each dimension and the patronized tree design. However, in our algorithm, we refocused on the classic DCT, which we claim is better for hyperspectral image compression. In the compression stage, combination of traditional DCT with 3D-SPIHT coding outperforms the DWT with 3D-SPIHT coding in most situations. This superior performance over traditional DCT-based JPEG algorithm is obtained carefully choosing block-size and level-shifting.

In this thesis, we propose a novel architecture for solving the AVIRIS hyperspectral image compression problem. In our approach, a hyperspectral image cube is first tiled into many small cubes on which the subsequent encoding and decoding techniques are applied. Then, These include 3D-DCT transformation, EBCOT-based quantization and 3D-SPIHT algorithm to fulfill our goal.

Because all the operations are based on independent cubes, parallel computing can be applied in this system. The details of this system architecture are described in the next chapter. To achieve these goals, we have specified and solved the problems of 3D-SPIHT generalization including the choice of code cube size.

2.5 Summary

In this chapter, we briefly reviewed some techniques that have already been applied in media compression. Some of the wavelet based compression algorithms are specified here which are the basis of this thesis. In practical implementations, a typical encoder/decoder implements not just one algorithm but a combination of several of them. Usually, the entropy coding and transform coding always work together to form the main part of the encoder and this structure is also applied in our proposed system.
Chapter 3

Architecture Details

In Chapter 2, the 2D-SPIHT algorithm is introduced. For multi-dimensional images in HSI data cube, a three-dimensional form of SPIHT (3D-SPIHT) algorithm is described in this chapter. Here, we specified the details of the 3D-SPIHT algorithm and the tiling strategy for DWT and DCT cube transform in AVIRIS data. Some basic performance evaluations of the SPIHT algorithm are also introduced in this chapter, which will help us analyze the compression results in Chapter 5.

3.1 3D-SPIHT

Xiaoli and William in their work \cite{18} firstly presented a new 3D-SPIHT and 3D-SPECK algorithm for hyperspectral images and showed good results using wavelet transform.

In this section, we introduce this 3D-SPIHT algorithm, which is generalized from traditional 2D-SPIHT algorithm. We first discuss how a code cubes is set up along with parent-children relationship between different subbands. We then present the main body of this algorithm.

3.1.1 Code-Block Introduction

From what we introduced in Chapter 2, the hyperspectral data is organized in a three dimensional cube structure. Therefore, the coding unit is based on a three dimensional cube. Here, we call the smallest coding as unit code cube. An entire hyperspectral image is tiled into many cubes and all these cubes can be encoded independently. After three-dimensional dyadic DWT, the code-cube will be divided into a 3D-pyramidal structure, which is in a frequency domain from lowest subband to highest subband. The level of the pyramid is decided by DWT decomposition level. The
roots of this code cube are located at the lowest frequency subband, which represents the coarse part of an image. The most detailed part of an image is located in the bottom part of the cube. Figure 3.1 showed this structure of code-cube.

From Figure 3.1, we notice that the structure is an extension of the traditional 2D-DWT, which divides a band into LL (vertical-lower subband, horizontal-lower subband), LH (vertical-lower subband, horizontal-higher subband), HL (vertical-higher subband, horizontal-lower subband) and HH (vertical-higher subband, horizontal-higher subband). In 3D-DWT for hyperspectral data, a new spectral dimension is added. As a result, a specific band is divided into 8 subbands after DWT. These bands are labeled as: HHH, HHL, HLH, HLL, LHH, LHL, LLH, LLL. The third symbol: H or L represents higher or lower bands respectively in spectra. To specify this, a third argument, a coordinator is set up for such a code cube, and each DWT coefficients in this cube is indexed by three arguments. For example, the coefficient $c_{x,y,z}$ is located at the $(x, y, z)$ point in the frequency domain.

In 3D-SPIHT, each pixel in LLL subband (except for the pixels in the highest frequency band) have 7 direct-children, which belong to HHH, HHL, HLH, HLL, LHH, LHL, LLH subbands separately. Some lower bands will have grandchildren. If all the pixels in a set originate from
one-pixel: Root, then these pixels are called the offspring of Root. This is illustrated in Figure 3.2.

![Figure 3.2: 3D-Hierarchical Tree Organization](image)

### 3.1.2 Algorithm

The implementation of 3D-SPIHT algorithm is straightforward from the traditional 2D-SPIHT algorithm [15]. The 3D-SPIHT algorithm maintains three-link list to track pixels in each cube as the 2D-SPIHT. These lists are defined as:

- **LIS**: List of Insignificant Set. This list records the pixels and their offspring that may need to be quantified in the further bitplane.
- **LIP**: List of Insignificant Pixels. This list records the single pixels that need to be quantified in the further bitplane.
- **LSP**: List of Significant Pixels. This list records the single pixels that need to be encoded and quantified in the current bitplane.
CHAPTER 3. ARCHITECTURE DETAILS

To develop these list a judgment function is defined below:

\[
J(\vec{I}) = \begin{cases} 
1 & \text{if } \max_{c_{x,y,z} \in \vec{I}}(|c_{x,y,z}| \geq T) \\
0 & \text{if } \max_{c_{x,y,z} \in \vec{I}}(|c_{x,y,z}| < T) 
\end{cases}
\]  

(3.1)

where \( T \) represents some predefined threshold. Also a new sign function is defined, which maps values to two symbols:

\[
\text{sgn}(c_{x,y,z}) = \begin{cases} 
+ & \text{if } c_{x,y,z} \geq T \\
- & \text{if } c_{x,y,z} < -T 
\end{cases}
\]  

(3.2)

In bitplane coding, \( T \) is commonly chosen as \( 2^n \) (where \( n \) is a positive integer). The set \( \vec{I} \) consists of values \( c_{x,y,z} \) from different locations in a cube.

For representing the hierarchical trees by tracking their root pixels, we define the following sets:

- \( \mathcal{O}(c_{x,y,z}) \): Set contains all the offspring of \( c_{x,y,z} \).
- \( \mathcal{C}(c_{x,y,z}) \): Set contains all the direct-children of \( c_{x,y,z} \). For convenience, we refer to this set as type B set.
- \( \mathcal{G}(c_{x,y,z}) \): Set contains and all the grandchildren of \( c_{x,y,z} \) which is \( \mathcal{O}(c_{x,y,z}) - \mathcal{C}(c_{x,y,z}) \). We refer to this set as type A set

A pixel is called significant if its transformed value is larger than the threshold \( T \), otherwise, it is called insignificant. If the MSB of this transformed value is 1 in its bitplane, the pixel should belong to significant value set and vice versa. Before the algorithm begins, the link-list and the threshold values are initialized as:

\[
T \leftarrow 2^{\lceil \log_2(\max(\mathcal{O}(c_{0,0,0})) \rceil)}
\]

(3.3)

\[
\text{LIP} \leftarrow H
\]

(3.4)

\[
\text{LIS} \leftarrow H(\text{type } A).
\]

(3.5)

The algorithm in Dominant Pass is given in Algorithm 1.
CHAPTER 3. ARCHITECTURE DETAILS

After dominant pass, all the values in the LSP are quantified in some bit-depth on the required reconstruction quality. This procedure is termed as the subordinate path which is given in Algorithm 2.

From Algorithm 1, every time the threshold $T$ refreshes, a new coding path is created on LIS. The procedure that completes encoding one bitplane is called a stage of the SPIHT algorithm, which finishes line 3 to line 34 in Algorithm 1. In each stage, the coefficients whose value are larger than the threshold are coded and quantized. If the transform is an orthogonal transform, then at the decoder side, the error between original image and recovered image will become smaller and smaller, resulting in larger SNR values. Therefore, the SPIHT algorithm is a SNR scalable algorithm which progressively enhances the SNR of a recovered image through the transmitted bitstream. From the feature of Rate-Distortion curve in Section 1.2.3, we can infer that, the Rate-Distortion curve of SPIHT algorithm will increase dramatically during the very first stages. However, as the threshold becomes smaller, enhancing the compression ratio will be much harder than to achieve than during the first several stages.

3.2 SPIHT Performance Evaluation

In this section, we simplify some coding cases and analyze the performance of 2D-SPIHT coding algorithm mathematically. The only difference between the 2D and the 3D SPIHT algorithm is just the number of children of each root. Therefore, the conclusions we derived in this section are also applicable in the 3D-SPIHT which will help us better analyze the results on the SPIHT coding performance in different scenarios.

As the coding cost in dominant pass contributes greater than the subordinate pass, the codeword we discussed here indicates that in the dominant pass only. The performance of the SPIHT coding wholly depends on the arrangement of different bitplanes. We give the following definition to form a general case:

Definition 3.1. A finite-ordered sequence of sets on one dimensional space is labeled as:

$$S_1, S_2, S_3, ..., S_k$$

and all these sets should satisfy: There always exists an onto function $f_j$, such that,

$$f_j : S_j \rightarrow S_{j+1} \oplus S_{j+2} \oplus S_{j+3}... \oplus S_k$$

(3.6)

where $\oplus$ is the symbol of direct sum.
CHAPTER 3. ARCHITECTURE DETAILS

The hierarchical trees in wavelet coding can be generated from these sets and the mapping rule. In hierarchical trees, all the transformed coefficients in a specific level of subband can be grouped in a set labeled as $S_i$. The index $i$ indicates which level of subband these transformed coefficients belongs to. The mapping rule illustrates the relationship between different roots and descendants. With these sets and mapping rules, the tree structure can be transformed into onto functions between several ordered sets. For instance, in a 2D-DWT image (all the word: image in this section is defined as transformed image), the coefficients in the lowest frequency level are marked as the elements in $S_1$, and the coefficients in the higher frequency level belong to $S_2$ and so on.

Here we define that, $S_i$ only contains all the coefficients’ absolute value in $i$th level of DWT. The integer $n_i$ indicates the number of elements in set $S_i$ and $k$ is the maximum level of DWT that is used. All the elements in the lower level are mapped to the elements in higher level which are in the same direction in DWT. Figure 3.3 showed the patronizing rule of these coefficients. For an image with non-zero power, the best case for the SPIHT coding is that, except for the elements in $S_1$ and $S_2$, all the elements in other sets are zero-valued. From the SPIHT algorithm we referred above, $T$ is the number of symbols for encoding, which can be easily calculated as:

$$T = (k + 1)n_2 + n_1$$  \hspace{1cm} (3.7)

where $k$ is the max number of bitplanes we need to compress.
CHAPTER 3. ARCHITECTURE DETAILS

Now, let’s consider a worse case: All the sets have a random value for each element, however, they also satisfied the following rule:

For any elements $e_i$ in $i$th set $S_i$ (except for elements in $S_1$),

$$2^{\lfloor \log_2 \min(S_{i-1}) \rfloor - 1} < e_i \leq 2^{\lfloor \log_2 \min(S_{i-1}) \rfloor} \tag{3.8}$$

For encoding all the coefficients in SPIHT algorithm, a number of:

$$T_k = 5n_2 + \sum_{i=3}^{k-2} 5n_i + 2n_{k-1} + n_k + n_1 \tag{3.9}$$

symbols are needed at most. For convenience, this situation is termed as: Case A. We can prove that the coding length for a single element in LIS from $S_n$ can be represented in a recursive function:

$$Y[i] = 4Y[i+1] + 5 \tag{3.10}$$

where $2 \leq i < k - 1$ and $Y[k - 1] = 6$.

For most natural images, the coefficients’ value decreases very fast in the first several levels of subbands. In these subbands, most of the coefficients obey the formula in 3.8 and the length of codeword in these subbands is very short which is still acceptable when compared to the best case.

Now, the situation in the last several subbands becomes worse. The coefficients in the higher frequency bands have higher values than the lower frequency bands in some cases. Consider this kind of situation: In an SPIHT coding process, all the elements’ values are the same as that in Case A, except for one element $e_{ck}$ in $S_k$ (the highest frequency subbands), whose value is the same as someone in $S_1$ (the lowest frequency subbands). For convenience, we termed this situation as: Case B. To better represent the encoding cost, we introduce the following recursive function:

$$X[i] = 3Y[i+1] + X[i+1] + 4(i+1) + 2 \tag{3.11}$$

where $2 \leq i < k - 1$ and $X[k - 1] = 3k + 2$.

In formula 3.11, $X[n]$ stands for the coding length for encoding a single element $e_i$ in LIS from $S_i$, and $e_{ck} \in f_{ik}(e_i)$ (where $k$ is the maximum level of DWT that is used).

Now, let’s consider the difference on coding length for these two different cases:

$$X[i] - Y[i] = X[i+1]Y[i+1] + 4(i+1) - 3 \tag{3.12}$$

If we define: $C[i] = X[i] - Y[i]$, formula 3.12 can be written in the following form:

$$C[i] = C[i+1] + 4(i+1) - 3 \tag{3.13}$$
CHAPTER 3. ARCHITECTURE DETAILS

where $1 \leq i < k$, therefore $C[1] > 0$ and sequence $C[i]$ shall be a positive-increasing sequence. From the result above, the encoding cost in Case B is higher than that in Case A.

Now, let’s take Case C into consideration. In Case C, the situation becomes worse. We assume that, there is another element $c_{ck}^\prime$ in $S_k$ which shares the same value as $c_{ck}$. For some positive integer $m$, there exist two different elements $c_m, c_m'$ in $S_m$ and an element $c_{m-1}$ in $S_{m-1}$, such that $c_{ck} \in f_{mk}(c_m), c_{ck}^\prime \in f_{mk}(c_m'), c_{ck}^\prime \in f_{m-1k}(c_{m-1})$ and $c_{ck} \in f_{m-1k}(c_{m-1})$. In traditional SPIHT image coding, the integer $m$ is a measurement of distance in space between $c_{ck}$ and $c_{ck}^\prime$, which indicates the level where the two coefficients were separated into two trees.

In Case C, the coding length $Z[i]$ can be represented in the following recursive function set:

$$
\begin{align*}
Z[i] &= 3Y[i+1] + Z[i+1] + 4(i+1) + 2 & i < m \\
Z[i] &= 2Y[i+1] + 2X[i+1] + 8(i+1) + 4 & i \geq m
\end{align*}
$$

(3.14)

The difference on coding length between Case B and Case C shall be:

$$
\begin{align*}
Z[i] - X[i] &= Z[i+1] - X[i+1] & i < m \\
Z[i] - X[i] &= C[i+1] + 4(i+1) + 2 & i \geq m
\end{align*}
$$

(3.15)

where $C[i+1] > 0$, $i > 1$. The function is a positive-increasing function. Therefore, the coding cost in Case C is higher than that in case B, and smaller the $m$ higher the coding cost we have. In an image compression, it will be better for the SPIHT coding if the large valued coefficients in same frequency bands are concentrated in space. For a more general case, if there are more elements like $c_{ck}$ that exist in the highest frequency level, the recursive function set can be written in:

$$
Z[i] = (4 - r)Y[i+1] + rX[i+1] + r(i+1) + 2r & m_{r+1} > i \geq m_r
$$

(3.16)

where $r$ is an integer ranging from 1 to 4, which indicates the number of significant values of an element’s children.

For some more general cases, these recursive functions can be applied to the roots in LIS respectively and calculation of the codeword length will become more complex. However, the following conclusions are obvious:

1. Lower values in high-frequency bands results in better compression performance.
2. Many large-valued coefficients group as clusters in the same subbands results in better compression performance than which scatter in the same subbands.

These conclusions will be used in the following analysis in Chapter 5.
CHAPTER 3. ARCHITECTURE DETAILS

3.3 Cube Transformation & Organization

For the purpose of compression, two different kinds of transforms are selected. In this section, we firstly introduced how the tiled blocks are transformed. Then, we talked about how to tile the blocks in a spectral sense.

3.3.1 3D-DCT

The transformation of a single block in DCT is very simple, which can be derived from the one dimensional form. The transformed coefficient $I(\omega, \mu, \gamma)$ in 3D-DCT formula is shown below:

$$I(\omega, \mu, \gamma) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} \sum_{z=0}^{S-1} I(x, y, z) \alpha(\omega) \alpha(\mu) \alpha(\gamma) \cos \left( \frac{2x + 1}{2M} \omega \pi \right) \cos \left( \frac{2y + 1}{2N} \mu \pi \right) \cos \left( \frac{2z + 1}{2S} \gamma \pi \right)$$ (3.17)

where $M, N, S$ represent the size of an image and $I(x, y, z)$ is the pixel’s value at point $(x, y, z)$ in the whole image cube. The $\alpha$ function is the same as that in 2.6.

3.3.2 3D-DWT

The DWT provides a way to decompose an image into multi-resolution subbands for analyses in a pyramidal structure. The signals are processed by passing through a tree-structured filter banks who can perfectly reconstruct decomposed signals. It can be proved that the DWT process in any levels can be represented in a multi-rate filter banks with finite impulse response (FIR) filters [12]. Empirically speaking, many regions in subbands may share some similar patterns that corresponding to the same place in the original image. The multi-resolution filter banks is shown in Figure 3.4.

In Figure 3.4, a two levels of tree-structured filter banks is applied to a one-dimensional signal $x[n]$. The signal $x[n]$ is passed through a high-pass filter (HPF): $h_0[n]$ and a low-pass filter (LPF): $h_L[n]$. The HPF separates the high-frequency component $y_H[n]$ from original signal $x[n]$, and the LPF separates the low-frequency component $y_L[n]$ from signal $x[n]$. After downsampling by 2 in the highest frequency level, the high-frequency signal $y_H[n]$ is passed to the next module for further operation. However, the coarse signal $y_L[n]$ is passed through the same filter banks again, which are $h_0[n]$ and $h_L[n]$, to generate the high-frequency and low-frequency part of the signal $y_L[n]$. After downsampling, the 2-level decomposition of signal $x[n]$ by DWT is finished and the reconstruction part is just the inverted structure of the decomposition part.
CHAPTER 3. ARCHITECTURE DETAILS

To generalize one-dimensional DWT to three-dimensional DWT, one simple approach is to decompose the signal in each dimension separately. In this approach, all the signals in a horizontal direction is processed through one-dimensional filter banks, and then, the processed signal in a vertical direction is passed through the same filter banks. At last, the same decomposition process is applied in the spectral direction individually. Figure 3.5 showed this process. Figure 3.6 represents this process into a filter bank form. In this figure, filter $H_{0,0,0}$ is a three-dimensional filter, which is the tensor product of three low pass filter $h_0$ and we can deduct the other three-dimensional filters from this one. For example, filter $H_{0,1,1}$ is the tensor product of filter $h_0$ in horizontal direction, filter $h_1$ in the vertical and spectral direction. The three-dimensional filter can be represented in the following formula:

$$H_{i,j,k} = h_i \otimes h_j \otimes h_k$$  \hspace{1cm} (3.18)$$

$$G_{i,j,k} = g_i \otimes g_j \otimes g_k$$  \hspace{1cm} (3.19)$$

where $\otimes$ is the tensor outer product and $i, j, k = 0 \text{ or } 1$. In such way, each level of decomposition will have 8 subbands.
CHAPTER 3. ARCHITECTURE DETAILS

3.3.3 DWT Cube Organization

For a typical AVIRIS data, the tiling procedure in spatial domain won’t be the main problem that we need to take into consideration since each image may contain thousands of lines and samples in the spatial domain. However, the number of pixels in the spectral domain is quite limited when compared to the spatial domain, which is only 224. Therefore, the tiling process in space has more freedom and how to patronize the 224 bands into different band sets will directly affect the performance of compression.

The number 224 can be represented as $7 \times 2^5$, which is not in a form of $2^n$. As a result, to keep the length of each subband in spectral domain is decreasing by a factor of 2, the DWT can only be performed on the whole spectral domain for 5 times at most. Therefore, to simplify the hierarchical tree organization, a very naive approach is to choose an image in a size of $(2^5 \times X) \times (2^5 \times Y) \times 224$ (Where $X$ and $Y$ are positive integers), and the DWT should be performed on this cube at most 5 times using periodic mode. In this mode, the DWT procedure can make the number of pixels per subband per axis decreasing at a factor of 2, which is accordance with the hierarchical tree organization rule. Figure 3.1 showed this kind of cube organization.

Another way to patronize the image is to divide the 224 bands into $7 \times 2^m$ band sets, and
each band set will have $2^{5-m}$ bands, where $m$ is a positive integer. To make a better comparison with DCT transform and keep each band set have a same number of bands, the integer $m$ should only vary from 0 to 2. In the spatial domain, the number of samples and lines keep the same as the bands in spectra. Figure 3.7 showed this kind of patronization. In Figure 3.7 the left one was tiled in a larger size of code cube than the right one for a same HSI data cube.

Figure 3.7: Cube Tiling Strategy

### 3.3.4 DCT Cube Organization

The implementation of DCT cube organization is quite simple, which just needs to generalize the DCT block-tiling process to three-dimensional cube-tiling process. The traditional DCT tiling strategy is to tile an image into $8 \times 8$ small blocks for transformation, which has been already used in the JPEG standard. The main consideration for tiling size is the computational complexity (The traditional DCT computational complexity is $O(N^2)$, while using the Fast Cosine Transform algorithm is $O(N \log N)$). In our compression system, $8 \times 8 \times 8$ and $16 \times 16 \times 16$ cubes are used for DCT. The larger cube size can also be made. However, to reduce the computational complexity, the largest code cube is only $16 \times 16 \times 16$ cube.

### 3.4 Level Shifting

In the JPEG standard, a Level Shifting procedure is applied in the preprocessing stage. For an 8-bit image, this can be done by shifting the value by one bit, which is a $-128$ bias on the original
signal. This procedure removes the DC component and converts the original signal from unsigned values to signed values, which makes the transformed values around zero.

In our work, we found that the level shifting operation vitiate the compression results for hierarchical tree coding. The reason is the level shifting breaks the well-organized transformed values from low-frequency subbands to high-frequency subbands in a decreasing order if the mean is very large.

For a small-valued mean, this effect is obvious in the high-frequency subbands, especially for some small values. Figure 3.8 showed how this procedure brake the well arrangement of the hierarchical tree. In Figure 3.8, the last 100 values with and without level shifting in subband HHH are arranged in a three-dimensional baseline scan order, which arranges all the values in a cube from low-frequency subbands to high-frequency subbands.

The performance of an SPIHT coding algorithm is based on its coefficients arrangement or power concentration. In a pyramidal structure, we need the coefficients to be arranged in a decreasing order from low-frequency subbands to high-frequency subbands. In Figure 3.8, many coefficients in the higher frequency subbands have much larger values than the ones without mean elimination. This greatly vitiates the compression performance of SPIHT coding algorithm.

Figure 3.8: Last 100 Coefficients with (and without) Level Shifting
CHAPTER 3. ARCHITECTURE DETAILS

3.5 Summary

In this chapter, we introduced the 3D-SPIHT algorithm from the pyramidal structure of a transformed cube. After defining the hierarchical tree relationship between each coefficient, the 3D-SPIHT process can be easily derived. The cube tiling rules are also introduced here. We intentionally spared a section to discuss how the performance of the SPIHT algorithm is affected by the arrangement of different coefficients. As a result, for a better compression ratio, we want the large coefficients to be concentrated at the lower frequency bands amongst different subbands. We also want the large coefficients can be concentrated as clusters in each subband. At last, we showed that the level shifting process is not needed in our algorithm which is different from the JPEG standard.
CHAPTER 3. ARCHITECTURE DETAILS

Algorithm 1 Dominant Pass

1: while $T > T_c$ do
2:     $T = T/2$
3:     for every $c_{x,y,z} \in LIP$ do
4:         output $sgn(c_{x,y,z})$
5:     if $J(c_{x,y,z}) \neq 0$ then
6:         LSP $\leftarrow c_{x,y,z}$
7:     end if
8:     end for
9:     for every $c_{x,y,z} \in LIS$ do
10:        if $c_{x,y,z}$ is type A then
11:           Output $J(\cup(c_{x,y,z}))$
12:           if $J(\cup(c_{x,y,z})) \neq 0$ then
13:              for every $c_{x,y,z} \in C(c_{x,y,z})$ do
14:                  Output $sgn(c_{x,y,z})$
15:              end for
16:           else
17:              LIS $\leftarrow c_{x,y,z}$
18:           end if
19:        end if
20:        if $G(c_{x,y,z}) \neq \emptyset$ then
21:           set $c_{x,y,z}$ as type B, move $c_{x,y,z}$ to then end of LIP
22:        else
23:           remove $c_{x,y,z}$ from LIS
24:        end if
25:     end if
26:     else
27:        Output $J(G(c_{x,y,z}))$
28:        if $J(G(c_{x,y,z})) == 1$ then
29:           LIS $\leftarrow C(c_{x,y,z})$
30:           Delete $c_{x,y,z}$ from LIS
31:        end if
32:     end if
33: end if
34: end for
35: end while
CHAPTER 3. ARCHITECTURE DETAILS

Algorithm 2 Subordinate Pass
1. for every $c_{x,y,z} \in LSP$ do
2. Quantizer $\leftarrow c_{x,y,z}$
3. Output quantized value: $Q$
4. end for
Chapter 4

Practical Implementations

In Chapter 3, we discussed the preprocessing approaches, which includes the block arrangement, multidimensional transformation, and coding algorithm. In this chapter, we first introduce some of the problems that we encountered in implementations and how we solved these problems. Next, we introduce how the parallel computing is implemented in our system to enhance the compression speed. Finally, the PCRD algorithm which can enhance the compression performance by balancing the coding stage of each band sets is presented in detail.

4.1 Compression System

For implementation, the original image is tiled and transformed (this order may be order changed between these two steps depending on whether a large or small cube is compressed). Next, the SPIHT coding process is applied to the transformed coefficients in each code cubes. Finally, the encoded codewords are compressed again using arithmetic coding. The process is shown in Figure 4.1 for large code-cubes and in Figure 4.2 for small code-cubes:

![Diagram of Compression System](image)

Figure 4.1: Large Code Cube Compression System
In our compression system, there is a difference between the large code cube compression and the small code cube size compression system. In a large code cube compression system, the tiling process comes after the computation of the 3D-DWT. However, in small code cube size system, the tiling process is implemented before the transform process. The reason for this design is that a large code cube can take advantage of the parallel computing in this structure, which can dramatically enhance the compression speed.

In practical implementations, if a compression system or computer has multi-encoding units or cores (like GPU), we can distribute the code cubes to each of them for processing. Here is an example. Assume that an HSI data cube with a size of $32 \times 32 \times 32$ is needed for compression at a code cube size of $16 \times 16 \times 16$ using DWT transform. In this situation, the cube can be divided into 8 code cubes, and these code cubes are distributed to each core for processing. Figure 4.3 shows this process.
CHAPTER 4. PRACTICAL IMPLEMENTATIONS

However, for example, if we perform the DWT on the whole 512×512×512 image, the 3D-SPIHT encoding process will be performed on this large code cube with only one encoder, which makes the processing speed very slow. Fortunately, the SPIHT algorithm is based on hierarchical trees. The large code cube can still be divided into independent hierarchical trees for compression, which makes the parallel computing for large code cubes available. We discuss this in the following section.

After encoding, the encoded codewords are sent to an arithmetic encoder for further compression. For the SPIHT algorithm in dominant path, there exist four symbols in the encoded words, which are +, −, 0, and 1. In subordinate path, the significant values are quantified into binary numbers, which can be assumed to be symbol, 0 and 1 in dominant pass. As a result, for an arithmetic encoder, only four symbols need to be encoded. In our system, the arithmetic encoding process is performed on each code cube. However, in practical implementation, the adaptive arithmetic encoder can also be used to enhance processing speed at a cost of some compression ratio loss.

In our compression system, the bitstream is in a cube order, and in each bitstream of a code cube, the coefficients are transmitted from large values to small values each stage. The bitstream is shown in Figure 4.4.

At the receiver side, the bitstream’s transmission can be stopped at any point, which means that the transmission can be stopped at any stage in any code cube. Hence, the organization of this bitstream is SNR scalable as well as spatially and spectrally accessible. In this way, the receiver can have more freedom to balance the quality of recovery in each location in order to enhance the transmission efficiency. In some situations, the absorption bands have little effect on the analysis of the objects’ spectral signature. Therefore, the quality of these recovered bands can be sacrificed to achieve a better compression ratio. On the other hand, if some regions of interest in an HSI need a better quality of recovery, and other regions are also needed as a reference, the encoder can just encode the region of interest in a relatively higher SNR than others.

Figure 4.4: Bitstream design Whole Image
4.2 Post Compression Rate Distortion (PCRD) Algorithm

The PCRD algorithm is developed to optimize the Rate-Distortion ratio by choosing the optimal cutting points. The mathematical model of this algorithm was firstly introduced by Hugh Everett in his paper, who used a generalized Lagrange multiplier methods for solving this problem [19].

For a simplified presentation, we ignore the size of each cube and their contributed difference to the whole image. We denote, the code length of the whole image after compression by $L$, and the length $i$ for each cube by $L_i$. As a result, the code length of the whole image can be represented as:

$$L = \sum_i L_i$$  \hspace{1cm} (4.1)

In our algorithm, we can get different code lengths at different distortions since the SPIHT algorithm is SNR scalable. If for some reasons, we want the whole image after compression to be less than some values of length $L_{\text{max}}$, then the code length of different cubes in the whole image should be truncated at some truncation point $c_i$. The code length of the image can now be represented as:

$$L = \sum_i L_{c_i,i} \leq L_{\text{max}}$$  \hspace{1cm} (4.2)

where $L_{c_i,i}$ is the coding length of $i$th block truncated at $j$th point $c_{i,j}$.

The other factor that we need to take into consideration is distortion. At a compression and coding stage, we assume that the transform is orthogonal. So that, the distortion can be directly calculated from the difference between encoded transformed coefficients and the original ones. In our system, it’s obvious that DCT is an orthogonal transform. Although some wavelets, for example, bi-orthogonal wavelets CDF 9/7 and CDF 5/3 wavelets are not orthogonal wavelets, the distortion still can be calculated approximately from the wavelet coefficients because of their near-orthogonal feature. In our system’s division of cubes, the distortion is summable since all the cubes are independent of each other. If we take the truncation points into consideration, the overall distortion $D$ can be represented as:

$$D = \sum_i D_{c_i,i}$$  \hspace{1cm} (4.3)

where $D_{c_i,i}$ is the distortion of $i$th block truncated at $j$th point $c_{i,j}$. 

44
CHAPTER 4. PRACTICAL IMPLEMENTATIONS

For the SNR criterion, the distortion stands for the noise in a power sense, which can be represented as:

$$D = \sum_{y=0}^{M-1} \sum_{x=0}^{N-1} \sum_{z=0}^{S-1} (C(x, y, z) - \hat{C}(x, y, z))^2$$  \hspace{1cm} (4.4)

where $C(x, y, z)$ is the original transformed coefficient in a code cube and $\hat{C}(x, y, z)$ is the recovered value of $C(x, y, z)$. Integers $M, N, S$ represent size of each code cube.

To optimize the compression problem, we need to choose the optimal truncation points $c_i$ for each $D_i$ to minimize the whole distortion $D$ at the shortest code length $L$. Using Lagrange multiplier method, the optimization process is to choose some truncation points $c_i$ to minimize:

$$D + \lambda L = \sum_i D_{c_i,i} + \sum_i \lambda L_{c_i,i}$$ \hspace{1cm} (4.5)

It is obvious that, at the optimal truncation distortion $D$ can not be further reduced except for increasing the code length $L$ which is:

$$\arg \min_L D := \{ L | L \leq l \}$$ \hspace{1cm} (4.6)

where $l$ is a specific code word length.

Since the value of $c_i$ is a discrete value, we can’t find an exact value of $\lambda$ to minimize the equation above. However, an algorithm for discrete situation can be applied to find the optimal truncation points. Towards this we define:

$$d_{i,j} = \frac{D_{c_i,j,i} - D_{c_i,j-1,i}}{L_{c_i,j-1,i} - L_{c_i,j,i}}$$ \hspace{1cm} (4.7)

where $d_{i,j}$ represents the slope of $i$th cube at $j$th truncation point and $c_i^j$ stands for the $j$th truncation point of $i$th cube. From what we referred to in Section 1.2.3, the Rate-Distortion curve should be a convex hull function. As a result, the slope $d_{i,j}$ should be a strict decreasing variable. The algorithm to find the optimal truncation points is shown in Algorithm 3.

The inner loop in this algorithm will stop at some $j$, since the variable $d_{i,j}$ is a strict decreasing variable.

PCRD algorithm is widely used in image compression. In JPEG2000 standard, this algorithm is implemented, which is called the EBCOT. The JPEG2000 standard decides the truncation points for each block.
Algorithm 3 PCRD

Initialize $\lambda$

Initialize $c_{i,j} = 0$

for every cube $i$ in the whole image do

while $d_{i,j} \leq \lambda$ do

    $j = j + 1$

end while

$c_{i,j} \leftarrow j$

end for

4.3 PCRD Algorithm Optimization

If we directly generalize the PCRD algorithm to each code cube, the performance should be better than any other arbitrary truncation points. However, the calculation of this algorithm is time-consuming, because the system has to calculate the Rate-Distortion curve for every cube. Therefore, more time is needed for calculation, which impairs the whole system’s performance. In our system, we optimized this algorithm to achieve a high compression ratio with lesser compression time. In the Chapter 3, we tiled the whole bands into several continuous band sets. In this way, we can take advantage from the signature of spectra to enhance the compression performance. Figure 4.5 showed the whole bands of a single pixel which was equally divided into 7 parts. This is possible since when the whole HSI data cube is tiled into $32 \times 32 \times 32$ code cubes.

![Full Band Divided into 7 Parts](image.png)

Figure 4.5: A Full Band Divided into 7 Parts

In Figure 4.5, the 4th and 6th band sets with wide absorption bands of carbon dioxide,
vapor, and nitrogen have relatively low power. The 7th band set also has a relatively lower power
than first few band sets because of the low solar radiation power in these bands. If we reduce the
code cube size, different power of band sets can be more precisely achieved. Figure 4.6 shows how
pixels in spectral direction were equally divided into 14 parts. This happens when the whole HSI
data cube is tiled into $16 \times 16 \times 16$ code cubes.

Figure 4.6: A Full Band Divided into 14 Parts

In most AVIRIS HSI data, some bands are always in a relatively low power state due
to atmospheric absorption and solar radiation, which is briefly introduced in Chapter 1. These
low-powered band sets contribute a relatively minor component to the whole data cube in a power
sense, and a high-SNR compression for these band sets may not be a wise choice. The solar radiation
power in NIR and SWIR bands is much lower than the visible band region. As a result, the power at
these bands often cannot exceed the power in visible band. In a spatial sense, the whole image in
these spectral regions often keep a very low value. That is because the concentration of different
components in atmosphere is always constant in a relatively small region. In absorption bands, most
of the radiation power is blocked by the atmosphere. In summary, the band sets in the absorption
bands may always perform in a similar pattern, which make them amenable for higher compression.

Taking advantage from the different patterns in band sets, we modified the PCRD algorithm
from cube truncation to band set truncation. In this way, each specific truncation point is chosen for
all the code cubes in the same band set and each band set will only have one truncation point. On
the contrary, in the PCRD algorithm without optimization, each code cube will have one specific
truncation point for itself. Hence, this will have more truncation points to calculate. This optimization
accelerates the compression speed for processing lesser truncation points. In AVIRIS data, every
single image is obtained in the same flight. The solar radiation and atmosphere often keep the same
within a single flight. To some degree, the approximate power of different band sets is predictable.
CHAPTER 4. PRACTICAL IMPLEMENTATIONS

The data of each flight contain thousands of scan-lines and pixels per scan line. We can eliminate some small regions in spatial domain for choosing optimal truncation points, and apply these points to the whole image. In this way, the time for calculating truncation points can be saved. A comparison between code-cube-based and band-set-based PCRD is shown in Figure 4.7.

![Figure 4.7: PCRD Control Unit](image)

4.4 Parallel Computing

For small code cubes, such as $8 \times 8 \times 8$, $16 \times 16 \times 16$ or $32 \times 32 \times 32$ cubes, all the cubes are independent of each other in the whole transformed HSI data cube. Hence, the cubes in an image can be parallel computed at a time. However, as the size of code cube grows larger, the computational complexity will grow at a $n^3$ size without considering transformation’s effect on the larger code cube. In this section, we provide our solution to this problem.

The SPIHT algorithm is based on hierarchical trees and all the descendants from different ancestors that in the same decomposition level are independent. This relationship is shown in Figure 4.8 which shows that different shaded blocks in the same decomposition level are independent of each other. In order words, the descendants of each root in LIS will be encoded independently. The encoding process of the SPIHT algorithm implies that the process of each tree is independent of others and the encoded bitstream of roots in LIS is unchanged. We can reorganize the whole image after transforming it into the form in Figure 4.9 which reunites the ancestor pixels in the lowest frequency level and its descendants in the higher frequency level together to formulate a smaller cube. If we view this small cube in a multi-resolution pyramidal structure, the ancestor pixel can be viewed as the lowest frequency level and the pixels from the higher frequency levels in the original
large code cube will be viewed as the same level of high frequency components in the small cube. To show it in a clearer way, we used a 2D-DWT as an example in Figure 4.9.

In this way, parallel computing can be performed on each new small code cube which is almost the same as the encoding process performed on the large cube. Although the codeword is changed in order, the length of the codeword is unchanged. Thus the order of bitstream does not interfere the encoding efficiency.
CHAPTER 4. PRACTICAL IMPLEMENTATIONS

For example, a large code cube is chosen at a $224 \times 224 \times 224$ size. After 5 levels of decomposition, the lowest frequency domain cube should be a $7 \times 7 \times 7$ cube, which contains 343 pixels. Following the approach that we have already discussed, the whole $224 \times 224 \times 224$ image cube will be reorganized into 343 cubes with a size of $32 \times 32 \times 32$ (which is $2^{\text{level}}$). Hence the parallel computing can be easily applied to these 343 independent code cubes. At the receiver side, all we need to do is just reunite these small recovered cubes into the whole transformed HSI data cube.

We have to note that, after reorganization, the whole HSI data cube has both spatial and spectral accessibility at the receiver side, since each pixel in the lowest frequency level and its descendants keep features of the original HSI data in the similar place. Also, for each small code cube, the SPIHT attributes make it SNR scalable. So that, the format of bitstream for the whole image cube will be in a code cube by code cube order with optimization but a stage by stage bitstream order without optimization. The bitstream is shown in Figure 4.10.

Another advantage is that, any transmission error in a single cube will not affect the recovery of the whole image cube and SNR scalability can be achieved at any spatial and spectral in a $32 \times 32 \times 32$ region of accuracy.

4.5 Summary

In this chapter, we discussed how to accelerate compression speed in practical implementations. By exploiting the spectral signature of AVIRIS data, the optimal truncation points can be chosen based on band sets for the whole HSI data cube. In this way, the whole HSI data can be
CHAPTER 4. PRACTICAL IMPLEMENTATIONS

compressed without calculating these truncation points which are decided from sample data. By utilizing the independence of different hierarchical trees in DWT, a large transformed cube can be decomposed into small code cubes for parallel encoding. The only difference is in the output codeword order. However, the reorganization of the codeword order makes the recovery of the whole HSI data cube spatial and spectral accessible and the transmission reliability is enhanced by this approach.
Chapter 5

Results & Analysis

The testing data in this thesis for different schemes comes from the four benchmarks AVIRIS images from NASA. These HSI data were widely used in other compression studies. Figure 5.1 through Figure 5.6 showed the representative images of HSI data cubes that $16 \times 16 \times 16$, $32 \times 32 \times 32$ and $448 \times 448 \times 224$ coding cubes.

5.1 Compression Results

Taking computational complexity of DCT operation into consideration, we only applied the compression algorithm on $16 \times 16 \times 16$ and $8 \times 8 \times 8$ code cubes. In DWT compression, CDF 9/7 wavelet is applied, which is used in lossy image compression in JPEG2000 standard.
CHAPTER 5. RESULTS & ANALYSIS

5.1.1 Results in SNR

Figure 5.4 through Figure 5.6 show the compression performance between all different schemes for the testing HSI data: Moffett Field, Cuprite, and Sample. For DCT, each image cube was tiled into sizes of $8 \times 8 \times 8$ and $16 \times 16 \times 16$ cubes for transformation and compression independently. For DWT in small code cube compression, each image cube was tiled into sizes of $32 \times 32 \times 32$ and $16 \times 16 \times 16$ with DWT decomposition levels of five and four. For DWT in large code cube compression, the transformation (five levels of DWT) and compression algorithm was performed on the whole HSI data cube in a size of $448 \times 448 \times 224$.

![Bitrate-SNR of Moffett Field](image)

Figure 5.4: Bitrate-SNR of Moffett Field

5.1.2 Results in Spectral Profile

In Figure 5.7 through Figure 5.9, the whole image was compressed at bitrates of 1.3 bpppb, 0.3 bpppb and 4.5 bpppb separately of Moffett Field. For each figure, the first graph shows the original signal in the spectral domain and the second one shows the recovered signal. The last graph is the error between the original signal and the recovered one.
CHAPTER 5. RESULTS & ANALYSIS

Figure 5.5: Bitrate-SNR of Cuprite

Figure 5.7: Spectral Profile at 1.3 bpppb
CHAPTER 5. RESULTS & ANALYSIS

Figure 5.6: Bitrate-SNR of Sample

Figure 5.8: Spectral Profile at 0.3 bpppb
5.2 Analysis

Obviously, the DCT on $16 \times 16 \times 16$ code cubes outperformed all the other ways of compression. Traditionally speaking, the wavelet transform should have a better performance than discrete cosine transform, which has already been proved in 2D image compression, such as the JPEG2000 standard. On the other hand, the tree-structured way of coding is designed for wavelet transform which has a unique pyramidal structure. Empirically, the combination of these two features should have a better result than the DCT coding. In many literatures of wavelet coding, tree-structured way of coding takes advantage from the strong correlation between different subbands in the same direction. Stronger the correlation, better the compression results we will have. On the other hand, the tree-structured way of coding should be decided by the power concentration in each subbands. Hence, to achieve a better compression performance of tree-structured coding, we want the power of the whole HSI data concentrates in low subband, meanwhile, inside each subband, we want the large values are grouped together as clusters.

From the SPIHT coding algorithm and our analysis in Chapter 3, the performance of SPIHT algorithm is decided by the bitplanes’ arrangement. To form a larger zero tree for a better compression performance, all values in this tree should be low-valued and this cannot be judged by the correlation between bands.
CHAPTER 5. RESULTS & ANALYSIS

Because of the time and frequency locality, the wavelet transform has its unique advantage in dealing with signal’s spikes and abrupt changes. In the spectral domain, wavelets can easily deal with abrupt changes of atmosphere’s absorption bands. In the small size of code cube, to ensure the tree-structured way of coding can be performed, each dimension should be applied periodization DWT which uses the circular convolution to process the signal. However, this strategy sometimes introduces additional high-frequency components to the original signal. As a result, the more parts we divide, the more additional high-frequency components will be added to the whole signal. Figure 5.10 shows the DWT coefficients of 900 code cubes for the first 32 bands using 32-band and 224-band transform at a level of 5. It is obvious that, the 224 length of DWT has a better power concentration,

32 length DWT

224 length DWT

and the values at higher-frequency subbands have relatively low magnitude. This explained why larger the DWT code cube better the SPIHT coding performance we will have.

In Figure 5.11 and 5.12, all the coefficients in the code cubes are arranged in a baseline order. which ensures all the coefficients are in a tree-structured order. The adjacent coefficients always come from the same root. The horizontal axis is the index of each coefficient in a code cube. The vertical axis is the index of each code cube in the whole HSI data cube. In these figures, black dot means 1 in this bitplane and white one means 0. We can see that, in the first bitplane, all the significant values are concentrated in the lowest frequency bands and as the bitplane approaches LSB, the significant values are in a trend of moving to the higher frequency bands. From the first several bitplanes, the distribution of significant values in DWT cubes seems has no big difference from the DCT ones. However, as the bitplane moves to LSB, the DWT cubes’ bitmap performs fuzzier than the DCT ones. From the figure of 10th bitplane, we can find that the significant values in

Figure 5.10: DWT Coefficients of Different Lengths of DWT
CHAPTER 5. RESULTS & ANALYSIS

10th bitplane always appear in some specific positions in most of the cubes. On the contrary, the DWT cubes’ significant values distribute in a wider range.

The situation becomes the worst in the 13th bitplane. The DWT cubes’ bitplane becomes fuzzier and we can see several dark lines after 2000 samples, which belong to the highest frequency bands. From what we have discussed in the former chapter, large values in highest frequency subbands significantly vitiates the efficiency of the SPIHT algorithm, and almost all the code cubes in this whole HSI data cube have this deficiency. Compared to the DWT cubes, the DCT cubes seem didn’t change much from the 10th bitplane, only some bits are significant in the highest frequency subbands and this explained why DCT based algorithm has a better performance.

5.3 Summary

In this chapter, we applied DWT and DCT to three different HSI data cubes. Taking the computational complexity into consideration, we only perform DCT on $8 \times 8 \times 8$ and $16 \times 16 \times 16$ code cubes. From the results, we conclude that, the DCT based 3D-SPIHT algorithm with a $16 \times 16 \times 16$ code cube size achieves the best compression ratio under given SNR. From the bitplane map of DCT and DWT coefficients in the same HSI data cube, the DWT based code cubes’ bitplane map is fuzzier than the DCT ones, especially when the bitplane goes toward LSB, though there are no big difference in the first several bitplanes. This is in accordance with the conclusion we made in Chapter 3 and this phenomenon also implies why the difference of compression results between these two transformations are not very big when the bitrate is very low. Hence, although the SPIHT algorithm is designed for DWT, in some specific situations (regarding cube size) the SPIHT algorithm with DCT still can have a competitive performance. DCT does not have the pyramidal structure in DWT, however, in some situations, its bitplanes’ arrangement is more suitable for SPIHT algorithm.
CHAPTER 5. RESULTS & ANALYSIS

Figure 5.11: Bitplanes of $16 \times 16 \times 16$ DCT Coefficients
Figure 5.12: Bitplanes of 16 \times 16 \times 16 DWT Coefficients
Chapter 6

Conclusions and Future Work

6.1 Conclusion

Lossy HSI data compression significantly helps in HSI data storage and transmission. By discarding some information of the original data under an acceptable range, HSI data can be compressed to a very small size. In this thesis, we presented a novel architecture for compressing AVIRIS data with pre-processing and compression stage. In our test, we showed that this system is proved to be effective in compressing HSI data. In the compression stage, we take practical implementations into consideration. In this stage, we modified the PCRD algorithm in AVIRIS data compression and implemented parallel computing. These modifications make the compression system more efficient. From previous works in HSI data compression using DWT, the approaches are often the combination of three-dimensional DWT and wavelet coding algorithm (such as SPIHT, EZW or SPECK). However, we focused on the traditional DCT in our work, and combined this transformation with the wavelet coding algorithm. The results is better than the traditional DWT. In our system, the 3D-SPIHT coding algorithm is applied and results that can predict the performance of SPIHT coding theoretically are also derived. These conclusions helped us analyze compression results.

In the compression stage, the system’s performance is enhanced by implementing parallel computing and the modified PCRD algorithm. In parallel computing, taking advantage from the independence of hierarchical trees, the large code cube can be decomposed into independent small code cubes for compression. As a result, the spatial and spectral accessibility can be both achieved. For reducing the computational complexity in the PCRD algorithm, we modified the the PCRD process from code cube based to band set based. Taking advantage from the features in spectra of
CHAPTER 6. CONCLUSIONS AND FUTURE WORK

AVIRIS data, the times of PCRD procedure to be performed can be highly reduced.

Finally, we gave our results in different cube sizes of compression. We found that, larger the code cube, better the compression results will be for the DWT coding. However, the $16 \times 16 \times 16$ size of code cube while using DCT outperformed all the others. We analyzed our results using bitplanes and the results were compatible with our theoretical conclusion.

6.2 Future Work

In this thesis, we applied cosine signal and CDF 9/7 wavelet as our basis for decomposition. It has already been proved that, the discrete cosine basis are the suboptimal basis for decomposing natural images. However, for hyperspectral images, the image from a single band is fuzzier than the common images and as the spectral dimension is added in, the situation becomes complex. Hence, the future work can focus on finding some optimal or suboptimal basis for hyperspectral image compression.

Most works concerning the transform coding for hyperspectral images used transformation to each dimension independently. To deal with the multi-dimensional decomposition, tensor is an option for this problem. However, with the computational complexity and the freedom of decomposition, we don’t know whether this way is applicable in practical. Hence the future work can focus on whether tensor can be applied in practical use in hyperspectral image compression.
Bibliography


Appendix A

SPIHT Recursive Functions

A.1 Encoding Cost in Best Case

In general, the encoded SPIHT codeword will contain 4 different kinds of symbols, which are \(0, 1, +, -\). The 0 and 1 indicate testing results of each hierarchical tree in LIS. In a \(m\) (where \(m > 2\)) level of DWT decomposition, the pixels in LIS from \(S_2\) will be tested as Type A and Type B twice and pixels in \(S_1\) will be recognized as single pixels without descendants. As in the best case, there are no other non-zero values in \(S_n\) (where \(n > 2\)). The testing procedure of pixels from \(S_2\) will be kept on a status of type A. Therefore, in a \(k\) bitplane testing system, there will be \(n_2\) number of 0 to be outputted for \(k\) times. In the first stage, all the values in \(S_1\) and \(S_2\) are significant and there should be \(n_2 + n_1\) symbol of + to be outputted. As a result, the total encoding cost \(C\) can be written in:

\[ C = (k + 1)n_2 + n_1 \]  
(A.1)

A.2 Encoding Cost in Case A

In case A, all the sets will have a random value for each element, however, they also satisfied the following rule:

For any elements \(e_i\) in \(i\)th set \(S_i\)(except for elements in \(S_1\)),

\[ 2^{\left\lfloor \log_2 \min(S_{i-1}) \right\rfloor - 1} < e_i \leq 2^{\left\lfloor \log_2 \min(S_{i-1}) \right\rfloor} \]  
(A.2)

For a pixel \(e_i(i \leq M - 1)\) just added in LIS, it shall be tested as significant pixel, pixel as type A and pixel as type B. In bitplane \(i\), \(e_i\) will firstly be tested as significant pixel and the system will output
APPENDIX A. SPIHT RECURSIVE FUNCTIONS

its sign as + or −. Then in LIS, \(e_i\) will be marked as type A and testing result in this stage shall be \(J(\emptyset(e_i))\) which is 0.

In bitplane \(i + 1\), \(J(\emptyset(e_i))\) will output 1 as Type A and 0 as Type B. In bitplane \(i + 2\), \(J(\emptyset(e_i))\) will output 1 as Type B and the pixel \(e_i\) will be removed form LIS.

From the tracking of pixel \(e_i\) in LIS, we can conclude that, each pixel from \(S_2\) to \(S_{m-2}\) will be tested as significant pixel for one time, Type A for two times and Type B for two times. Such that, each pixel will output 5 symbols in the whole encoding procedure. Therefore, the encoding cost of pixels from \(S_2\) to \(S_{m-2}\) will be written as: 

\[
T_k = 5n_2 + \sum_{i=3}^{k-2} 5n_i
\]  

(A.3)

In a recursive form, we assume that, the coding cost of a single pixel \(e_i\) is \(Y[i]\). The coding cost of \(e_i\) can be written into a form of the summation of its 4 direct-children’s coding cost and the side information that generated by \(e_i\) itself. As what we referred above, the side information of pixel \(e_i\) should be the testing results as significant pixel for one time, Type A for two times and Type B for two times, which are 5 symbols. Therefore, the coding cost of pixel \(e_i\) in a recursive form is:

\[
Y[i] = 4Y[i+1] + 5
\]  

(A.4)

A.3 Encoding Cost in Case B & Case C

In case A, none of the pixels will be in LIP. However, in case B, all the direct-children(except for the pixels in the highest level) of pixel \(e\) whose descendants contains \(e_{ck}\) will be added into LIP. Pixel \(e_{LIP}\) with a value between \(2^i\) to \(2^{i+1}\) was added into LIP at a stage whose judging function is judged at a threshold of \(2^n\). From the SPIHT coding rule, we know that, before removing from the LIP, this pixel will be tested and the system will output 0 at every stage until the \((n - i + 1)\)th stage comes (In this stage, pixel \(e_i\) will be viewed as significant value and removed from LIP).

In case B, the coding cost of pixel \(e_n\) can be represented in a form of the coding cost of its 3 direct-children without containing \(e_{ck}\), 1 direct-child containing \(e_{ck}\), 4 direct-children in LIP and the side information of \(e_n\) itself. Because all the pixels in LIP are added at the first stage, the coding cost of each pixel \(e_i\) from \(S_i\) in LIS should be \(n + 1\) and there will always be 4 pixels removed in each stage. In SPIHT coding algorithm, pixel \(e_n\) will only be tested as Type A and Type B for one time.
APPENDIX A. SPIHT RECURSIVE FUNCTIONS

Therefore, there will only be 2 symbols outputted as side information. In sum, the coding cost of pixel $e_n$ in $S_i$ can be written in:

$$X[n] = 3Y[n + 1] + X[n + 1] + 4(n + 1) + 2$$  \hspace{1cm} (A.5)

In case C, the following recursive function set can be naturally deduct from the function in case B.

\[
\begin{cases}
    Z[n] = 3Y[n + 1] + Z[n + 1] + 4(n + 1) + 2 & n < m \\
    Z[n] = 2Y[n + 1] + 2X[n + 1] + 8(n + 1) + 4 & n \geq m
\end{cases}
\]  \hspace{1cm} (A.6)