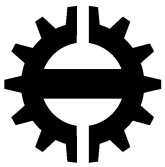

TAMPERE UNIVERSITY OF TECHNOLOGY
Department of Information Technology

Jani Huhtanen

LOSSLESS COMPRESSION OF MULTI- AND HYPERSPECTRAL DATA

Master of Science Thesis

Subject approved by the department council on March 8, 2006

Examiners: Prof. Tarmo Lipping
Dr. Tech Jari Turunen

Prologue

I have done my thesis in Tampere University of Technology - Pori between January and August 2006 as a part of 3DSata project. In my thesis I compared several lossless image compression algorithms and developed a new algorithm specifically meeting the requirements of hyperspectral imagery and their applications.

I would especially like to thank the examiners of my work, professor Tarmo Lipping and Dr. Tech Jari Turunen, for their helpful comments, support and motivation. I thank the financiers of the 3DSata project. I would also like to thank the developers of the free open source editing platform TeXmacs, which I used to write my thesis. And at last but not least, I wish to thank my wife Jatta, for her love and understanding during the writing of this thesis.

Jani Huhtanen
August 9, 2006, in Pori
Keulantie 3 B 30
28190 Pori
phone: +358 40 7358516
e-mail: jani.huhtanen@tut.fi


This document has been produced using GNU TeXmacs (see `http://www.texmacs.org`).

Table of contents

Abstract

TAMPERE UNIVERSITY OF TECNOLOGY
Department of Information Technology
Information Technology, Pori
HUHTANEN, JANI: Lossless Compression of Multi- and Hyperspectral Data
Master of Science Thesis, 76 pages
Examiners: Professor Tarmo Lipping and Dr. Tech Jari Turunen
Financing: Tekes, City of Pori, Kemira, Ramboll Finland
August 2006
Keywords: compression, image processing, hyperspectral imaging, adaptive prediction, entropy coding, wavelet transform
UDC: 004.627, 004.932, 621.391.1

Aerial and satellite images have often high resolution and they cover large areas. As a result the dimensions of the images are large and require huge amounts of storage resources. Lossless compression is one technique used to alleviate the problem. In this thesis, a novel lossless compression algorithm for multi- and hyperspectral data is presented. The compression ratio for hyperspectral images and the features of the algorithm are compared to those of other lossless image compression algorithms.

The proposed algorithm uses the wavelet transform and a novel inter-band adaptive predictor to achieve high compression ratio while preserving low complexity and flexibility. Properties such as scalability in resolution and fast spatial and spectral random access are guaranteed. Adaptation of the inter-band predictor coefficients is based on multiresolution analysis provided by the 5/3 wavelet transform. Furthermore, low complexity implementation of the wavelet transform, called the lifting scheme, is employed. Prediction errors are entropy coded with fast adaptive Golomb-Rice coder.

Results from the comparison with other algorithms show that compression ratio provided by the proposed algorithm is very competitive. In some cases the proposed algorithm provides over 20% increase in compression ratio when compared to JPEG2000 compression algorithm. These results are not suprising because of the more sophisticated exploitation of strong inter-band dependencies in hyperspectral images than what is used in JPEG2000. When the features, such as scalability in resolution and fast random access, are considered, the proposed algorithm proves to be very competitive against all the tested algorithms: JPEG2000, JPEG-LS, ICER-3D and the so called Fast Lossless algorithm.

Tiivistelmä

TAMPEREEN TEKNILLINEN YLIOPISTO
Tietotekniikan osasto
Tietotekniikka, Pori
HUHTANEN, JANI: Lossless Compression of Multi- and Hyperspectral Data
Diplomityö, 76 sivua
Tarkastajat: Professori Tarmo Lipping ja tekniikan tohtori Jari Turunen
Rahoittajat: Tekes, Porin kaupunki, Kemira, Ramboll Finland
Elokuu 2006
Avainsanat: pakkaus, kuvan käsittely, hyperspektrikuvantaminen, adaptiivinen ennustus, entropiakoodaus, wavelet-muunnos
UDK: 004.627, 004.932, 621.391.1

Ilma- ja satelliittikuvien tarkkuus on usein korkea ja kuvat kattavat laajoja alueita. Hyperspektrikuvissa kanavien määrä voi olla jopa yli 200. Näistä syistä kuvat vaativat paljon tallennuskapasiteettia. Yksi ratkaisu tallennuskapasiteetin säästämiseen on häviötön pakkaus. Tässä työssä esitellään uudenlainen moni- ja hyperspektrikuvien häviötön pakkausmenetelmä. Lisäksi uuden menetelmän pakkaussuhdetta ja ominaisuuksia verrataan muihin menetelmiin.

Ehdotettu algoritmi saavuttaa korkean pakkaussuhteen, ollen samalla laskennallisesti kevyt, käyttämällä wavelet-muunnosta ja uudenlaista kanavien välistä adaptiivista ennustajaa. Algoritmi takaa nopean satunnaissaannin sekä kuva-avaruudessa, että kanavien välillä. Lisäksi algoritmi tukee skaalautuvuutta kuvan tarkkuudessa. Kanavien välisen ennustajan kertoimien mukauttaminen perustuu moniresoluutioanalyysiin. Moniresoluutioanalyysi suoritetaan 5/3 wavelet-muunnoksella. Wavelet-muunnoksen toteuttamiseen käytetään laskennallisesti nopeaa, niin kutsuttua lifting-menetelmää. Ennustusvirhe pakataan käyttäen adaptiivista Golomb-Rice-koodausta.

Tulokset vertailuista osoittavat, että ehdotetun algoritmin mahdollistama pakkaussuhde on erittäin kilpailukykyinen muihin menetelmiin nähden. Tietyissä tapauksissa voidaan saavuttaa, jopa 20% etu pakkaussuhteessa JPEG2000:een nähden. Tämä tulos ei ole yllättävä, koska ehdotetun menetelmän kanavien välisten riippuvuuksien hyödyntäminen hyperspektrikuvilla on kehittyneempi kuin JPEG2000:n vastaava. Kun tarkastellaan ehdotetun algoritmin ominaisuuksia, kuten nopeaa satunnaissaantia, osoittautuu menetelmä erittäin kilpailukykyiseksi kaikkiin testattuihin algoritmeihin (JPEG2000, JPEG-LS, ICER-3D ja niin kutsuttu Fast Lossless) nähden.

Abbreviations and notation

Abbreviations

Abbreviation	Explanation
AA	Approximation–Approximation part
AD	Approximation–Detail part
CALIC	Context-Based Adaptive Lossless Image Coder
CCD	Charge Coupled Device
DA	Detail–Approximation part
DD	Detail–Detail part
EM	Electromagnetic
FIR	Finite Impulse Response
FOV	Field of View
GIS	Geographic Information System
GPU	Graphics Processing Unit
I/O	Input/Output
i.i.d.	Independent and Identically Distributed
IFOV	Instantenous Field of View
IIR	Infinite Impulse Response
JPEG-LS	Joint Photographic Experts Group – Lossless
JPEG2000	Joint Photographic Experts Group 2000
LOCO-I	Low Complexity Lossless Compression for Images
LTI	Linear Time Invariant
MRA	Multireresolution Analysis
MSE	Mean Squared Error
pmf	Probability Mass Function
QMF	Quadrature Mirror Filter
RGB	Red, Green, Blue
RV	Random Variable
SAM	Spectral Angle Mapper
VLC	Variable Length Coding

General notation

Notation	Explanation
a	Scalar variable, constant or symbol
A	Random variable or set
$\boldsymbol{a}$	Column vector
$\boldsymbol{A}$	Matrix or random vector variable
A_n	Set n or random variable n
a_n	Sample n of random variable A_n or sample n of a signal
a_n^i	Element i of neighborhood of sample a_n
$a[n_0, n_1,..., n_{M-1}]$	Sample from M-dimensional signal or a signal, where $n_0, ..., n_M \in \mathbb{Z}$
$a(t_0, t_1, ..., t_{M-1})$	Function, where $t_0, ..., t_{M-1} \in \mathbb{R}$
$\boldsymbol{a}^T, \boldsymbol{A}^T$	Transpose of a column vector or a matrix
$\boldsymbol{a}\boldsymbol{b}^T$	Outer product of column vectors $\boldsymbol{a}$ and $\boldsymbol{b}$
$\boldsymbol{a}^T\boldsymbol{b}$	Inner product of column vectors $\boldsymbol{a}$ and $\boldsymbol{b}$
$\boldsymbol{a}[n_0, n_1,..., n_{M-1}]$	Vector valued sample from M-dimensional array, where $n_0, ..., n_M \in \mathbb{Z}$
$\boldsymbol{A}[n_0, n_1,..., n_{M-1}]$	Matrix valued sample from M-dimensional array, where $n_0, ..., n_M \in \mathbb{Z}$
$\int_a^b \cdot dt$	Integration from a to b over t
$\sum_{n=a}^{b} \cdot$	Sum where n goes from a to b
$\sum_n \cdot$	Sum where n goes over all values it is defined for
$\sum_{n_0,...,n_{M-1}} \cdot$	Shorthand for $\sum_{n_0}, ..., \sum_{n_{M-1}} \cdot$
$\cdot^*$	Complex conjugate
$< f(t), g(t) >$	Inner product of functions defined as $\int_{-\infty}^{\infty} f(t)g^*(t)dt$
$\arg\min \cdot$	Returns the argument which minimizes the given function
$x \in A$	x is in set A
$A \subset B$	A is a subset of set B
$\operatorname{span} A$	Given set of functions or vectors, A, span returns the set the elements of A span
$A \perp B$	Two sets A and B are orthogonal
$A \oplus B$	Direct sum of sets A and B
$\mathbb{R}$	Set of real numbers
$\mathbb{Z}$	Set of integers
$\mathbb{N}$	Set of natural numbers
$\mathbb{N}_0$	Set of natural numbers augmented with 0
$L_2\{\mathbb{R}\}$	Set of functions for which $< f(t), f(t) >$ is finite and $t \in \mathbb{R}$, i.e., set of square-integrable functions

$a \Leftrightarrow b$	Equivalence of two propositions a and b
$\lfloor \cdot \rfloor$	Rounding towards negative infinity
$[\cdot]$	Rounding towards the nearest integer or a indexing operator
$\mathrm{A}\{\cdot\}$	Operator

Symbols, functions and notation with special meaning

Notation	Explanation
$p(x)$	Probability mass function, where $x \in \mathbb{Z}$
$p_X(x)$	Probability mass function corresponding to a random variable X, where $x \in \mathbb{Z}$
$H(X_0, X_1, ..., X_{M-1})$	Joint entropy of random variables $X_0, X_1, ..., X_{M-1}$
$H(X_0 \vert X_1, ..., X_{M-1})$	Conditional entropy of random variable X_0 given $X_1, ..., X_{M-1}$
μ	Mean
$\hat{\mu}$	Average, estimate of mean
z	Complex variable used in the $z-$transform
ω	Angular frequency in radians
$A(z)$	$Z-$transform of an 1D signal or system function
$A(\omega)$	Fourier$-$transform of an 1D signal or frequency response
$H(z)$	System function of the lowpass synthesis QMF or system function of a general FIR filter
$\tilde{H}(z)$	System function of the lowpass analysis QMF
$G(z)$	System function of the highpass synthesis QMF
$\tilde{G}(z)$	System function of the highpass analysis QMF
$\varphi(t)$	Scaling function
$\psi(t)$	Wavelet function
$\delta[n]$	Unit-impulse
e_n, $e[n_0, ..., n_{M-1}]$	Sample of residual signal or a residual signal
x_n, $x[n_0, ..., n_{M-1}]$	Sample of input signal or an input signal
y_n, $y[n_0, ..., n_{M-1}]$	Sample of output signal or an output signal
$w[n]$, $\boldsymbol{w}$	Predictor coefficients in signal and vector notation
$a[n]$, $a_i[n]$	Denominator coefficients of an LTI filter or approximation part of the wavelet transform at level i
$b[n]$	Nominator coefficients of an LTI filter
$d[n]$, $d_i[n]$	Detail part of the wavelet transform at level i
$\mathcal{J}(\cdot)$, $\hat{\mathcal{J}}(\cdot)$	Cost function and its approximation

1 Introduction

1.1 Objectives of the work

Multispectral image is a term used of images that have more than the three traditional spectral channels (i.e., red, green, and blue). Often these extra channels are for color infrared. Similarly, hyperspectral images have even more channels, sometimes as much as over 200. Channels of hyperspectral images may only contain bands belonging to visible area of the spectrum or they may span the spectrum more widely ranging from X-Rays to radiowaves. When an image has over 200 channels, it has a high spectral resolution and consequently the amount of data can grow over 60-fold when compared to traditional RGB-images with the same spatial resolution. Such images are often acquired from an aeroplane or satellite and can be further analyzed and processed to obtain information about the target (e.g., forrest, soil, enemy base, etc.). Obtaining information about target without actually having any contact to it is called remote sensing which is explained in more detail in chapter (2).

Often multi- and hyperspectral images are analyzed and studied with computer aided methods and not only by visually interpreting the images. Human visual system cannot always recognize degradation in image altough it is present. Degradations, invisible to human, can be caused by number of different processes. One such process is lossy compression where information, not important to humans, is discarded in favor of more efficient compression. Algorithms do not, in general, have the property of human visual system (i.e., being neutral to the lost information). More precisely numerical algorithms view the data differently and thus what is important to algorithm might not be important to human visual system.

Algorithms are numerous, but as an example one could mention classification of regions of the image to different types of soils. Classification accuracy suffers from any degradation of the compressed image. By using *lossless* compression methods it can be guaranteed that further analysis of the image is reliable. Lossless compression achieves compression, not by removing "unimportant" information, but finding more compact representation for the existing information. This guarantees that no degradation what so ever will happen to the compressed image.

Aerial images have often high resolution or they cover large areas. As a result the dimensions of the images are large and often only small region of the image is visible when the image is viewed. These kind of situations, where only part of the image is visible, arise also when one, for example, magnifies a portion of the image to see smaller details. The *multiresolution* representation is a technique where image (or any data) is stored as a low resolution version with the corresponding detail layers.

Detail layers contain the information required to reconstruct the original data in full resolution. Multiresolution representation helps when the image is viewed in a resolution where not all of the details are visible by lowering the need for resources. When the multiresolution representation is already built in the compression stage, viewing the image at coarser resolution requires only partial decompression. This is also one approach in providing a *lossy preview* of the image data and should lead to more efficient use of for example the network bandwidth.

Magnifying comes with the need for panning, but panning may also be required for non-magnified images. For smooth operation the panning has to work quickly. Delays caused by loading new parts of the image causes unnecessary frustration for the end user which can be avoided by using smart techniques for *fast random access*. Fast random access means that any part of the image can be accessed quickly and without significant loading or processing times. Panning needs fast spatial random access, but when dealing with hyperspectral images, often also fast spectral random access is required. By spectral random access it is meant access to different channels or bands of the image. The need for fast access to different bands follows from the fact that usually only 3 bands can be visualized simultanously. These three bands are used to view the important channels of the hyperspectal image. The importance of a certain channel of the image depends on what the user is interested of. For example different types of surfaces have different spectral properties and can be identified from certain channels better than from others.

Fast random access can be achieved in numerous ways and the best approach depends on the use. When image is to be used and viewed over network or from slow harddrive it is important that transfers are minimized. The amount of the data to be transfered can be minimized by transfering the data in compressed form. Unfortunately the images are large and transfering a whole image in order to view a small portion of it is waste of resources. Therefore being able to divide a large image into subimages without complete decoding is important. This way only the required part of the image can be sent in compressed form and decoded in the receiving end. Fast random access requires also that the decoding process has to be quick and thus low complexity decoding is necessary. Low complexity decoding means that decompression requires only little processing resources which in turn directly reflects to the cost of the required equipment.

The developed algorithm should meet all the requirements explained above and still provide high compression ratio. It should be noted that high compression ratio for lossless image compression is very different from what can be achieved by lossy compression. Lossless algorithms typically achieve compression ratios of 2:1 to 3:1, but this is of course highly dependant on the given image. Especially noise is a feature which restricts lossless compression ratios. Therefore it is often not only hard to achieve higher compression ratios but it is impossible because of the presence of the noise. This should be acknowledged when balancing between compression ratio and complexity of the algorithm. Nevertheless, higher compression ratios could be achieved when compared to general image compression algorithms because of the specific knowledge on the target (i.e., hyperspectral aerial imagery). This means that it might be possible to model the target images more accurately when the algorithm is only used for certain subset of all possible images.

The purpose of this work is to *a*) compare existing image compression algorithms and their applicability to multi- and hyperspectral data and *b*) develop a new algorithm suitable for the compression of large hyperspectral images. The developed algorithm should meet the special requirements posed by the later use of the compressed images.

2 Remote Sensing: Image acquisition and analysis

The term *remote sensing* is often defined to mean acquisition of information about a target without physically touching it. Most living creatures do this all the time by, for example, looking, hearing or smelling the target. This is a very broad definition of remote sensing, however, and in this thesis I concentrate only on image acquisition by sensing electromagnetical radiation. More specifically, I concentrate on raster image data of the earth surface obtained by airborne or satellite sensors.

Applications of remote sensing are numerous. The most obvious and perhaps the oldest application is military. Satellite or aerial images of the enemy base bring irreplaceable information about the enemy. Obviously, this is not the only use for remote sensing. Aerial images can be used, for example, to assess the state of the nature, level of pollution, or for temporal analysis of the forrest growth. Cities may use orthoimages (i.e., images with orthographic projection) as support material for town planning. Surface classification techniques can be used, e.g., to detect roads and automatically calculate road coverages. Remote sensing is also used to support strategic planning. For example, in Canada and USA remote sensing information is used for wildfire assessment and as a help for wildfire managers to make strategic decisions (Quayle et al., 2004).

2.1 Electromagnetic radiation

Electromagnetic (EM) radiation is the carrier for the visual information we perceive. Human eye is sensitive only to a specific spectral range of electromagnetic radiation, often called just light, but this is not a universal restriction for imaging. In fact we can simultaneously take images of the electromagnetic radiation of many separate bands of the spectrum, not restricted to visible light. Often instead of frequency the more common measure, wavelength, is used when characterizing the radiation. Frequency (v) and wavelength (λ) are inversely proportional (eq. (2–1)). In equation (2–1) symbol c represents the speed of light. The unit of the frequency is Hertz and the unit of the wavelength is meter. If one visualizes an electromagnetic wave as a sinusoid then the wavelength is the distance between the two closest maximums of the wave.

$$\lambda = \frac{c}{v} \tag{2–1}$$

In figure (2–1) part of the electromagnetic spectrum is shown. The spectral range marked as "Visible" is the range to which the human eye is sensitive (i.e., 400 nm – 700 nm). Although the eye is sensitive to the whole visible range of the spectrum, we still can not directly "see" the spectrum. Instead, we sense the three primary colors: red, green, and blue. These three colors are not sensed as discrete narrow bands of the spectrum but wider bands overlapping each other. Together they span the whole visible range. Similarly, imaging sensors can span wide ranges or, alternatively, just narrow bands. However, sensors in general are not restricted to the visible range.

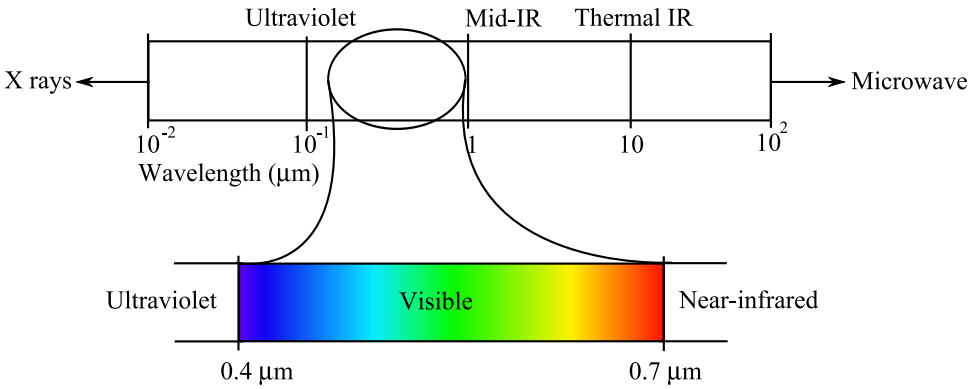

Figure 2–1. Part of the electromagnetic spectrum.

In figure (2–2) two primary sources of electromagnetic radiation are shown together with the corresponding distributions of the energy as a function of wavelength. The surface temperature of the Sun is approximately 6000 K and the surface temperature of the Earth is approximately 300 K. Clearly the visible range of the EM radiation is dominated by the radiation originating from the Sun. At the range of thermal infrared (i.e., around 10 μm) Earth's radiation gets increasingly important.

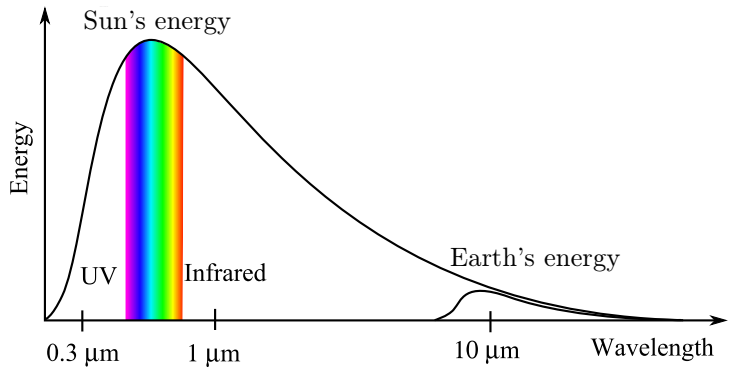

Figure 2–2. Energy of the electromagnetic radiation from the Sun and the Earth. (The graph is not in scale.)

2.2 Image acquisition

In the context of this work all images are assumed to be aerial or satellite images. Aerial images are usually taken from an aircraft. In this case aircraft has an imaging sensor or a film camera onboard, which collects the data while the aircraft flies over the desired target area.

Incoming energy of the radiation is transformed into an output voltage in the sensor. The level of the voltage is proportional to the energy of the incoming radiation, but is also affected by the wavelength of the radiation. Sensor responsivity is a measure of the electrical response per watt of incoming radiation. Responsivity is a function of the wavelength and thus different sensor materials can be used only in certain ranges of the spectrum. For example, silicon can be used when sensing radiation from 400 nm to 1100 nm. Lead sulfide has good responsivity from 1100 nm to approximately 5000 nm. In order to obtain finer spectral resolution than what the sensor material itself provides filters, prisms and refractive gratings are used to further decompose the radiation into the spectral components. (Landgrebe, 2003, p. 60–66)

Important property of a sensor is its signal-to-noise ratio. It is especially important in the context of lossless compression because it is impossible to compress uniformly distributed white noise losslessly. This means that noise is the ultimate limiting factor of lossless compression. Often the dominant type of noise in the sensor is thermal (or Johnson) noise (Landgrebe, 2003, p. 62). Thermal noise is white and follows Gaussian distribution and thus especially problematic when considered from the compressions point of view.

When designing hyperspectral sensors there is a tradeoff between spatial resolution, spectral resolution and signal-to-noise ratio. This is easy to see when one notes that thermal noise is constant and independent of the incoming energy (assuming approximately constant sensor temperature). Finer spatial resolution means that the radiation reaching the sensor corresponding to one pixel (see section 2.3) comes from smaller area and thus has less energy. The same is true for finer spectral resolution. If the spectrum is divided into smaller bands the total energy stays equal but a single band contains less energy. At some point the energy of the sensor noise reaches the energy of the signal. Therefore, hyperspectral sensors have often coarser spatial resolution than the sensors with coarsers spectral resolution. (Landgrebe, 2003, p. 6 and 70)

To give a better understanding about the formation of the image I shortly present few common sensor types as an examples. Here, sensor type refers to the way the ground is scanned or imaged. In figure (2–3) the operation of the line scanner is presented. Line scanner operates by scanning the ground line by line. Scanning is implemented by rapidly changing the sensing angle. This can be achieved, for example, by the use of spinning mirrors. In figure (2–3), acronyms IFOV and FOV refer to the terms *instantenous field of view* and *field of view*, correspondingly. IFOV is the angle the sensor sees at one time instant (i.e., the grey box) and FOV is the angle the sensor sees during scanning of the line. By sampling the sensor output,

the gray box effectively becomes one pixel at the raw data output (see section 2.3). A variation of the line scanner, the whisk-broom, scans several lines simultaneously. This allows the sensor to achieve higher spatial resolution. (Scott, 1997, p. 148, 154)

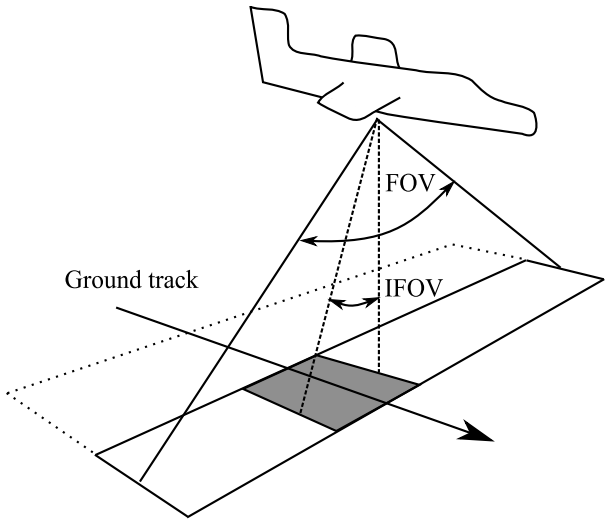

Figure 2–3. Airborne line scanner

An example of instruments using whisk-broom scanning is AVIRIS. AVIRIS is NASAs commercial imaging spectrometer which can be used in different aircrafts. AVIRIS collects information from 224 spectral bands (or channels) simultaneously. Every band has approximately 10nm bandwidth and the 224 bands span the spectrum from 400 nm to 2500 nm (i.e., from visible violet to mid-IR). (NASA AVIRIS website, 2006)

It should be noted that it is impossible to scan a straight line in finite time because of the movement of the aircraft. Furthermore, the area under a single pixel is not constant because of the slightly different view angles during scanning. Also, changes in the orientation of the aircraft during the flight result in further distortions. By tracking these anomalies during the flight (e.g., by use of gyroscopes) it is possible to correct the distortions by post-processing. (Scott, 1997, p. 148)

Other type of sensors is the so called bush-broom sensor. This sensor is able to collect a complete line simultaneously by using linear array (see figure (2–4)). Using two dimensional array one can collect several complete lines simultaneously. This approach is very similar to common consumer digital cameras where a two dimensional CCD-matrix (charge coupled device) is used as a sensor.

Different spectral bands are captured using one linear array (or matrix) per band. Before sensing, the radiation has to be decomposed into the spectral bands using, e.g., prisms, refractive gratings or filters and diffracted to correct sensors with lenses and mirrors. For example, some digital cameras have three CCD-matrices and often use dichroic mirrors for the band separation (i.e., red, green, and blue). (Scott, 1997, p. 158 and 152)

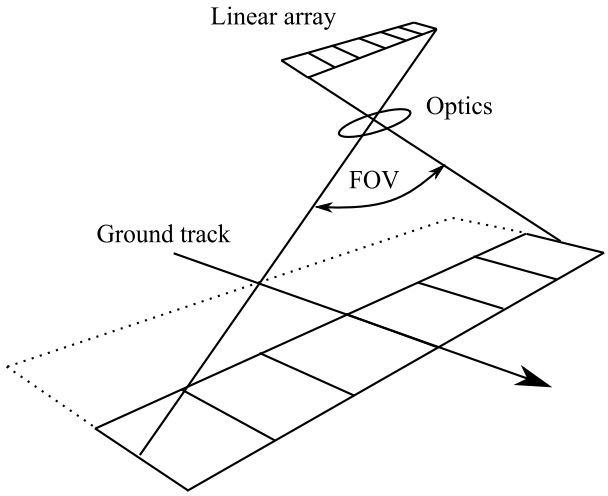

Figure 2–4. Bush-broom sensor

2.3 Multi- and hyperspectral images

In order to obtain a digital image, the continuous image has to go through two processes: sampling and quantization. Sampling is a process where continuous image is transformed into discrete samples. One sample corresponds to a certain area in the original image and every sample has a certain value representing that sample. In general, sample may have any real value. The area corresponding to one sample tells the resolution of the image. For example, an aerial image may have a sample size (or ground pixel size) of 16 meters, which means that one sample in the image corresponds to area of 16 by 16 meters in the ground.

Because computers represent the sample by a finite number of bits, the sample value has to be quantized. Quantizer maps the real valued samples to discrete valued samples. This process is irreversible. It should be noted that although quantization is irreversible the sampling is reversible if certain conditions are met (Jain, 1989, 84–87). Sampling and quantization are illustrated in figure (2–5).

Quantized values are represented by bits. In two's complement representation N bits can represent 2^N different values. The resolution of the quantization is often expressed by *bitdepth*, referring to the number of bits used to represent each sample. Common bitdepth is 8 bits enabling to represent 256 quantization levels. In this text only images where quantization is uniform are considered, meaning that the distance between two adjacent quantization levels (i.e., possible values for the sample) is constant as in figure (2–5). In the case of uniform quantization, the maximum absolute error caused by the quantization is half the quantization interval (i.e., the distance between two adjacent levels, see figure (2–5)). The number of quantization levels is a choice which is affected by the intended use of the image and by the signal-to-noise ratio of the sensor. Quantization itself introduces noise in the form of quantization error (see figure (2–5)), but it is clear that adding quantization levels does not lower the signal-to-noise ratio significantly when the sensor is the limiting

factor. For more information about sampling and quantization see (Jain, 1989, p. 80–131).

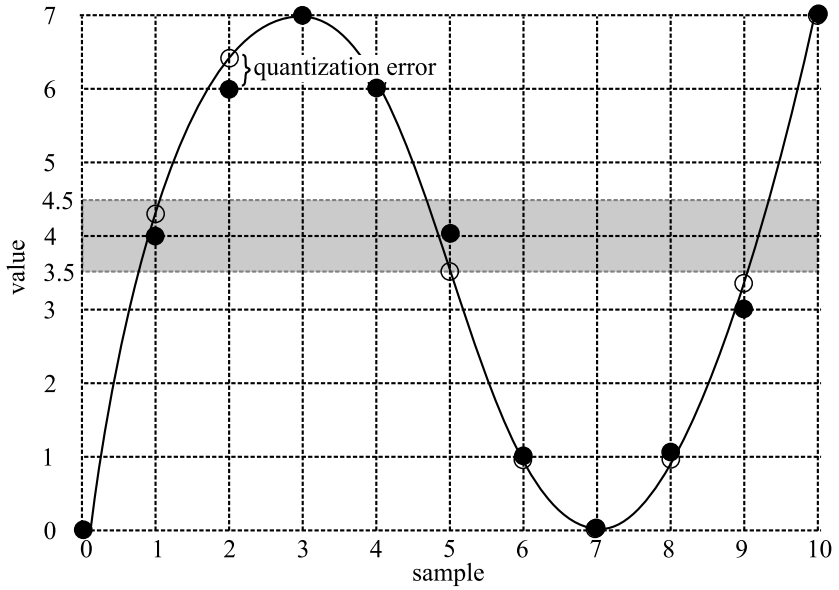

Figure 2–5. The sampling and quantization. Original continuous signal is represented by the solid line, white circles represent values of the sampled signal and black circles represent quantized samples. Grey area encloses the sample values that will be mapped to value 4 (assuming rounding towards the nearest integer).

A gray scale image can be thought of as a matrix of discrete quantized values (often representing intensity) as expressed in equation (2–2). A single value is referred to as a pixel (i.e., picture element). In this work pixel is denoted as $x[m,n]$, where m is the column index and n is the row index. Indexing starts from zero and $x[0,0]$ is the top left pixel of the image.

$$I = \begin{pmatrix} x[0,0] & x[1,0] & \dots & p[M,0] \\ x[0,1] & x[1,1] & & p[M,1] \\ \vdots & & \ddots & \vdots \\ x[0,N] & x[1,N] & \cdots & p[M,N] \end{pmatrix} \tag{2–2}$$

The channels of an image can be thought of as discrete layers forming a three dimensional array. In the multichannel case, a pixel can be seen as a vector of values from the different channels corresponding to the same location as indicated in equation (2–3). Often it is useful to consider just one channel as gray scale image. In this case a pixel can be expressed as a triple indexed variable $x[m,n,k]$, where ki is the channel index, or just as $x[m,n]$ if the channel number is clear from the context.

$$\boldsymbol{x}[m,n] = \begin{pmatrix} x[m,n,0] & x[m,n,1] & \dots & p[m,n,K] \end{pmatrix}^T \tag{2–3}$$

In figure (2–6) three different types of images are presented. They differ from one another by the number of channels and how the channels are interpreted. RGB-image has three channels representing three colors (i.e., red, green, and blue). When

displayed on screen we see a mixture or linear combination of these colors. For visual purposes all other colors can be mixed from red, green and blue. However, this does not mean that the original spectrum could be perfectly reconstructed from these three colors. Many spectra will map to the same RGB-triplet.

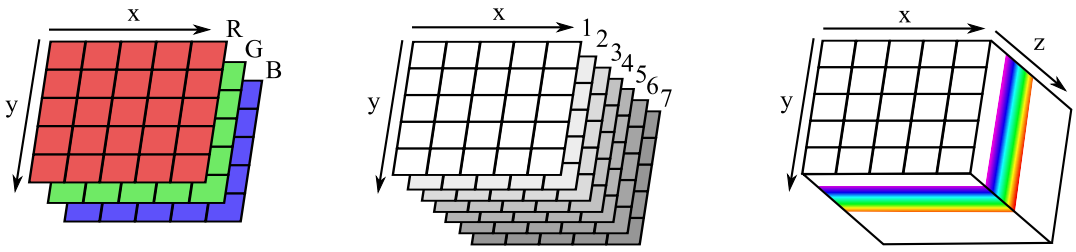

Figure 2–6. **a** **b** **c** a) Common RGB image b) Multispectral image c) Hyperspectral image

When the number of channels exceeds 3, the term *multispectral image* is used. These channels may contain the common color channels (R, G, and B), but also several others such as color infrared or panchromatic channels. Panchromatic channel has wide bandwidth and may span many other channels such as the already mentioned RGB-channels.

Although the channels of a *hyperspectral image* in figure (2–6) are presented as a continuum they are still discrete "layers". The reason for this representation is to emphasize the higher spectral resolution. When the channel count is over 100 and the channels are ordered by the wavelength, it is useful to view the hyperspectral image as a three dimensional object. There are still the two usual spatial dimensions but now the third dimension emerges from the finer spectral resolution. The third dimension presents essentially the sampled and quantized spectrum of the corresponding pixel. This is well visualized by the AVIRIS image cube in figure (2–7). It should be noted that sampling in spectral dimension may not always be uniform and a unique ordering by the wavelength may not be possible to achieve.

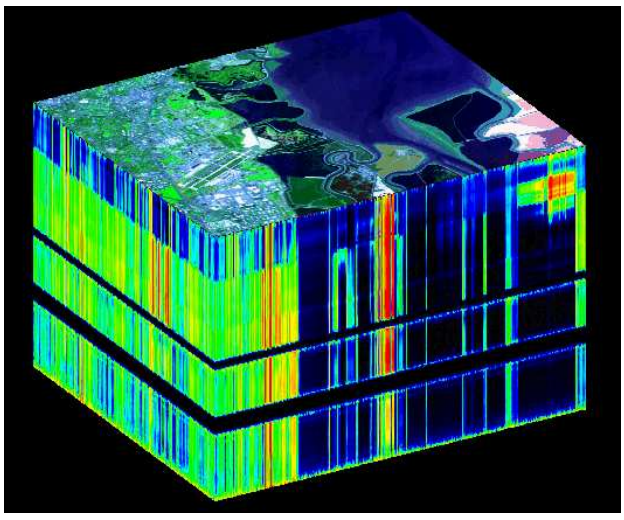

Figure 2–7. AVIRIS image cube, showing the two spatial dimension and one spectral dimension (224 channels). (NASA AVIRIS website, 2006)

As said in section (2.2), AVIRIS data has 224 channels which correspond to 224 distinct spectral bands. The width of each band is approximately 10 nm and bands range from 400 nm to 2500 nm. Bands span wide range of the spectrum with high resolution and thus atmospheric effects are strongly present. Comparing the AVIRIS image cube in figure (2–7) and the atmospheric transmission in figure (2–8) reveals that the two spectral ranges in the image cube showing in black are caused by the almost nonexistent transmission in the neighborhood of wavelengths 1.4 μm and 1.9 μm.

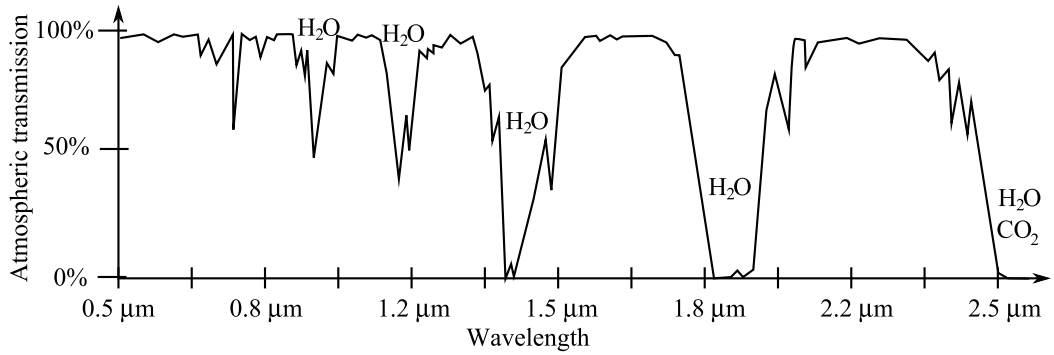

Figure 2–8. Sketch of atmospheric transmission and absorbing molecules. Adapted partially from (Scott, 1997, p. 86).

2.4 Analysis of hyperspectral images

In order to understand why lossless compression is required and why lossy compression is not adequate, it is necessary to briefly describe some methods used in the analysis of multi- and hyperspectral images. This section is mostly based on (Landgrebe, 2003). Two common tasks to be performed are *classification* and measuring of objects from the image. Typically, the former is performed automatically and the latter is done either manually or automatically after the classification.

Classification starts by extracting *features* from the image. When classifying pixels of a hyperspectral image features could be, for example, all the N values corresponding to the samples of the given N-channel pixel. If these values fulfill the criterion of a particular class the pixel is said to belong to that class. As an examples of possible classes, water and asphalt can be mentioned.

Commonly, features are represented by real valued numbers and for notational simplicity the numbers are collected to a vector. Often a feature vector is also referred to as a feature because vector can be interpreted as a single point in N-dimensional space. This space is called the *feature space*. In equation (2–3) pixel is denoted as a vector function. This is the feature that will be used as an example in this section.

Classification is divided into two main categories: unsupervised and supervised. Unsupervised classification is performed without any a priori information about the classes although the number of the classes maybe known. Supervised classification takes advantage of the known properties of the classes. Such properties could be

statistical parameters as mean and covariance.

Lets consider some simple supervised classifiers. Classifier is a function taking a feature vector parameter and returning a number which can be used as a measure in deciding into which class the vector belongs to. For every class there is one classifier. In equation (2–4) a classification rule is presented. If the classifier g_i returns the smallest or equal value among all the classifiers g_j, where j and $i \in \{1, ..., m\}$, then the feature $\boldsymbol{x}$ belongs to class i.

$$g_i(\boldsymbol{x}) \leqslant g_j(\boldsymbol{x}), \text{for all } j \in \{1, 2, ...m\} \tag{2–4}$$

The *Spectral angle mapper* (SAM), presented in equation (2–5), is a supervised classifier which takes advantage of the known mean of the class. It measures the angle between the class mean vector and the feature vector. The feature gets classified into class i when the angle between the mean of the class i and the feature vector is the smallest among all classes.

$$g_i(\boldsymbol{x}) = \cos^{-1}\left(\frac{\boldsymbol{x}^T \boldsymbol{\mu}_i}{\sqrt{\boldsymbol{x}^T \boldsymbol{x}} \sqrt{\boldsymbol{\mu}_i^T \boldsymbol{\mu}_i}} \right), \text{where } \boldsymbol{\mu}_i \text{ is the mean of the class } i \tag{2–5}$$

Another classifier is based on the Euclidean distance between the feature vector and class means. Such classifier is called the minimum distance classifier and is presented in equation (2–6).

$$g_i(\boldsymbol{x}) = (\boldsymbol{x} - \boldsymbol{\mu}_i)^T (\boldsymbol{x} - \boldsymbol{\mu}_i), \text{where } \boldsymbol{\mu}_i \text{ is the mean of the class } i \tag{2–6}$$

Minimum distance classifier classifies the feature to the class with the closest mean.

From these two examples one gets a hint why lossy compression is difficult to design so that the original classification accuracy is maintained. Decrease in classification accuracy comes from the fact that lossy compression removes some information from the image which is in some sense unnecessary. In the case of general image compression algorithms the human visual system is the judge of the necessity of the information. In the case of computational algorithms the judge is the performance of the algorithm. For example the spectral angle mapper is sensitive to the errors in the angle between the feature and class mean, but not in the Euclidean distance.

In order to perform classification equally well with both presented classification schemes one would have to maintain the distance to all the means and the angle to that of all the means. Of course in practice the sensitivity to the errors in the features depends on the data and the analysis methods, however, one can rarely optimize the lossy compression algorithm for all the possible analysis algorithms.

Our example feature was composed of the spectral characteristics of the pixel and it was noted that compression may affect the features (spectral components in our case) in a harmful way. It is also true and even more obvious that lossy compression may affect the spatial characteristic in a harmful way. This is illustrated by the image pair in figure (2–9).

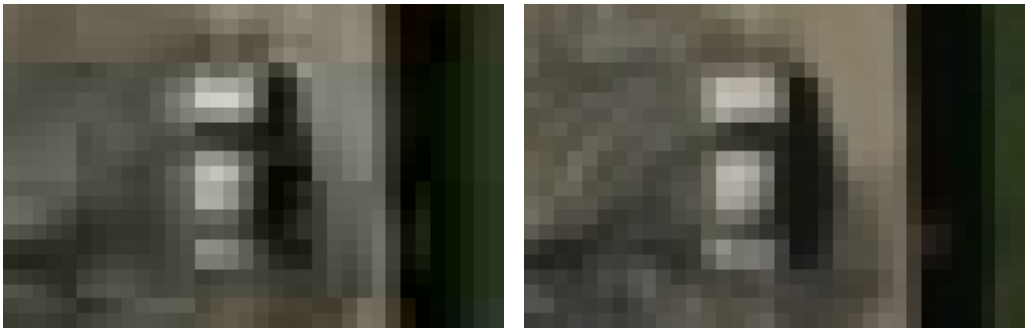

Figure 2–9. a | b a) Image with compression artifacts b) No compression artifacts (pixelation is intentional).

Consider a case where one wishes to classify cars by their color (spectral property) and then measure the width of the car. At the right in figure (2–9) one sees a gray or silver car and it is possible to measure the width of the car with moderate confidence. At the left this is not true. The car is identifiable and the width is measurable but it is hard to say whether the measured width is accurate or not because of the blocking artifacts and smearing.

3 Data Compression

Data compression can be roughly grouped into two categories: lossless and lossy compression. Lossy compression achieves often higher compression ratios when compared to lossless compression. This higher compression ratio comes with the price of loss in information. Many lossy compression algorithms are designed so that it is nearly impossible for a human to perceive the difference between the original and compressed data. However, lost information could be important in numerical data analysis as discussed in section (2.4). Aerial images are not only for visual inspection. Therefore, in this thesis I concentrate only on the lossless compression methods.

It is essential to form a basic knowledge on the theory of lossless data compression. Without this knowledge all the methods presented later would not be justified by anything more than intuition. However, intuition is often a good way for having a broader view an a subject and thus it is good to start with a short practical example.

3.1 Case study: Compression of a noisy sine wave

In figure (3–1) a noisy sampled and quantized sine wave is shown (signal $x[n]$). Signal is quantized to 5 bits and thus the signal may contain only $2^5 = 32$ different levels (or values). The possible values for the sample are -16...+15. The total number of samples in the signal is 63. From this we can calculate that the whole signal can be represented by $5\,\text{bits/samples} \cdot 63\,\text{samples} = 315\,\text{bits}$.

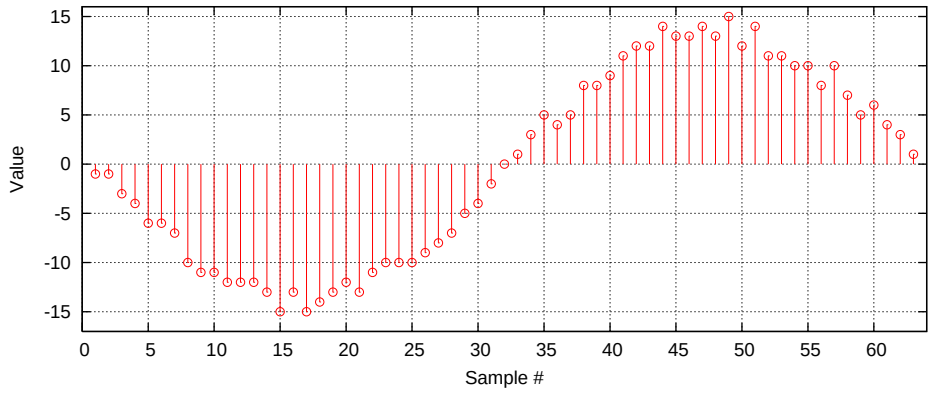

Figure 3–1. Noisy sinewave quantized to 32 distinct levels

Looking closely, figure (3–1) reveals that the signal uses only 31 levels. There is no value -16 and thus one possible quantization level is wasted. In two's complement it

is not possible to represent this particular signal more efficiently. Removing one most significant bit would mean that the levels are restricted to $-2^3 \ldots 2^3 - 1$ and this would cause information loss. Thus if two's complement representation is required then different approach has to be used for this particular signal.

One possibility is to code the first sample as it is and code all further samples as differences. This probably leads to a more compact representation because two consecutive samples are likely to have only a small difference. When the input signal is denoted as $x[n]$ and the new signal as $y[n]$ then the relation between them is

$$y[n] = \begin{cases} x[n], & n = 0 \\ x[n] - x[n-1], & n > 0. \end{cases} \tag{3–1}$$

The result of applying this transformation to the signal $x[n]$ is shown in figure (3–2).

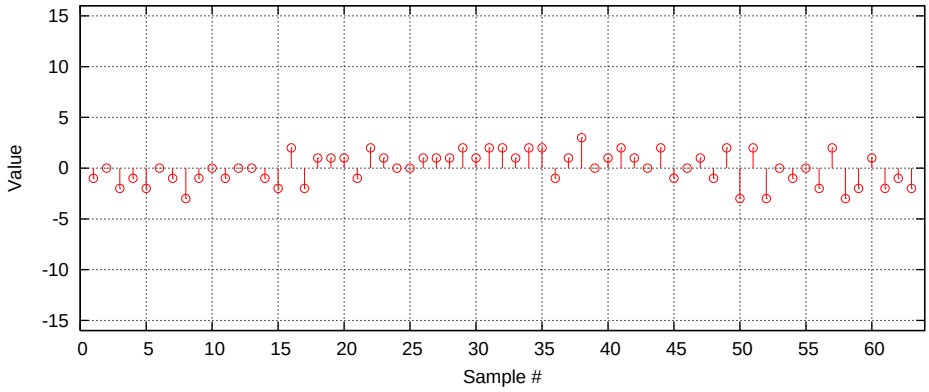

Figure 3–2. Transformed signal $y[n]$ (see equation (3–1)).

If one looks again closely at figure (3–2) one sees that only levels -3...3 are used. This means that it is possible to represent the transformed signal y with only three bits per sample ($2^3 = 8$ quantization levels) and thus the signal is compressed from 315 bits to $3\,\text{bits/sample} \cdot 63 = 189$ bits. A compression ratio of 5:3 is achieved. But is the compression lossless? To answer this it is necessary to examine how the bit sequences are coded and decoded and secondly it is necessary to verify the existence of the inverse transform for the transform in equation (3–1).

The first four samples of the original signal $x[n]$ are $\{\ -1\ \ -1\ \ -3\ \ -4\ \}$ and thus the four first samples of the signal $y[n]$ are $\{\ -1\ \ 0\ \ -2\ \ -1\ \}$. $x[n]$ can be represented a binary sequence $\{\ 1\ \ 1\ \ 1\ \ 1\ \ 1,\ \ 1\ \ 1\ \ 1\ \ 1\ \ 1,\ \ 1\ \ 1\ \ 1\ \ 0\ \ 1,\ \ 1\ \ 1\ \ 1\ \ 0\ \ 0,\ \ \ldots\ \}$ and $y[n]$ as a $\{\ 1\ \ 1\ \ 1,\ \ 0\ \ 0\ \ 0,\ \ 1\ \ 1\ \ 0,\ \ 1\ \ 1\ \ 1,\ \ \ldots\ \}$ (bits corresponding to the decimal values are delimited by commas). Assuming that the number of bits used for one sample (i.e., the bitdepth or the word length) is known in advance it is possible to uniquely decode the bit patterns into sample values.

Transform as indicated in equation (3–1) is invertible if the relation between the two signals is a bijection (i.e., there is a one to one mapping between all the possible signals $x[n]$ and $y[n]$). It can be shown that the relation is in fact a bijection. The

recursive inverse relation is

$$x[n] = \begin{cases} y[n], & n = 0 \\ x[n-1] + y[n], & n > 0. \end{cases} \tag{3–2}$$

All the calculations are made for $x[n]$, $y[n] \in \mathbb{Z}$ and only summation is present. Therefore, substituting $y[n]$ from equation (3–1) into equation (3–2) it is easy to confirm invertibility as indicated by equation (3–3).

$$\begin{aligned} x[n] &= \begin{cases} x[n], & n = 0 \\ x[n-1] + x[n] - x[n-1], & n > 0 \end{cases} \\ \Rightarrow x[n] &= \begin{cases} x[n], & n = 0 \\ x[n], & n > 0 \end{cases} \Rightarrow x[n] = x[n], n \geqslant 0 \,\square \end{aligned} \tag{3–3}$$

Thus it is verified that compression in the case of this simple example is in fact lossless. It is important to notice that the more compact representation (i.e., signal $y[n]$) has actually been obtained by modeling the signal $x[n]$. The model is in this case given by equations (3–1) and (3–2). In a sense, the chosen representation for the quantization levels, the two's complement, is also a part of the model. If the data does not fit the model compression is not necessarily achieved and the data may expand (e.g., random noise would not compress).

3.2 Statistics and entropy

Assume that in a signal some sample values are more frequent than others. For example, signal $x[n] = \{\, 1\ 0\ 0\ 1\ 1\ 4\ 0\ 0\ 2\ 1 \,\}$ has four '0's, four '1's, one '2' and one '4'. '3' does not occur. Clearly, '0' and '1' occur more frequently than '2', '3' or '4'. So in order to compress the signal it is possible to exploit this property by discarding the two's complement representation and choosing better codes for the quantization levels. Codes for the symbols should be chosen so that the total bit length of the signal is minimized for signal $x[n]$. This is called *variable length coding* (VLC). Table (3–1) shows one possible code for signal $x[n]$. It has the property that no codeword is a prefix of another codeword. Such a code is commonly called as *prefix code*.

Table 3–1. One optimal prefix code for given signal $x[n]$

Value	Binary code	Occurences	Total bit length
0	0	4	4
1	10	4	8
2	110	1	3
4	111	1	3

Total bit length of the variable length coded signal is now $4+8+3+3=18$ bits (see table (3–1)). Minimal two's complement representation takes three bits per sample (dynamic range is five levels) for the given signal $x[n]$ and thus would have taken

the total of 30 bits. Again 5:3 compression ratio is achieved. Further information about coding comes in section (3.3).

If the frequencies of the four values in the signal segment used as an example represent well the frequencies of the values of a larger set of signals one wishes to compress, they can be used to calculate the probabilities of the corresponding values. For example, value zero would have the probability of $2/5$. By making probabilistic interpretations of the sample values they are treated as random variables. Let the probability mass function of the random variable n, X_n, be $p_{X_n}(x_n)$.

In the short VLC example in table (3–1) *independent and identically distributed* (i.i.d.) random variables were assumed, although this was not explicitly said. Such random variables satisfy equations

$$p_{X_0,\ldots,X_{M-1}}(x_0,\ldots,x_{M-1}) = p_{X_0}(x_0)p_{X_1}(x_1)\ldots p_{X_{M-1}}(x_{M-1}), \tag{3–4}$$

and

$$p_{X_n}(x_n) = p_{X_m}(x_m), \forall n, m \tag{3–5}$$

and the realizations can be though of to be generated by a memoryless source. Samples from memoryless source do not depend on any other sample. Note that in equations (3–4) and (3–5) x_n is an argument of probability mass function corresponding to the random variable X_n. Generally, random variables $X_0,\ldots,X_i$ are not bound to any specific relative sample of a signal (i.e., they do not define any specific neighborhood). However, if they are bound to a specific sample, a realization of a random variable will be denoted as a sample (e.g. $x[n]$, or $x[n,m]$).

In (Shannon, 1948) a measure of uncertainty of a random variable was introduced. The measure was named *entropy*. One way to define entropy of a random variable X is

$$H(X_n) = E\{-\log_2 p(X_n)\} = -\sum_{x_n} p_{X_n}(x_n)\log_2 p_{X_n}(x_n). \tag{3–6}$$

It slightly differs from the definition given by Shannon. Here the logarithm is fixed to base 2. In this case the unit of the measure is bit. It is clear that entropy is always non-negative because the probability mass function is always non-negative and $\sum_{x_n} p_{X_n}(x_n) = 1$.

How is entropy connected to data compression? If every possible value of a random variable is assigned a specific codeword, then the lower bound for the number of bits needed to code the values of the particular random variable on the average is given by Shannon's entropy (Wang et al., 2001, p. 227). It should be noted that VLC does not in general achieve this lower bound except for in special cases. When the coder (i.e., a physical or virtual unit performing the coding of symbols) considers only one sample at a time, coding is often called *scalar coding* as, for example, in

(Wang et al., 2001, p. 227).

Entropy can be estimated from the data, for example, by assuming that $p(y) = \frac{n_y}{N}$, where n_y is the number of occurences of the sample value y and N equals to the number of samples. The entropy estimate in the case of the previous example is approximately 1.73 bits. The optimal VLC in the example achieved 1.8 bits/sample. This is 0.07 bits more than what the entropy estimate suggest as the lower bound. Even optimal prefix codes do not achieve the lower bound in every case because of some restrictions of the prefix codes discussed in section (3.3).

I.i.d. random variable is not a realistic representation of an image pixel. It is intuitively clear that a given pixel will, in general, depend on its neighborhood. In a Mth order Markov process sample $x[n]$ depends on its M predecessors $x[n-1]...x[n-M]$ (Wang et al., 2001, p. 222). This dependence is formally defined as

$$p(x[n] \,|\, x[n-1], x[n-2]...x[0]) = p(x[n] \,|\, x[n-1], x[n-2], ..., x[n-M]). \tag{3–7}$$

Equation (3–7) is for 1D signals only, however it can be expressed more generally as

$$p_{X_n \,|\, X_n^0, X_n^1 ... X_n^{N-1}}(x_n) = p_{X_n \,|\, X_n^0, X_n^1 ... X_n^{M-1}}(x_n), \tag{3–8}$$

where X_n^i is the ith member of a set of random variables corresponding to X_n (i.e., neighborhood of X_n), $\{X_n^0, ..., X_n^{N-1}\}$ is a set of all random variables excluding X_n and $N \geqslant M$.

Equation (3–8) can be interpreted so that the random variable X_n depends only on a finite set of other random variables $X_n^0, ..., X_n^{M-1}$ and does not depend on any other random variable. The set of random variables that X_n is dependent of can now be easily expressed in a way suitable for image compression. Such definitions are usually based on some neighborhood of a pixel as in figure (3–3). In fact, equation (3–8) enables one to define the dependence in any dimension as long as the random variables can be consistently enumerated. In 2D the index can be found, for example, as $n + Nm$, where n is the column index, m is the row index and N is the number of columns.

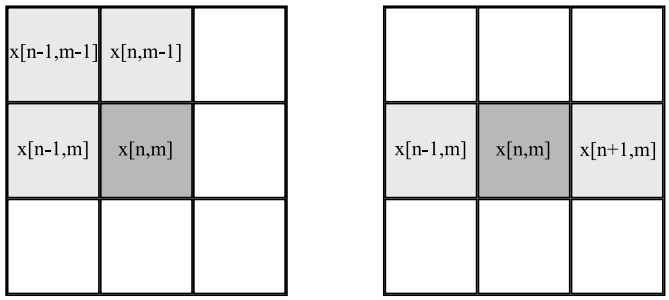

Figure 3–3. **a** **b** Two example neighborhoods a) $p(x[n,m]|x[n-1,m], x[n-1,m-1], x[n,m-1])$ b) $p(x[n,m]|x[n-1,m], x[n+1,m])$. White squares are pixels of which $x[n, m]$ is assumed not to depend on.

Conditional probabilities as defined by equation (3–8) imply $M-$dimensional joint probability mass functions (pmf) and for joint pmf *joint entropy* can be defined, in

general case, as

$$H(X_0,...,X_{M-1}) =$$

$$-\sum_{x_0}...\sum_{x_{M-1}} p_{X_0,...,X_{M-1}}(x_0,...,x_{M-1})\log_2 p_{X_0,...,X_{M-1}}(x_0,...,x_{M-1}), \quad (3\text{–}9)$$

where summations go over all values x_i are defined for and where $X_0,...,X_{M-1}$ are arbitrary random variables.

Examples of joint and marginal pmf for an aerial image are shown in figure (3–4). Joint entropy tells the lower bound for the number of bits needed on the average to jointly code random variables $X_0,...,X_{M-1}$. Joint coding is often also called *vector coding* (Wang et al., 2001).

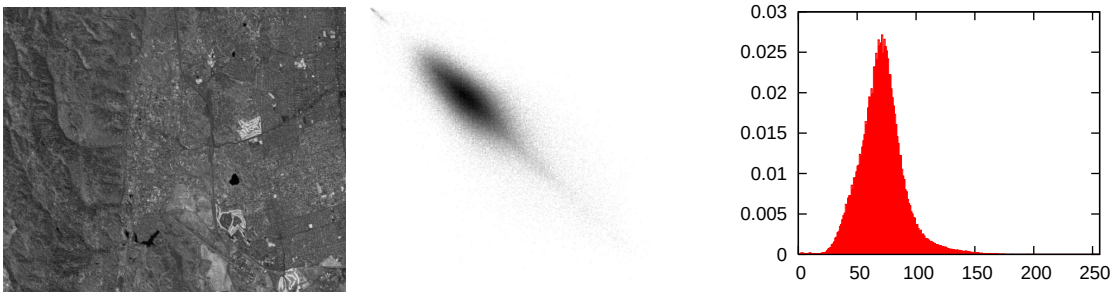

Figure 3–4. **a** **b** **c** a) Aerial image from Jasper Ridge (NASA AVIRIS website, 2006) b) estimate of the joint pmf $p(x[n,m],x[n,m-1])$ c) estimate of the marginal pmf $p(x[n,m])$ (256 bin histogram).

A very useful inequality

$$\sum_{i=0}^{M-1} H(X_i) \geqslant H(X_0,...,X_{M-1}) \quad (3\text{–}10)$$

states that joint entropy is always less than or equal to the sum of entropies of the random variables. That is, joint coding requires less or equal amount of bits on the average than coding the random variables independently. This result (i.e., equation 3–10) is simple to prove by using Gibb's inequality. The equality is true if and only if $X_0,...,X_{M-1}$ are statistically independent. In this case, joint coding does not bring any advantage over scalar coding.

Conditional entropy can be defined as

$$\begin{aligned} H(X_0|X_1,...,X_{M-1}) &= -\sum_{x_0}...\sum_{x_{M-1}} p_{X_0,...,X_{M-1}}(x_0,...,x_{M-1})\log_2 p_{X_0|X_1...,X_{M-1}}(x_0) \\ &= H(X_0,...,X_{M-1}) - H(X_1,...,X_{M-1}). \quad (3\text{–}11) \end{aligned}$$

If x_1, ..., x_{M-1} are already known then possibly something is known about X_0 as well. This knowledge can be used to reduce the uncertainty (i.e., entropy) of the

estimate for X_0. This uncertainty is quantified by conditional entropy.

A similar inequality as (3–10) can be proven for conditional entropy:

$$H(X_0) \geqslant H(X_0 \,|\, X_1, ..., X_{M-1}), \tag{3–12}$$

It states that if the coder already knows the random variables $X_1, ..., X_{M-1}$, then the random variable X_0 can be coded with less or equal amount of bits than without the knowledge of the $X_1, ..., X_{M-1}$. Again, equality is true if and only if $X_0, ..., X_{M-1}$ are independent. If they are independent, conditional coding does not bring any advantage over scalar coding. Conditional coding is usually referred to as *context-based coding*.

Furthermore, it can be shown that the lower bound for coding set by conditional entropy is lower or equal to the lower bound set by joint entropy:

$$\frac{1}{M} H(X_0, ..., X_{M-1}) \geqslant H(X_0 | X_1, ..., X_{M-1}). \tag{3–13}$$

Therefore, context-based coding can theoretically achieve better compression ratios than vector coding.

In a sense, a form of context-based coding was used in the example of section (3.1). The preprocessing stage used the knowledge of the previous sample to reduce the entropy of the coded random variable (i.e., the estimate for the next sample).

3.3 Coding

The real compression happens at the point where values are assigned to certain codewords. These codewords should be as short as possible. Entropies, defined earlier, give lower bounds to what coding can achieve. In this section constraints for coding are viewed (i.e., why do there exist a lower bound?) and some coding algorithms are presented.

Pigeonhole principle

Pigeonhole principle, also know as Dirichlet's box principle, is a simple theorem with surprisingly many applications, most notably in the field of set and number theory. The principle could be phrased as:

Let there be n pigeons and m pigeonholes. If $m < n$ then two or more pigeons have to share a common pigeonhole.

Pigeonhole principle gives some insight to lossless compression and why lossless compression has a lower bound. Let S_N be the set containing all the N bits long messages and $S_{<N}$ be the set of all the messages shorter than N bits. Messages can be seen as N bits long vectors or alternatively as N-tuples (important is that the representation is ordered). Now it is easy to count the cardinality of the sets:

$\#S_N = 2^N$ and $\#S_{<N} = 2^N - 2$ (exluding the message with 0 length).

Say that all N bits long messages should be compressed losslessly. This means that a bijection should be established between messages in S_N and those in $S_{<N}$. By pigeonhole principle it is known that some messages from S_N will map to a common message in $S_{<N}$, because $S_{<N}$ has smaller cardinality than S_N. Thus the compression is not a bijection and it follows that it is impossible to losslessly compress every message from the S_N regardless of the compression algorithm. This can be interpreted so that the model defines the set of messages from S_N that will be compressed. Messages that do not fit the model are not important and they may expand.

Kraft's inequality

After compression the borders which divide the bit sequences into separate codewords are not generally known beforehand. Therefore, codewords have to be constructed in such way that the borders can be deduced from the bit sequence itself. One method is to use so called prefix-codes. Prefix-codes have the property that a given code is not a prefix of another code. Prefix-codes were shortly introduced in section (3.2) and an example of such a code can be seen in table (3–1).

It is intuitively clear that the codeword lengths cannot be just blindly selected and assumed that there exist a prefix-code. For example, if there are 5 codewords, it is not possible to find a prefix-code for which they are all 2 bits long. This is clear because there is only 4 codewords with length 2. *Kraft's inequality* provides necessary and sufficient conditions for the existence of a prefix-code. If the lengths of the codewords satisfy Kraft's inequality:

$$\sum_x 2^{-l_x} \leqslant 1, \tag{3–14}$$

where $l_x \in \mathbb{N}_0$ is the length of the codeword for symbol x, then it is possible to find a prefix-code with those lengths. For the example code in table (3–1) the sum of equation (3–14) evaluates to 1 and thus the code satisfies the equality. It is not possible to come up with a shorter code as the sum would be unavoidably larger than 1. For a proof of the Kraft's inequality see, for example, (Hankerson et al., 1998, p. 62).

Coding is essentially an optimization problem: minimize the total length of the message with respect to the constraint posed by Kraft's inequality. Coder assumes or estimates the probabilities for certain symbols (and thus for certain codewords). The total length of the message can be estimated from the expected length of the codeword. Expected length of the codeword is

$$l = E\{l_X\} = \sum_x p(x) l_x, \tag{3–15}$$

where l_x is the length of the codeword for symbol x and $p(x)$ is the pmf (compare (3–15) and (3–6)).

Every coded symbol will increase the length of the message by l bits on average. Based on equations (3–15) and (3–14) the optimization problem can now be defined formally as

$$L_X = \min_{\sum_x 2^{-l_x} \leqslant 1} \arg \sum_x p(x) l_x, \tag{3–16}$$

where the constraint is Kraft's inequality defined in (3–14) and L_X is the set of optimal codeword lengths for the given pmf $p(x)$.

Huffman coding

Solution to the problem stated in equation (3–16) is given by *Huffman codes*. The procedure for generating Huffman codes was introduced in (Huffman, 1952). The algorithm starts from the original sorted symbol probabilities (see figure (3–5)). At every step, two least probable symbols are merged. To solve the ambiguity between the two merged symbols, bit 0 is attached to one of the symbols and bit 1 is attached to the other symbol. The resulting probabilities are again sorted and the algorithm starts over. The algorithm stops when the probability of a symbol reaches 1. The attached bits form the codewords.

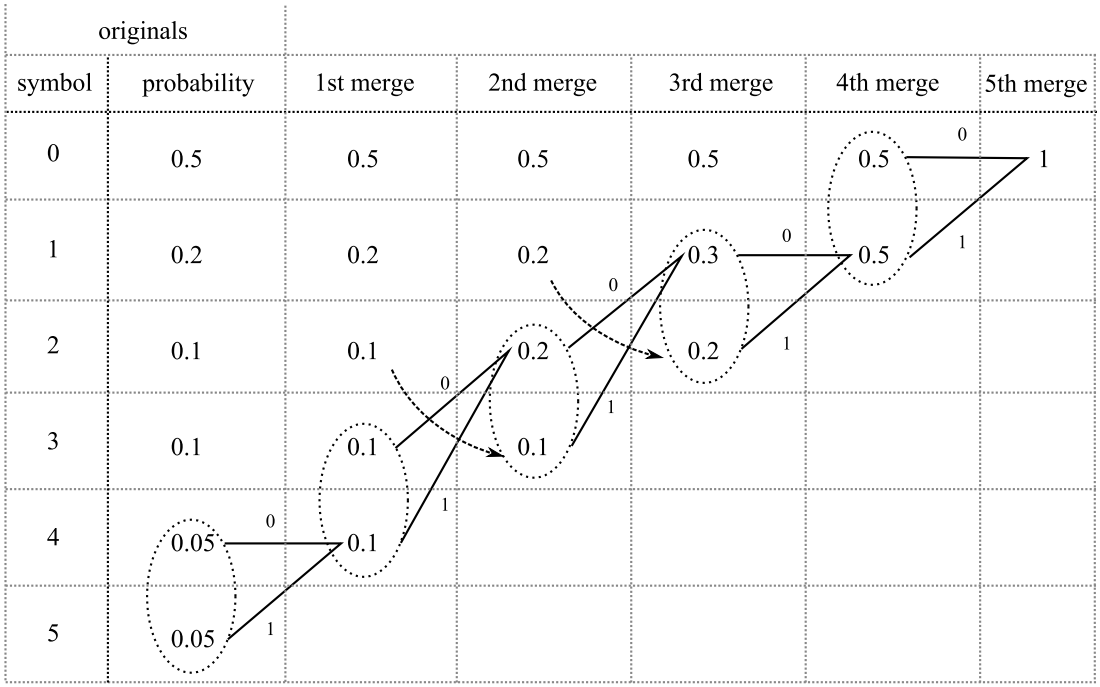

Figure 3–5. Construction of the Huffman-tree

In figure (3–5) the curved arrows mark change of symbol order due to sorting. The codewords can be read by starting from the root of the tree and inserting 0 or 1 to the codeword, depending on the taken branch and stopping at the particular leaf of the tree. For example, codeword for symbol 2 is found by following the lower branch once, then taking the upper branch and finally the lower branch once more

("101"). Note that the symbol order has changed, so although we ended up at the fourth row, it actually corresponded to symbol 2. Full set of codewords for the given diagram can be seen in table (3–2). Note that codewords in table (3–2) are not the only possible set of codewords for the given probabilities. The bits can be selected arbitrarily for the branches. Therefore, in this particular case there exists $2^5 = 32$ possible Huffman codes. All these codes can be uniquely encoded and decoded as long as both ends know which particular code is used.

Table 3–2. Huffman-codes for figure (3–5)

symbol	probability	word length	codeword
0	0.5	1	0
1	0.2	2	11
2	0.1	3	101
3	0.1	4	1000
4	0.05	5	10010
5	0.05	5	10011

Expected length of the codeword is $0.5+2\cdot 0.2+3\cdot 0.1+4\cdot 0.1+5\cdot 0.05+5\cdot 0.05=2.1$ bits (see table (3–2)). Thus, if it is known in the end of encoder that the next symbol X_n has the given pmf (or conditional pmf) then the coder needs to use only 2.1 bits on the average to code the symbol. Entropy for the symbols of the above example is $H(X)\approx 1.43$ bits, which is approximately half a bit less than what Huffman-coding achieved.

It can be shown that the entropy lower bound is achieved if the probabilities are of the form $p(x)=2^{-l_x}$. For example, consider the constraint required for pmf

$$\sum_x p(x)=1\,, \tag{3–17}$$

which in this case can be written as

$$\sum_x 2^{-l_x}=1\,. \tag{3–18}$$

Comparing equation (3–18) to Kraft's inequality (i.e., equation (3–14)) reveals that the summations are identical. Thus for pmf of the form $p(x)=2^{-l_x}$, l_x is a valid length of a codeword for prefix coding.

Now consider the expected length of codewords and entropy:

$$l=E\{l_X\}=\sum_x 2^{-l_x}l_x\,;\;\; H(X)=-\sum_x 2^{-l_x}\log_2 2^{-l_x}=l. \tag{3–19}$$

They are equal and thus codeword lengths l_x form the optimal set of codeword lengths. As a conclusion, Huffman codes achieve the lower bound given by entropy when $p(x)=2^{-l_x}$.

Once the set of codewords is known coding and decoding are straightforward. For coding symbol x insert the corresponding codeword into the bit sequence. Decoding takes advantage of the fact that Huffman codes are prefix codes. For example, bit sequence { 1 0 0 0 1 1 1 0 0 1 0 1 1 1 0 0 1 1 } is Huffman-coded with codewords in table (3–2). After the first bit is read the correct symbol could be anything from 1 to 5. Second bit restricts the set of values to 2...5. When fourth bit is read it is clear that the first symbol has to be 3. Next symbol is decoded in similar fashion. Correct symbol sequence is { 3 1 4 1 5 }.

Golomb coding

Huffman coding has one significant drawback: code generation becomes impractical with high symbol counts. In (Golomb, 1966) a novel run-length coding scheme was introduced. Run-length is a term referring to count for repeating consecutive symbols. The need for *Golomb coding* arises from the fact that run-lengths can have infinite lengths. Therefore, it is computationally impractical to construct Huffman codes for run-lengths. Golomb codes have closed form solutions for the codewords (unlike Huffman) and thus their construction is highly efficient.

Run-lengths often follow geometric distribution

$$p(x) = pq^x, x = 0, 1, 2, ... \tag{3–20}$$

The rationale for considering Golomb coding in the context of lossless image compression comes from the fact that prediction errors often approximately follow two sided geometric distribution if the symbol count is large.

Although Golomb codes were originally designed for run-lengths, they have applications in other fields of lossless compression, especially when Huffman codes are too expensive to be used. Hyperspectral data has often bitdepth of 16 bits. This correspond to 2^{16} symbols. Thus symbol count is high and closed form solutions for the codewords are very convenient.

The lowest computational complexity of useful Golomb codewords is attained with the Golomb-Rice codes, which require only bitshifts and summation. These codes are a subset of the Golomb codes and will be treated in detail in the following.

Golomb codes can be divided into two parts. Those parts will be called as *unary* and *remainder* part. The unary part is formed from the non-negative integer u defined as

$$u = \lfloor x/m \rfloor \overset{\text{Golomb-Rice}}{=} \lfloor x \cdot 2^{-b} \rfloor \tag{3–21}$$

with integer parameter m which is of the form 2^b in the case of Golomb-Rice coding. The remainder part is formed from the non-negative integer r defined as

$$r = x - u \cdot 2^b \tag{3–22}$$

In equations (3–21) and (3–22) x denotes the non-negative integer to be coded.

The unary part is formed from u by concatenating u 1-bits and a single 0-bit. This is called *unary coding*. The decoding of the unary part is straightforward: read the data bit by bit and count all the 1-bits until the first 0-bit is encountered. The remainder part is stored simply as the binary representation of the non-negative number r. It is simple to show that r requires at most b bits to be stored. Decoding can simply be done by reading the b bits and treating those as b bits long integer. The original integer x can be found simply as:

$$x = r + u\,2^b. \tag{3–23}$$

The complete code is formed of $u+1$ bits long unary part followed by b bits long remainder part. The dependence of the codeword length on x is

$$l_x = b + \left\lfloor x \cdot 2^{-b} \right\rfloor + 1. \tag{3–24}$$

When the data follows geometric distribution defined in equation (3–20) the optimal parameter m is

$$m = -\log_2 q. \tag{3–25}$$

First 6 Golomb-Rice codewords for $b = 2$ or equivalently $m = 4$ are given in table (3–3), where data is assumed to be from geometric pmf.

Table 3–3. Golomb-Rice codes for parameter $b = 2$ and corresponding geometric distribution with $p = 1 - \left(\frac{1}{2}\right)^{1/4}$

x	**probability**	**codeword**
0	0.159	000
1	0.134	001
2	0.113	010
3	0.095	011
4	0.080	1000
5	0.079	1001
6	0.067	1010

As real world data rarely follows the assumed geometric distribution perfectly, it is useful to see what is the optimal value for b for the given data, in the sense of compression ratio. The cost function $\mathcal{J}(b)$ can be constructed as

$$\mathcal{J}(b) \;=\; \sum_{i=1}^{N} \frac{l_{x_i}}{N}$$

$$= \frac{1}{N}\sum_{i=1}^{N} b + \lfloor x_i \cdot 2^{-b} \rfloor + 1$$
$$= b + 1 + \sum_{i=1}^{N} \frac{\lfloor x_i \cdot 2^{-b} \rfloor}{N}. \qquad (3\text{–}26)$$

$\mathcal{J}(b)$ is the average length of the codeword for a given data $x_1 ... x_N$. The approximation

$$\hat{\mathcal{J}}(b) = b + 1 + 2^{-b} \sum_{i=1}^{N} \frac{x_i}{N} \qquad (3\text{–}27)$$

is done to obtain a differentiable function.

In the case of large sample counts (i.e., high N) $\hat{\mathcal{J}}(b)$ is a very good approximation of $\mathcal{J}(b)$. Especially when one considers the location of the global minimum. This is visualized in figure (3–6). In figure $\mathcal{J}(b)$ is plotted as a continuous function of b, although b should be an integer. This is done to show the justification for the method used for searching the minima (i.e., $\hat{\mathcal{J}}$ is smooth in b). The point of global minimum for $\mathcal{J}(b)$ is denoted as b_{opt} and the point of global minimum for $\hat{\mathcal{J}}(b)$ is denoted as $\hat{b_{\text{opt}}}$. The minimum of $\hat{\mathcal{J}}(b)$, $\hat{b_{\text{opt}}}$, is an estimate of b_{opt}.

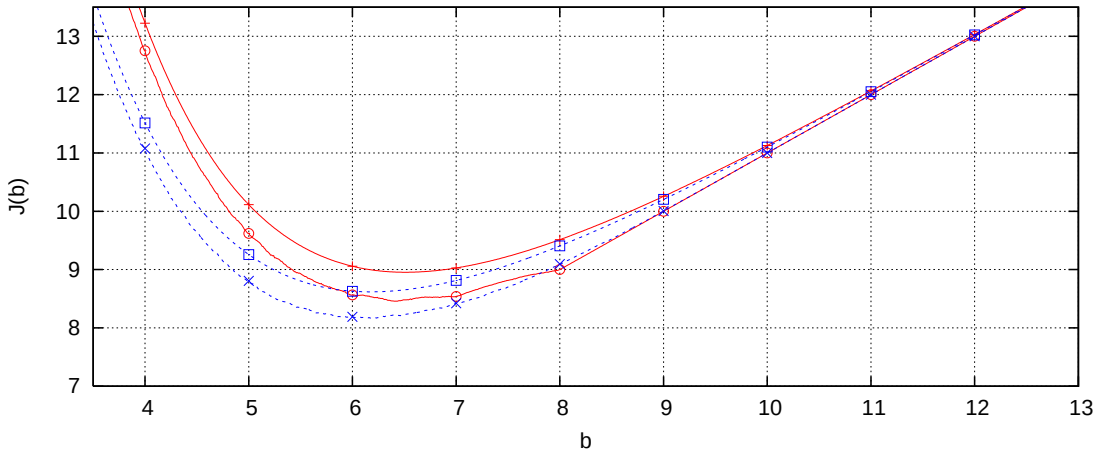

Figure 3–6. Cost functions $\mathcal{J}$ (marked by circles and ticks) and $\hat{\mathcal{J}}$ (marked by crosses and boxes) as a function of b. Red line is for uniformly distributed data and blue dash line is for geometrically distributed data (1000 data points). Both data sets have equal entropies (approximately 8 bits).

The partial derivative of equation (3–27) is

$$\frac{\partial}{\partial b}\hat{\mathcal{J}}(b) = 1 - 2^{-b} \ln 2 \sum_{i=1}^{N} \frac{x_i}{N}$$
$$= 1 - 2^{-b} \ln 2 \hat{\mu} \qquad (3\text{–}28)$$

Solving the partial derivative for zero gives the point at which the equation (3–27) reaches its global minimum ($\hat{b}_{\text{opt}}$). The solution is

$$\begin{aligned} \frac{\partial}{\partial b}\hat{\mathcal{J}}(\hat{b}_{\text{opt}}) &= 0 \\ \Rightarrow 2^{-\hat{b}_{\text{opt}}}\ln 2\hat{\mu} &= 1 \\ \Rightarrow 2^{\hat{b}_{\text{opt}}} &= \ln 2\hat{\mu} \\ \Rightarrow \hat{b}_{\text{opt}} &= \log_2(\ln 2\hat{\mu}) \\ &= \log_2(\ln 2) + \log_2\hat{\mu} \end{aligned} \tag{3–29}$$

where the average of the data is denoted as $\hat{\mu}$. In the case of Golomb-Rice coding the value of b has to be a non-negative integer and thus $\hat{b}_{\text{opt}}$ has to be rounded. If it is not desirable to evaluate either $\mathcal{J}(b)$ or $\hat{\mathcal{J}}$(b) in order to decide the direction of the rounding, $\hat{b}_{\text{opt}}$ should be rounded up. This is because $\hat{\mathcal{J}}$ grows linearly when b gets larger and exponentially when b gets smaller than the $\hat{b}_{\text{opt}}$. Note also the subtle irregularities of $\mathcal{J}(b)$, especially in the case of uniformly distributed data.

Golomb coding works only for non-negative numbers. Fortunately this restriction is easy to fix. There are at least two simple methods for this. One could code the sign-bit separately and use Golomb coding for the absolute value. Alternatively, one could re-map the values so that negative values map to positive odd values and positive values map to non-negative even values.

Arithmetic Coding

Arithmetic coding is actually not a variable length coding as it does not code symbols to distinct codewords. Instead, arithmetic coder codes the whole ensemble of symbols to a single codeword which represents a value in the half-open interval [0, 1). Thus, arithmetic coding is not restricted by Kraft's inequality in the same way as Huffman coding is. Therefore, it is possible for an arithmetic coding to approach closer to the theoretical lower limit posed by the entropy.

The idea of arithmetic coding is to represent the message as an number in the half-open interval $[0, 1)$ with as few bits as possible. The message could be represented as

$$c = \frac{x}{2^L} = b_1 2^{-1} + b_2 2^{-2} \ldots + b_L 2^{-L}, \tag{3–30}$$

where $x \in \mathbb{N}_0 \wedge 0 \leqslant x < 2^L$ and $b_n \in \{0, 1\}$.

Thus the message will be coded as a fixed-point representation of c. Some fixed-point representations of fractions are shown in table (3–4). It is seen that some fractions are representable with less bits than others. Arithmetic coder finds the fraction that lies in a certain given interval and has the shortest possible representation.

Table 3–4. Some fixed-point representations for different fractions.

L	3		2		1	
c	x	fp	x	fp	x	fp
0	0	.000	0	.00	0	.0
1/8	1	.001	-	-	-	-
1/4	2	.010	1	.01	-	-
3/8	3	.011	-	-	-	-
1/2	4	.100	2	.10	1	.1

Coding starts from the interval [0, 1). First this interval is partitioned into N disjoint subintervals where N is the number of possible symbols. Every subinterval corresponds to a certain symbol. The size of the subintervals are chosen to be (at least roughly) proportional to the probabilities of each symbol. When a symbol is coded, a subinterval corresponding to the symbol is divided further into subintervals. Again the subintervals are chosen to be proportional to the probabilities of the next symbol.

For example, subinterval partitions for the bit sequence { 0 0 1 0 } are done as in figure (3–7). In this example the probabilities are assumed to be 2/3 for 0 and 1/3 for 1. The final interval is in this case [19/26, 49/81) which has the length of approximately 0.12. If the symbols can be assumed to be from. i.i.d. source the length of the final interval is the probability of the message. The "shortest" c is in this case $5/2^3$ and therefore the fixed-point bit representation is { 1 0 1 }. Thus compression ratio of 4:3 is achieved. Note that Huffman coding in its basic form could not have achieved any net compression as there were only two distinct symbols.

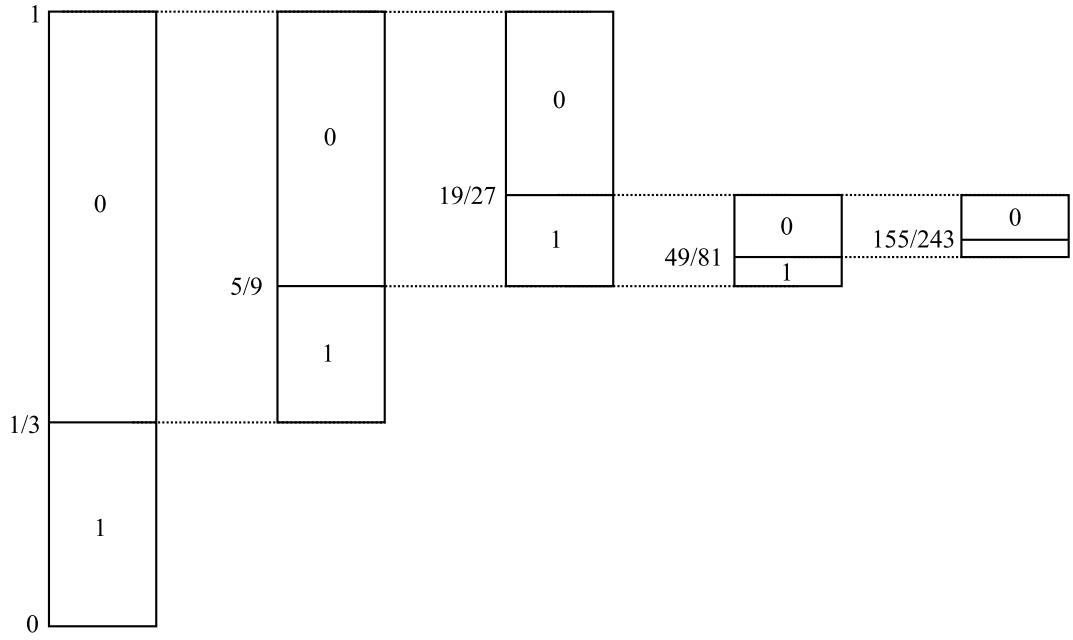

Figure 3–7. Subinterval partitioning in arithmetic coding

In practise the coding does not happen as in the example. In the example it seems that all symbols have to be coded into an interval before any bits can be written.

In fact, bits can be output as table (3–5) suggests. In table, the known bits are the highest bits of the fraction that are known at a given point. First bit can be written after the second symbol is coded (i.e., when the interval is $[5/9, 1)$). This is because every $c \in [5/9, 1)$ will have 1 as the most significant bit (see table 3–5). After the fourth symbol the second bit can be written (upper and lower limits have two highest bits in common). As the fourth symbol was also the last symbol, the last bit can be written. The last bit has to be chosen so that the resulting fraction will fall into the correct interval ($[49/81, 19/27)$).

Table 3–5. Known bits for intermediate intervals (sequential encoding)

current symbol	-	0	0	1	0
current interval	$[0, 1)$	$[1/3, 1)$	$[5/9, 1)$	$[19/27, 1)$	$[49/81, 19/27)$
upper limit	.1111...	.1111...	.11111...	.1111...	.1011...
lower limit	.0000...	.0101...	.10001...	.1011...	.1001...
known bits	-	-	.1	.1	.10

Decoding could proceed by interpreting the whole codeword as an fraction and constructing the subintervals as in figure (3–7). From there the symbols can be read by noting into which subintervals the given fractions falls. However, often sequential decoding is preferred. That way decoding can start before all of the data is received (as with VLC coding). Before all the bits are available it is possible to approximate the fraction by the already received bits as in equation (3–31).

$$\hat{c} = \frac{y}{2^N} = \lfloor c2^N \rfloor 2^{-N} = b_1 2^{-1} + ... b_N 2^{-N}, \tag{3–31}$$

where N is the number of received bits less than L. It is clear that $\hat{c} \leqslant c$, however, it can be shown that

$$\hat{c} \leqslant c < \hat{c} + 2^{-N} \tag{3–32}$$

This result can be used to judge into what subinterval the c actually falls.

In the previous example the first two symbols can be decoded by using the inequality as in table (3–6). Last two symbols can be decoded by knowing that .101 is actually the c. In this example the total number of symbols in the message was so small that the information about the number of symbols would have expanded significantly the resulting file. With large number of coded symbols this should not be a problem. An alternative approach is to use a low probability symbol marking the end of the file. When the subinterval of this additional symbol is reached decoding can be stopped. This approach is especially useful in online transmissions where the length of the message is not necessarily known beforehand.

Table 3–6. Sequential decoding

received bits	.1	.10	.101
N	1	2	3
$\hat{c} + 2^{-N}$	1	3/4	3/4
$\hat{c}$	1/2	1/2	5/8
known symbols	0	0	00

The arithmetic coding described here is called *dyadic fraction with least denominator* in (Hankerson et al., 1998). It can be shown that the average length L of the codeword has an upper limit as indicated in equation (3–33).

$$L < NH(X) + 1, \tag{3–33}$$

where N is length of the message. For the proof of this result see (Hankerson et al., 1998, p. 131-132). There are many subtleties to arithmetic coding, and especially to implementation, which are not discussed here. Care must be taken when limited precision is used to calculate the intervals. The reader can refer to (Hankerson et al., 1998, p. 119-154) for more thorough introduction to the subject.

Arithmetic coding is a highly efficient method when the entropy of the message is low. However, for high entropy messages Huffman or Golomb coding might work sufficiently well. Furthermore, the estimates of the probabilities are almost never exact and thus the upper limit in equation (3–33) does not hold. Arithmetic coding may require more computational resources than, for example, Golomb coding (depending on the platform) and thus when there is no significant advantage from the arithmetic coding VLC methods are preferable.

4 Decorrelation

4.1 Short introduction to digital filtering

In this section basic concepts of discrete-time digital filtering are established. The purpose of this section is to define basic terms and notations used afterwards. Only topics essential for this thesis are discussed.

Linear time-invariant (LTI) filters can be divided into two categories: FIR and IIR filters or filters with *finite-duration impulse response* and filters with *infinite-duration impulse response*, respectively. The term linear refers to how the filters behave as operators. Filters are linear if and only if

$$\mathrm{T}\{a\,x[n] + b\,y[n]\} = a\,\mathrm{T}\{x[n]\} + b\,\mathrm{T}\{y[n]\}, \tag{4–1}$$

where T is an operator performing the filtering operation for the signals $x[n]$ and $y[n]$. Symbols a and b refer to two arbitrary scalars. Roughly it could be said that the term time-invariant means that the effect of the filter does not depend on the time instant of its application. Filter is time invariant if and only if

$$h[n] = \mathrm{T}\{\delta[n]\} \text{ and } h[n-k] = \mathrm{T}\{\delta[n-k]\}, \forall k. \tag{4–2}$$

FIR

In discrete-time domain filtering with an FIR filter is identical to convolving the input signal $x[n]$ with the coefficient sequence of the filter $b[n]$:

$$y[n] = \sum_{k=0}^{K-1} b[k]x[n-k]. \tag{4–3}$$

Sometimes it is easier to use vector notation to write the convolution as:

$$y[n] = \boldsymbol{b}^T\boldsymbol{x}[n], \tag{4–4}$$

where

$$\begin{aligned} \boldsymbol{b}^T &= \begin{pmatrix} b[0] & b[1] & \dots & b[K-1] \end{pmatrix}, \\ \boldsymbol{x}[n]^T &= \begin{pmatrix} x[n] & x[n-1] & \dots & x[n-K+1] \end{pmatrix}. \end{aligned}$$

IIR

By IIR filters usually those defined by the following difference equation are considered:

$$\sum_{k=0}^{L-1} a[k]y[n-k] = \sum_{k=0}^{K-1} b[k]x[n-k], \tag{4–5}$$

where $a[k]$ and $b[k]$ are sequences defining the coefficients of the filter, and sequences $x[n]$ and $y[n]$ are the input and output signals respectively. If all coefficients $a[k]$ are 0 (except $a[0]$), difference equation reduces to the equation (4–3) (i.e., IIR filter reduces to FIR filter).

It is possible for an IIR filter to be unstable. A stable filter produces bounded output for any bounded input. Filter satisfying this is called BIBO stable. Sufficient condition for unstability is that the impulse response of the filter is unbounded.

Impulse response

An LTI system is uniquely defined by its impulse response. Impulse response of a filter can be identified by "feeding" it a unit impulse:

$$\delta[n] = \begin{cases} 1, & n = 0 \\ 0, & n \neq 0. \end{cases} \tag{4–6}$$

For an FIR filter the impulse response will be finite-duration and identical to its coefficients:

$$h[n] = \sum_{k=0}^{K-1} b[k]\delta[n-k] = b[n], \tag{4–7}$$

For an IIR filter the impulse response will be infinite in duration and defined by the following recursive difference equation:

$$h[n] = \frac{b[n] - \sum_{k=1}^{L-1} a[k]h[n-k]}{a[0]} \tag{4–8}$$

System function

System function of a causal filter is defined through unilateral z-transform of its impulse response:

$$H(z) = \sum_{n=0}^{\infty} h[n]z^{-n}, \text{where } z = re^{j\omega}. \tag{4–9}$$

System function can be solved directly from the difference equation using following-properties of the z-transform: z-transform is a linear operator and convolution in time domain corresponds to product in z-transform domain. Thus, z-transform of the difference equation (4–5) is:

$$A(z)Y(z) = B(z)X(z). \tag{4–10}$$

The system function is then defined as:

$$H(z) = \frac{Y(z)}{X(z)} = \frac{B(z)}{A(z)} = \frac{\sum_{n=0}^{K-1} b[n]z^{-n}}{\sum_{n=0}^{L-1} a[n]z^{-n}}. \tag{4–11}$$

Thus system function of an LTI filter defined by difference equation (4–5) is a rational function. Inverse z-transform of the system function would yield the impulse response of the filter.

FIR filters are often characterized by the zeros of the system function $H(z)$. Zeros are the solutions for the equation $H(z) = 0$. If a zero is close to unit-circle (i.e., $z = e^{i\omega}$) it results in strong attenuation in the frequency response (see below). A filter with zeros of $z = e^{i0} = 1$ is a highpass filter and a filter with zeros of $z = e^{i\pi} = -1$ is a lowpass filter (assuming absence of other zeros).

Frequency response

Complex valued frequency response of a filter is defined as the Fourier transform of the impulse response:

$$F(\omega) = \sum_{n=0}^{\infty} h[n]e^{-j\omega n}, \tag{4–12}$$

Frequency response can be easily expressed using filters system function:

$$F(\omega) = H(z)|_{z=e^{j\omega}} = H(e^{j\omega}) = \frac{\sum_{n=0}^{N-1} b[n]e^{-j\omega n}}{\sum_{n=0}^{N-1} a[n]e^{-j\omega n}}. \tag{4–13}$$

From equation (4–13) the amplitude response is found as the absolute value of $H(e^{j\omega})$ and phase response as the argument of $H(e^{j\omega})$.

4.2 Linear prediction

Linear prediction tries to model a sample from its neighbors as a linear combination (i.e., weighted sum). In the context of lossless coding, the neighborhood has to be

causal. Causality implies order. Order, in this thesis, is the order at which the coded samples are available at the decoder. Often samples of an image are transfered column by column or row by row. However, more complex orderings may also be used, such as zig-zag ordering. Thus equations presented here are not written using the usual notation for 1D or 2D sample data, but using a more general notation x_n^i, where i is an index to a member of the neighborhood of the sample x_n. x_n may refer to any sample from k-dimensional data. For example, mapping to 2D data may be formed as $x_{n+Nm} \leftrightarrow x[n,m]$, where N is the number of columns in the image. This is similar notation used earlier in section (3.2).

Linear predictors discussed in this thesis are defined as

$$\hat{x}_n = \boldsymbol{w}^T \boldsymbol{x}_n, \tag{4–14}$$

where $\boldsymbol{w}$ is a $M \times 1$ column vector containing the predictor coefficients and

$$\boldsymbol{x}_n^T = \begin{pmatrix} x_n^0 & x_n^1 & \dots & x_n^{M-1} \end{pmatrix}. \tag{4–15}$$

Linear predictor cannot model a given arbitrary signal perfectly. Thus the prediction error is formed:

$$e_n = x_n - \hat{x}_n = x_n - \boldsymbol{w}^T \boldsymbol{x}_n, \tag{4–16}$$

where x_n is the current sample and e_n is the prediction error or the residual. Linear predictor system is an FIR filter with the first coefficient always 1. In this thesis linear predictor refers to a filter the output of which is the predicted value (see equation (4–14)) and linear prediction system refers to a filter the output of which is the prediction error (see equation (4–16)).

Inverse filter is defined as

$$x_n = e_n + \boldsymbol{w}^T \boldsymbol{x}_n. \tag{4–17}$$

This is an IIR filter. Fortunately in the case of lossless coding the inverse operation is guaranteed to be stable with respect to output of the predictor system. However, in order to guarantee this, the effects of limited precision calculations have to be canceled by the use of rounding. In general, the following forward and inverse operations are guaranteed to be stable:

$$\begin{aligned} e_n &= x_n - \lfloor f(\boldsymbol{x}_n) \rfloor, \\ x_n &= e_n + \lfloor f(\boldsymbol{x}_n) \rfloor, \end{aligned}$$

where e_n and $x_n \in \mathbb{Z}$ and $\lfloor \cdot \rfloor$ denotes rounding towards negative infinity. Function f can apply any arbitrary operation on $\boldsymbol{x}_n$ and rounding will guarantee that the result is also in $\mathbb{Z}$. In the case of linear prediction f is inner product as in equation (4–4).

Note that rounding causes subtle nonlinearities to the prediction. Furthermore, it should be noted that sometimes linear prediction is presented from the generative model point of view. In such case the inverse filter and the predictor filter change roles.

Polynomial predictors

A common subset of predictors is formed by the so called polynomial predictors. The operation of the polynomial predictor filter is equal to polynomial extrapolation (see figure (4–1)). Predictor coefficients for the first three polynomial predictors can be seen in table (4–1). The residual or prediction error represents the difference between the extrapolated sample and the actual sample. If the predicted signal follows locally some polynomial with degree less than the degree of the predictor, the residual will be 0. In figure (4–1) sample 1 is predicted from the previous samples with 3 different predictors (see table (4–1)). The signal in figure does not exactly follow any polynomial of order 2 or less.

Table 4–1. 3 first polynomial predictors

$\boldsymbol{w}$	Polynomial order	Prediction
(1)	0	same as the previous sample
(2 −1)	1	on the same line as the two previous samples
(3 −3 1)	2	on the same parabola as the three previous samples

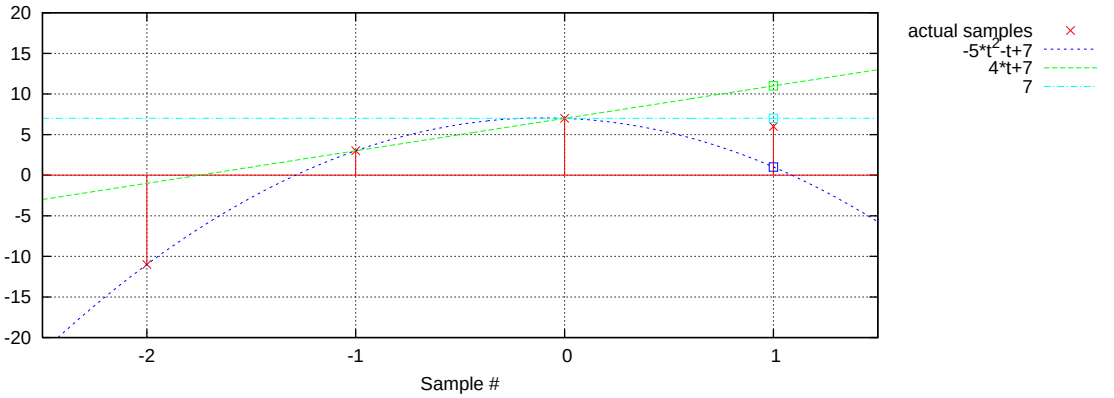

Figure 4–1. Example of polynomial fitting to data and extrapolation

Polynomial predictors are interesting not only because they are computationally cheap but also because they are "true" predictors. True in the sense that they have negative group delays (i.e., group advance) and 0 dB attenuation close to 0 frequency (see figure (4–2)). More precisely the group delay is -1 samples and thus the correct sample is predicted. Group delay is, roughly speaking, a term referring to the frequency dependent delay of the signal in samples. It is calculated as $\mathrm{grd}(\omega) = -\frac{d\varphi(\omega)}{d\omega}$, where $\varphi(\omega)$ is the phase response (see (Oppenheim et al., 1999)).

These two properties result in high attenuation for polynomial predictor system when frequency is below about 1 radians (see figure (4–3)). In many real world signals the frequency content is weighted towards low frequencies. The system function

of a polynomial predictor system has N zeros of $z = 1$, where N is the predictor order. However, higher order predictors also cause stronger gain at high frequencies, thus the worst case residual gets larger when the order increases as illustrated in figure (4–3).

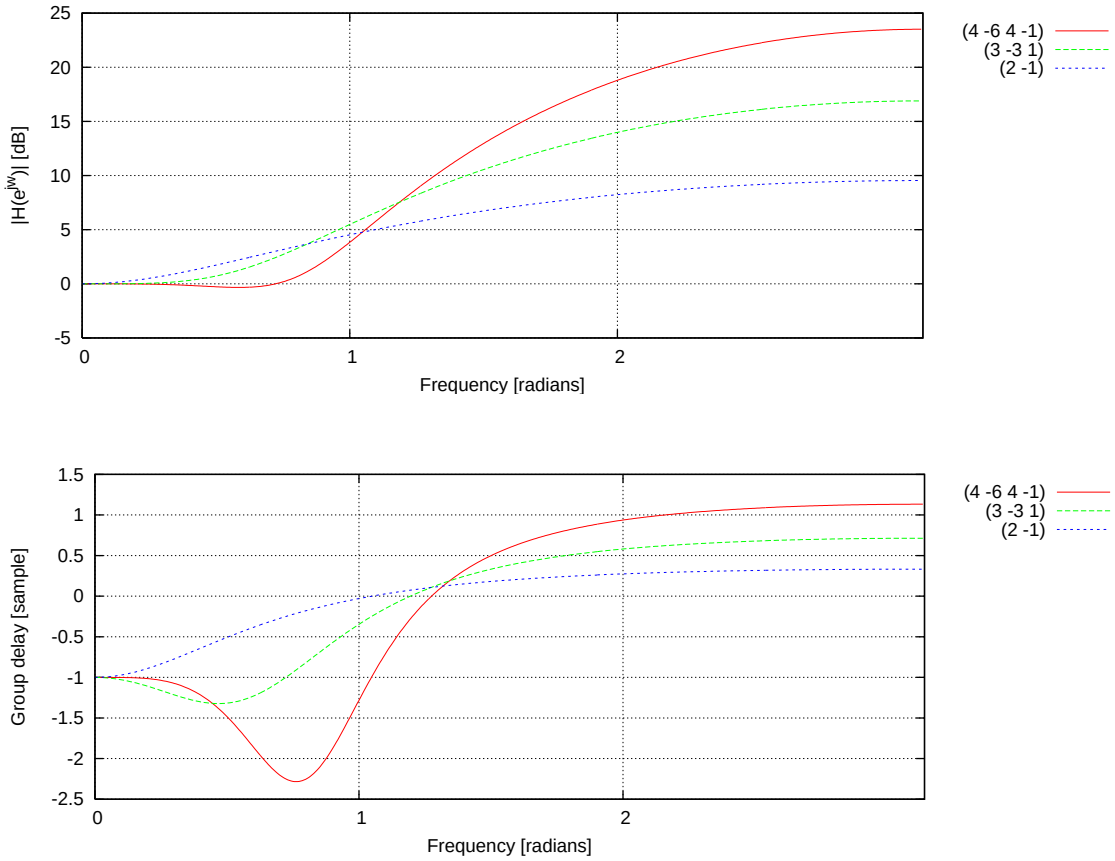

Figure 4–2. Amplitude responses and group delays for three polynomial predictors of order 1, 2, and 3.

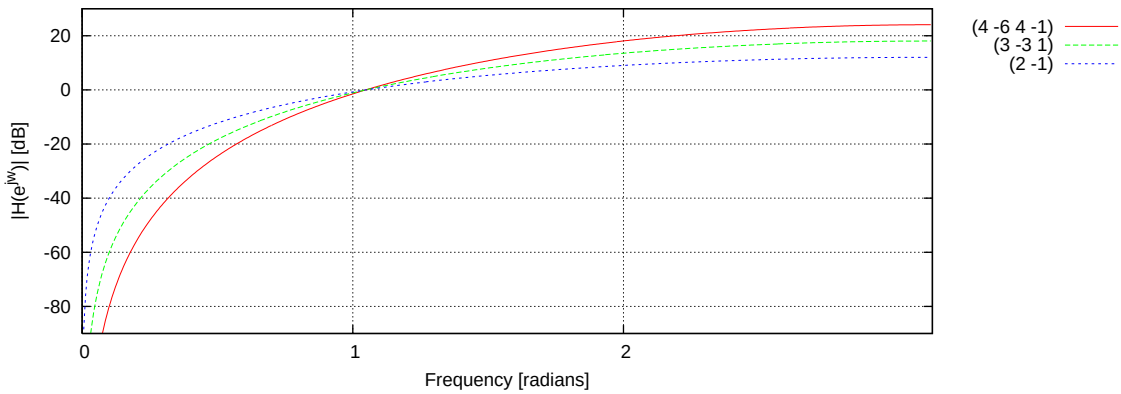

Figure 4–3. Amplitude response of the predictor system for the three predictors of order 1, 2, and 3.

Optimal linear prediction

In the context of lossless coding, optimal linear predictor would minimize the entropy of the residual. At first, such task sounds impossible as nothing can reduce the true entropy of a signal. However, in section (3.2) several different entropies were discussed. Particularly, conditional entropy of a signal was found to be smaller than its scalar entropy (or 0th order entropy). Thus, scalar entropy coder cannot, in general, achieve as high efficiency as a context-based entropy coder. However, linear prediction as a preprocessing stage can reduce scalar entropy. This is not contradictory as scalar entropy is not the true entropy. It is only an estimate and serves as an upper limit for the true entropy.

New signal (i.e., the residual) is generated by linear prediction which has entropy of

$$H(X_n - \left[\boldsymbol{w}^T \boldsymbol{X}_n\right]) = H(E_n) = -\sum_{e_n} p_{E_n}(e_n) \log_2 p_{E_n}(e_n), \tag{4–18}$$

where X_n is a scalar random variable, $\boldsymbol{X}_n$ is an M-dimensional vector random variable (the neighborhood of X_n) and $p_{E_n}(e)$ is the pmf of the residual. In the case of optimal prediction the conditional entropy should not change (i.e., the prediction should not increase conditional entropy) and thus the following equation holds:

$$H(E_n|\boldsymbol{X}_n^T) = H(X_n|\boldsymbol{X}_n^T). \tag{4–19}$$

Moreover, as the residual will be coded with a scalar entropy coder, it would be beneficial if the following holds:

$$H(E_n) = H(E_n|\boldsymbol{X}_n^T). \tag{4–20}$$

Equation (4–20) states, that the neighborhood does not contain any information about the residual. If equations (4–19) and (4–20) hold, scalar entropy of the residual equals to conditional entropy of the original signal:

$$H(E_n) = H(X_n|\boldsymbol{X}_n^T), \tag{4–21}$$

and scalar entropy coder together with the predictor would perform the task of context-based entropy coder.

However, linear prediction, in general, does not achieve the optimality in the entropy sense (i.e., the equality in equation (4–21)). Equation (4–20) states that E_n and $\boldsymbol{X}_n^T$ should be independent. Linear prediction cannot guarantee independence, but it can guarantee uncorrelatedness, which is a weaker requirement. Independence implies uncorrelatedness, but uncorrelatedness does not imply independence.

Common way of finding the predictor coefficients $w[k]$ is to minimize the *mean square error* (MSE). MSE is defined as $\frac{1}{N+1}\sum_{n=0}^{N-1}(x[n] - \hat{x}[n])^2$. It turns out that

minimizing the MSE provides uncorrelatedness. Squared prediction error is

$$\sum_n e_n^2 = \sum_n (x_n - \boldsymbol{w}^T \boldsymbol{x}_n)^2, \tag{4–22}$$

where summation goes over all indices n and m. Minimizing squared error is obviously equal to minimizing MSE because scaling does not affect the position of the minima.

In order to minimize equation (4–22), gradient is taken with respect to $\boldsymbol{w}$ and set to zero. Gradient of equation (4–22) is:

$$\begin{aligned} \sum_n \frac{\partial}{\partial \boldsymbol{w}} (x_n - \boldsymbol{w}^T \boldsymbol{x}_n)^2 &= \boldsymbol{0}, \\ \Rightarrow -2 \sum_n (x_n - \boldsymbol{w}^T \boldsymbol{x}_n) \boldsymbol{x}_n^T &= \boldsymbol{0}, \\ \Rightarrow -2 \sum_n x_n \boldsymbol{x}_n^T + 2\boldsymbol{w}^T \sum_n \boldsymbol{x}_n \boldsymbol{x}_n^T &= \boldsymbol{0}, \\ \Rightarrow 2\boldsymbol{w}^T \boldsymbol{R} - 2\boldsymbol{r}^T &= \boldsymbol{0}, \end{aligned} \tag{4–23}$$

where

$$\begin{aligned} \boldsymbol{R} &= \sum_n \boldsymbol{x}_n \boldsymbol{x}_n^T, \\ \boldsymbol{r} &= \sum_n x_n \boldsymbol{x}_n^T. \end{aligned}$$

MSE optimal coefficients are found by solving the linear system:

$$\boldsymbol{w} = \boldsymbol{R}^{-1} \boldsymbol{r}, \tag{4–24}$$

which requires that $\boldsymbol{R}$ is not singular. From the derivation of equation (4–23) it is possible to see that the MSE optimal predictor decorrelates e_n and $\boldsymbol{x}_n$:

$$\begin{aligned} \sum_n (x_n - \boldsymbol{w}^T \boldsymbol{x}_n) \boldsymbol{x}_n^T &= \boldsymbol{0}, \\ \Rightarrow \sum_n e_n \boldsymbol{x}_n^T &= \boldsymbol{0}. \end{aligned} \tag{4–25}$$

4.3 Wavelet transform

Wavelet transform decomposes the signal into components that could be described as the details and approximation of the original data. The most important property of the wavelet transform, in the context of this thesis, is the multiresolution representation of the signal it provides. Wavelet transform allows to view the signal at different resolutions. Roughly speaking this means that it is possible see both

the forrest and the trees at the same time. This way it is possible to catch the small details and the long trends simultaneously. Second important property of the wavelet transform is that there exist very fast algorithms for implementing it.

Image pyramid

Lets consider the idea of multiresolution decomposition through image pyramids before going into the theory of wavelets. In (Burt and Adelson, 1983) Laplacian pyramid was introduced for image compression. It is defined through the so called Gaussian pyramid. The idea of the Gaussian pyramid is simple: approximate the image at successively lower resolutions. In Laplacian pyramid the differences between the levels of the Gaussian pyramid are stored. The differences are calculated between the interpolated lower resolution image at level $n-1$ and the image at the level n. If the size of the original image is $2^J \times 2^J$ then its size at the next level is $2^{J-1} \times 2^{J-1}$ and so on until at the last level we have image of size 1×1 (Gonzalez and Woods, 2002, p. 351). The bottom (or the first) level of the Gaussian pyramid is the original image and thus the Gaussian and the Laplacian pyramids both double the amount of the pixels when J approaches infinity:

$$\sum_{i=0}^{J} \frac{2^{2(J-i)}}{2^{2J}} = \sum_{i=0}^{J} \frac{2^{2J}2^{2i}}{2^{2J}} = \sum_{i=0}^{J} 2^{-2i} = 2 - 2^{-J}. \tag{4–26}$$

Because of the more compact representation Laplacian pyramid still often achieves net data compression. The more compact representation is achieved by the pixel-to-pixel decorrelation which results in reduced variance and entropy (Burt and Adelson, 1983).

Figure 4–4. Four levels of the Gaussian pyramid on top and four levels of Laplacian pyramid at the bottom.

Laplacian pyramid captures the details at different resolutions and the common features for all levels are stored only in the final approximation (see figure (4–4)). Reconstruction happens by adding the details to the approximation until the original image is perfectly reconstructed. An example of the Gaussian and Laplacian pyramids is shown in figure (4–4).

Subband decomposition

Unfortunately Laplacian pyramid, while flexible, is hindered by the expansion in the number of pixels. One solution is to use wavelet transform. Wavelets are proposed in (Mallat, 1989) as a tool for *multiresolution analysis* (MRA) of the images. Moreover, the connection to the quadrature mirror filters and filter banks are made. In the following dyadic subband decomposition is briefly discussed.

In the following, up and down sampling operators are extensively used. Up sampling and down sampling refer to operations which rise and lower the sample rate, respectively. In time domain up sampling by M can be represented as

$$x_{\mathrm{up}}[n] = \begin{cases} 0, & n \neq kM, \forall k \in \mathbb{Z} \\ x[n/M], & n = kM, \forall k \end{cases} \tag{4–27}$$

and down sampling by M can be represented as

$$x_{\mathrm{down}}[n] = x[nM]. \tag{4–28}$$

For more information on up and down sampling see (Akansu and Haddad, 2001, p. 114–123).

In figure (4–5) a simple two band filter bank is illustrated. $\tilde{h}$ and h are the lowpass filters and $\tilde{g}$ and g are the highpass filters. Between the impulse responses and the system functions of the filters, the following relations hold: $\tilde{h}[n] \overset{z}{\leftrightarrow} \tilde{H}(z)$, $h[n] \overset{z}{\leftrightarrow} H(z)$ $\tilde{g}[n] \overset{z}{\leftrightarrow} \tilde{G}(z)$, and $g[n] \overset{z}{\leftrightarrow} G(z)$. Filters with tilde are called analysis filters and filters without tilde are called synthesis filters.

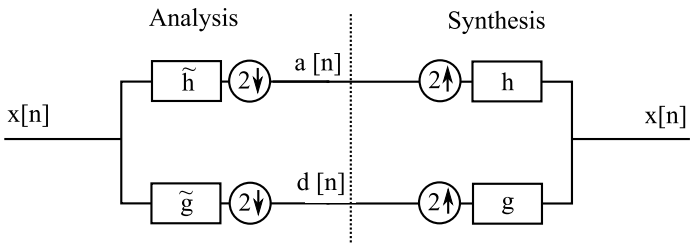

Figure 4–5. Two band lossless filter bank.

For reasons that will become more evident later, the samples of the lowpass band are denoted as $a[n]$, where the 'a' refers to 'approximation', while the samples of the highpass band is denoted by $d[n]$, referring to 'detail'. The number '2' and the arrow down in the circle denotes down sampling by two and conversely, arrow up denotes up sampling. Because we are dealing with lossless compression, the signal coming out of the filter bank is bit by bit accurate to the original signal $x[n]$. The compression itself would happen at the dotted line.

Not all filters lead to perfect reconstruction filter banks (i.e., filter banks that in the absence of round off errors reconstructs the original signal perfectly). Perfect reconstruction is guaranteed when the constraints:

$$G(z) = \tilde{H}(-z), \tag{4–29}$$
$$H(z) = -\tilde{G}(-z), \tag{4–30}$$

and

$$G(z)\tilde{G}(z) \text{ is a halfband filter} \tag{4–31}$$

are met (Strang and Nguyen, 1996, p. 105–106 and 211). These constraints force the filter pairs $\{\tilde{h}, \tilde{g}\}$ and $\{h, g\}$ to be quadrature mirror filters and thus their frequency responses are mirror images about $\omega = \pi/2$ of each other (see figure (4–6)). In this thesis $\tilde{h}$ is a lowpass filter and from the constraints it follows that $\tilde{g}$ has to be a highpass filter. For more information about QMF filters see (Akansu and Haddad, 2001, p. 135–136).

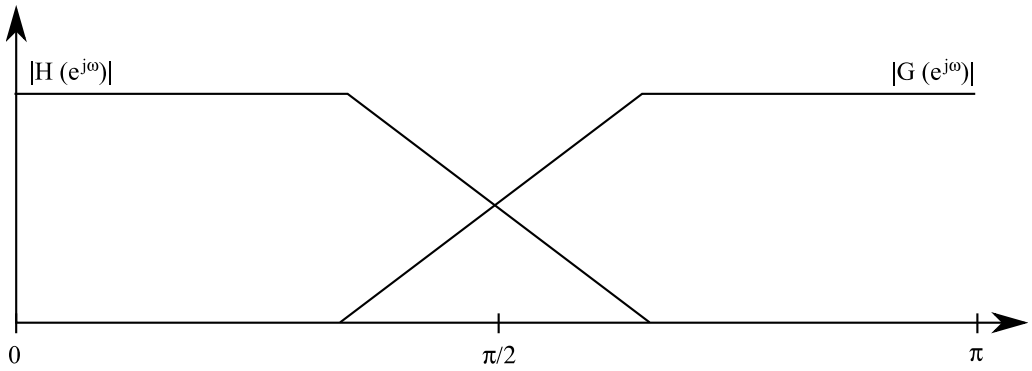

Figure 4–6. Sketch of amplitude responses of a QMF pair illustrating the mirror property.

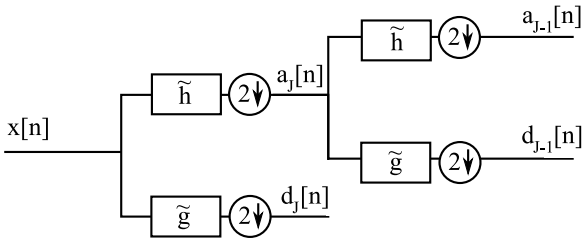

Figure 4–7. Two levels of the dyadic filter bank.

In this thesis, discussion is restricted to dyadic filter banks as they will lead to the algorithm called the fast wavelet transform. In figure (4–7) two levels of such a filter bank are shown. The idea is that the filters are arranged in a binary tree which branches only from the lowpass band. $a_j[n]$ refers here to the lowpass or approximation signal at the level j and $d_j[n]$ is the detail or highpass signal at the level j. When the structure in figure (4–7) is continued to level j, the original signal is divided into octave bands.

From the subband decomposition point of view, $d_J[n]$ ideally contains the information from the band $\omega = [\pi/2, \pi]$, $d_{J-1}[n]$ from the band $\omega = [\pi/4, \pi/2)$ and $d_{J-j}[n]$ from band $\omega = [2^{-j-1}\pi,\ 2^{-j}\pi)$. The final lowpass or the approximation signal contains the rest of the signal, i.e., the band $\omega = [0, 2^{-j-1}\pi)$. In reality the bands overlap with each other. One should note also that the sampling frequency halves at each level. This leads to the conclusion that sample at level $j-1$ corresponds to twice as long interval than the sample at the level j. Thus when when one goes further in dyadic decomposition the interval corresponding to a sample gets longer and the frequency band gets narrower.

Wavelet multiresolution analysis

The difference between wavelets and subband decomposition with QM filters is in the background theory. From the implementation point of view both appear alike. The theory of wavelets is strongly tied to the analysis of the continuous time functions.

In the context of image compression, and especially this thesis, *multiresolution analysis* (MRA) is important. MRA uses different time, or space, scales to analyze a given function. For a more indepth introduction to MRA, in context of wavelet transform, see (Mallat, 1989) or (Jawerth and Sweldens, 1994). Roughly multiresolution analysis could be based on the following assumptions (Fliege, 2000, p. 252-253):

There exist subspaces $V_j,\ j \in \mathbb{Z}$, such that

$$V_{-\infty} \subset \ldots \subset V_0 \subset V_1 \subset V_2 \subset \ldots \subset V_\infty = L_2(\mathbb{R}), \tag{4–32}$$

where the basis of V_j has finer time resolution than the basis of V_{j-1} and $L_2(\mathbb{R})$ is a set of functions that satisfy $\int_{-\infty}^{\infty} |f(x)|^2 dx < \infty$ (i.e., they have finite energy). There exists a set of scaling functions,

$$\varphi_k(x) = \varphi(x-k) \in V_0, k \in \mathbb{Z}, \tag{4–33}$$

that are translates of each other and span the set V_0:

$$V_0 = \text{span}\{\varphi_k(x)\}. \tag{4–34}$$

The translates are required to be orthogonal to each other:

$$< \varphi_k, \varphi_l > = \delta[k-l], l \in \mathbb{Z}, \tag{4–35}$$

where $< f, g >$ is a short hand for: $\int_{-\infty}^{\infty} f(x)g^*(x)dx$, such that $f(x), g(x) \in L_2(\mathbb{R})$, and where $*$ denotes complex conjugate.

If a set of functions span a space, then all functions in the space can be represented as a linear combination of those functions. And finally, functions between adjacent spaces are connected via two-scale property:

$$f(x) \in V_j \Leftrightarrow f(2x) \in V_{j+1}. \tag{4–36}$$

That is, functions in set V_{j+1} have finer time resolution than functions in V_j. One should also note that equation (4–32) implies that the basis of V_{j+1} also spans V_j, but not vice versa. Thus it holds that

$$\varphi(x) = 2\sum_k h[k]\varphi(2x-k), \tag{4–37}$$

for some sequence $h[k]$. Equation (4–37) is often called the *refinement equation*, the *dilation equation*, or the *two-scale difference* equation. For sequence $h[k]$ it holds:

$$\sum_k h[k] = 1, \tag{4–38}$$

which can be seen by integrating both sides of the refinement equation (4–37). It turns out that the sequence $h[k]$ defines the $\varphi(x)$.

So far only scaling functions have been discussed. Wavelet functions emerge from set W_j complementing V_j in V_{j+1}. That is, the basis of V_j and W_j together span V_{j+1}:

$$V_{j+1} = V_j \oplus W_j, \tag{4–39}$$

where $\oplus$ is the direct sum operator. It immediately follows that

$$\bigoplus_j W_j = L^2\{\mathbb{R}\}. \tag{4–40}$$

Functions $\psi(x-k) = \psi_k(x)$ spanning W_0 and satisfying similar conditions as $\varphi(x)$ (i.e., equations (4–33) and (4–35)) are called wavelet functions. Space W_j contains the details that are missing from the lower resolution space V_j with respect to V_{j+1}. Because wavelet space W_0 is in scaling space V_1 (see equation (4–39)) it holds that

$$\psi(x) = 2\sum_k g[k]\varphi(2x-k), \tag{4–41}$$

for some sequence $g[k]$ and similarly to the scaling function, the wavelet function is defined by the sequence $g[k]$. Wavelet and scaling functions are from now on denoted as

$$\varphi_{j,k}(x) = \varphi(2^j x - k), \tag{4–42}$$
$$\psi_{j,k}(x) = \psi(2^j x - k), \tag{4–43}$$

where j refers to scale and k to translation.

Orthogonal wavelets

Wavelet transform is called orthogonal if (Jawerth and Sweldens, 1994)

$$W_0 \perp V_0 \tag{4–44}$$

or equivalently

$$< \psi_{0,0}, \varphi_{0,l} > = 0. \tag{4–45}$$

Now orthogonal wavelet transform can be defined as

$$d_j[k] = < f, \psi_{j,k} >, \tag{4–46}$$

where f is the analyzed function in $L_2(\mathbb{R})$ and $d_j[k]$ is a sequence of detail coefficients for level j. That is, function f is projected to space W_j. Larger the j the finer are the details captured by the wavelet transform. Reconstruction, or inverse wavelet transform, is

$$f(x) = \sum_{j,k} d_j[k]\psi_{j,k}(x), \text{ where } j, k \in \mathbb{Z}. \tag{4–47}$$

Thus the function can be reconstructed from its details. If one has performed the scaling transform at level J:

$$a_J[k] = < f, \varphi_{J,k} >, \tag{4–48}$$

the reconstruction can be started from level J:

$$f(x) = \sum_k a_J[k]\varphi_{J,k}(x) + \sum_{j,k} d_j[k]\psi_{j,k}(x), \text{ where } j,k \in \mathbb{Z} \text{ and } j > J. \tag{4–49}$$

The sequences $a_j[k]$ are called approximation coefficients. V_I contains the approximation of the function f and the details are retrieved from spaces W_j:

$$L_2(\mathbb{R}) = V_J \oplus W_J \oplus W_{J+1} \oplus W_{J+2} ... \oplus W_\infty \tag{4–50}$$

Biorthogonal wavelets

Orthogonality provides many beneficial properties, such as that, energy of the detail coefficients being proportional to the energy of the signal (Parseval's theorem). However, orthogonality is also very restrictive from the design point of view. Thus, biorthogonal wavelets were introduced as a generalization. They have a few drawbacks, such as that, in general, Parseval's theorem no longer holds. Biorthogonal MRA introduces two new spaces called dual scaling subspace and dual wavelet subspace: $\tilde{V_i}$ and $\tilde{W_i}$. They have to satisfy the following properties:

$$\tilde{V_i} \perp W_i, \tag{4–51}$$
$$V_i \perp \tilde{W_i}, \text{and} \tag{4–52}$$
$$\tilde{W_i} \perp W_j, \text{for} \, i \neq j. \tag{4–53}$$

For dual spaces there exists the dual scaling function $\tilde{\varphi}_{i,k}$ and dual wavelet function $\tilde{\psi}_{i,k}$. They are defined by the sequences $\tilde{h}[k]$ and $\tilde{g}[k]$, respectively, through the refinement equation. The biorthogonal wavelet transform is defined as:

$$d_j[k] = < f, \tilde{\psi}_{j,k} >, \tag{4–54}$$

and the inverse transform as:

$$f(x) = \sum_{j,k} d_j[k]\psi_{j,k}(x), \text{ where } j,k \in \mathbb{Z}. \tag{4–55}$$

Notice that the analysis is performed by the dual wavelet function and the synthesis is performed by the wavelet function. However, the role of the functions are interchangeable. The reconstruction can be started from any level as with orthogonal wavelets, but the approximation coefficients have to be calculated using the dual scaling function.

Filterbank implementation

In (Mallat, 1989) an algorithm was introduced for calculating the wavelet transform using scaling coefficients $a_j[k]$. This algorithm is now known as Mallat's pyramid algorithm or the fast wavelet transform. It turns out that after the initial approximation coefficients are calculated, all the wavelet coefficients at coarser levels can be calculated based on those coefficients. That is, there is no need to evaluate the integral in the wavelet transform for every level and translate.

In practice one usually has neither the approximation coefficients nor the original continuous function. Approximation coefficients could be estimated from the sampled data, but often this is not done. Furthermore, in the context of lossless coding, this estimation would be difficult to implement in a lossless manner. Thus, common practice is to treat the sampled data as the approximation coefficients from level $J+1$.

After the approximation coefficients are obtained, the fast wavelet transform proceeds exactly as a dyadic filter bank (see subband decomposition in this section). The filter coefficients are those used to define the wavelet and scaling functions (i.e., the sequences $\tilde{h}[k], h[k], \tilde{g}[k]$ and $g[k]$). In figure (4–8), $x[n]$ is the discrete time signal to be transformed, corresponding to $a_{J+1}[n]$. Here the dual filters (i.e., the sequences defining the dual wavelet and scaling functions) are on the analysis side. The synthesis side uses sequences $h[k]$ and $g[k]$ as filter coefficients.

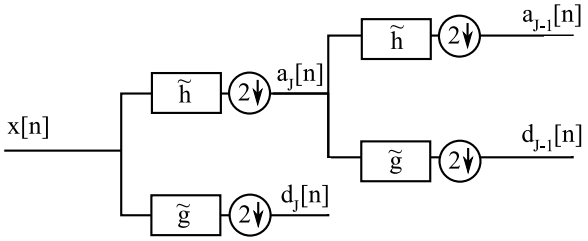

Figure 4–8. Two levels of the dyadic filter bank implementing the fast wavelet transform.

It turns out that the requirements for biorthogonal MRA are equivalent to the requirements for QMF filter bank (compare equations (4–51)-(4–53) to equations (4–29)-(4–31)).

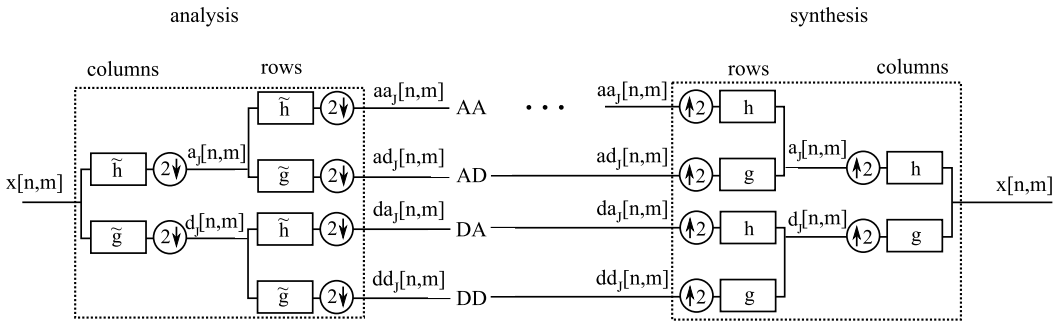

Figure 4–9. Separable 2D fast wavelet transform

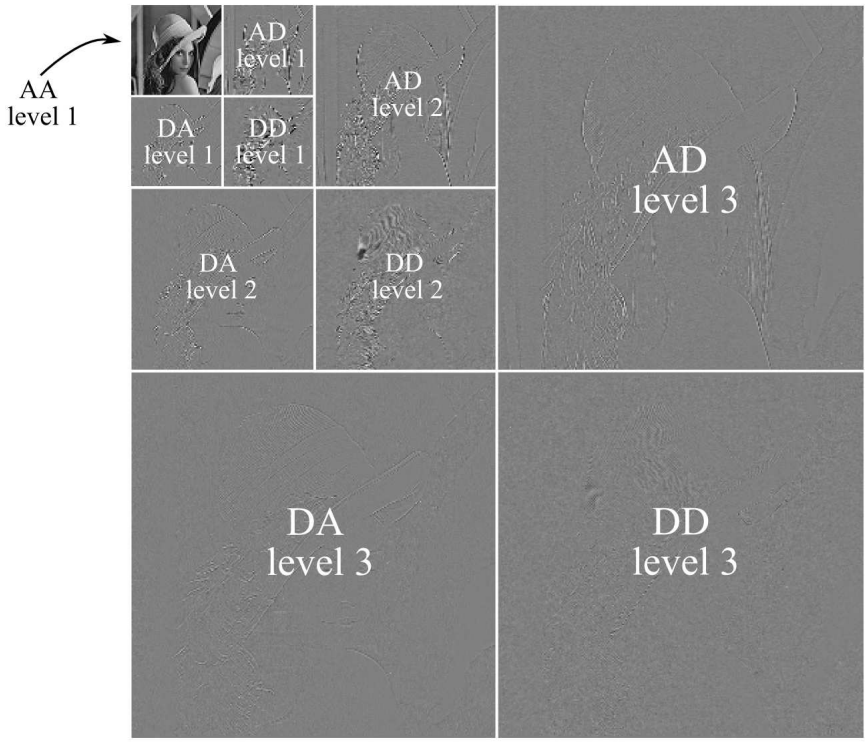

Figure 4–10. Example wavelet decomposition of the Lena image into approximation part and detail parts.

For 2D data, such as images, the fast wavelet transform is implemented similarly to 2D Fourier transform. That is, image is first transformed column by column (i.e., treating a column as an independent 1D signal) and then the result row by row as illustrated in figure (4–9). Instead of two bands or parts, 2D wavelet transform creates 4 parts per level: AA, AD, DA and DD. The ellipsis in figure marks the branch where further decompositions may occur. AA is the approximation part while AD, DA and DD are the detail parts of the image at a given level. In figure (4–9) the dashed block marks the part of the filter bank which implements one level of the transform.

AD part captures the vertical details, DA part captures the horizontal details and DD part captures the diagonal details. This can be seen in the example wavelet decomposition in figure (4–10) (compare to Laplacian pyramid in figure (4–4)).

Compression

Wavelets are well suited for compression as they provide more compact representation of the data. In figure (4–10) the mid-grey corresponds to 0 (in the AA part black corresponds to 0). It can be seen that most of the coefficients are very close to 0 in this particular example. It turns out that the properties of wavelets are close to those of fixed polynomial predictors. That is, if the signal locally follows some polynomial the detail coefficient will be small.

Wavelets are often characterized by their vanishing moments. These are related to the property mentioned above. The p-order moment of the wavelet function is defined as (Jawerth and Sweldens, 1994):

$$\mathcal{N}_p = \int_{-\infty}^{\infty} x^p \psi_{0,0}(x) dx,\ p \in \mathbb{N}_0, \tag{4–56}$$

$\tilde{\mathcal{N}}_p$ is defined similarly for the dual wavelet. A desired property of a wavelet function is that the first N moments vanish (i.e., are equal to 0). Note that the first moment equals to the average. Wavelet function has N vanishing moments if

$$\mathcal{N}_p = 0, \quad \text{for} \quad 0 \leqslant p < N \qquad \text{and} \qquad \mathcal{N}_N \neq 0. \tag{4–57}$$

Thus

$$< P, \psi_{j,k} > = 0, \tag{4–58}$$

if $P(x)$ is a polynomial of degree less than N (translation and scaling do not affect the order of a polynomial). It turns out that for a wavelet function to have N vanishing moments also the sequence $g[k]$, has to have N vanishing moments (Jawerth and Sweldens, 1994):

$$\sum_k k^p g[k] = 0, \text{for } 0 \leqslant p < N. \tag{4–59}$$

This is equal to $G(z)$ having N zeros at $z = 1$ (thus $G(z)$ is clearly highpass). Let $\mathcal{P}_j$ in

$$\mathcal{P}_j f(x) = \sum_k < f, \tilde{\varphi}_{j,k} > \varphi_{j,k}(x), \tag{4–60}$$

denote the projection of f to the scaling space V_j. It can be shown that the asymptotic convergence of the approximation using biorthogonal wavelets is (Jawerth and Sweldens, 1994):

$$\|\mathcal{P}_j f(x) - f(x)\| = \mathcal{O}(2^{-j\tilde{N}}). \tag{4–61}$$

Thus, theoretically, the higher the $\tilde{N}$ (i.e., vanishing moments of the dual wavelet) the faster the detail part approaches 0 when considered in order from the coarsest to the finest level. However, higher $\tilde{N}$ means longer filters and thus the signal has to follow a polynomial for a longer period of time. Results in papers (Adams and Kossentini, 2000) and (Calderbank et al., 1998) suggest that increasing number of vanishing moments at the analysis side does not always increase the compression ratio.

The effects of the wavelet transform on the entropy estimate could be analyzed similarly as was done for linear predictors, however, this analysis is omitted from this thesis. Empirically it is easy to judge that the entropy of the AD, DA and DD parts are lower than that of the original signal, based on the pmf estimates in figures (4–11) and (4–12). Note also that the probability mass functions of the detail parts follow two sided geometric distributions. This is one reason to consider Golomb coding (see section (3.3)).

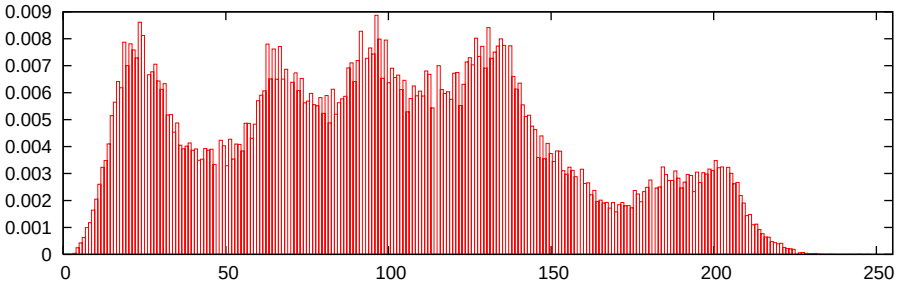

Figure 4–11. Estimate of the pmf of the original Lena image

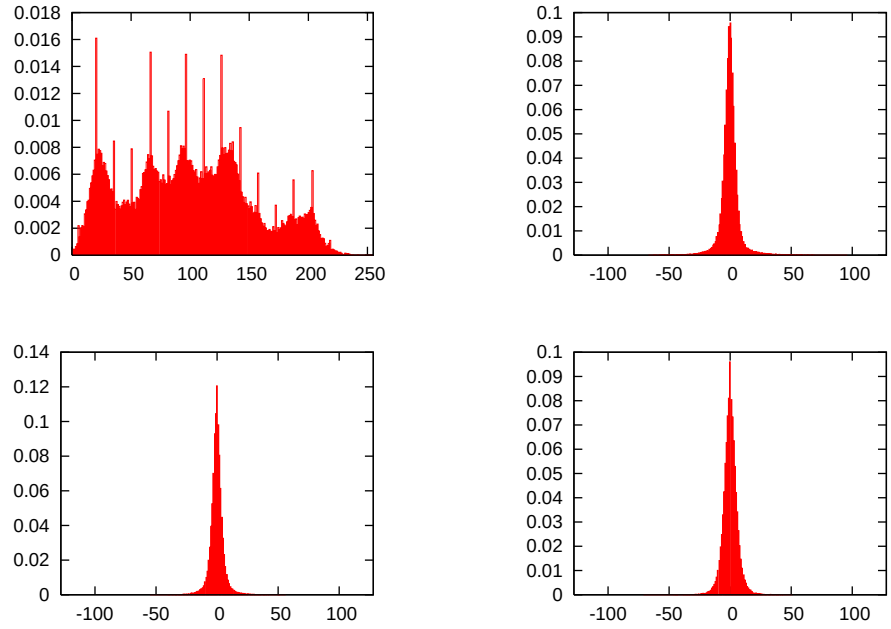

Figure 4–12. | AA | AD | / | DA | DD | Estimates of the pmfs of the approximation and detail parts at the finest level for Lena image.

4.4 Lossless implementation through lifting

Lifting was introduced in (Sweldens, 1996). Lifting scheme is a technique for implementing any wavelet transform efficiently. Lifting can also be used to derive new wavelet transforms based on the existing ones. In the context of this thesis, lifting is interesting mainly for two reasons: it allows for more computationally efficient implementation, and it is trivially easy to guarantee losslessness of the transform with lifting.

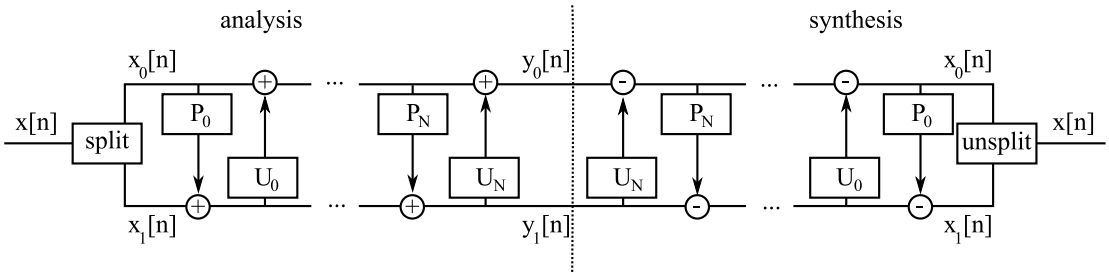

Figure 4–13. A general lifting scheme

In figure (4–13) a general lifting scheme is illustrated. In figure, the block labeled 'split' is used to divide signal $x[n]$ into two parts: $x_0[n]$ and $x_1[n]$. In general, splitting is completely arbitrary as long as there exists an inverse for it. The inverse of 'split' is labeled as 'unsplit' in figure (4–13). In the case of wavelet transform, 'split' and 'unsplit' together form the so called lazy wavelet. The 'split' is defined as

$$x_0[n] = x[2n], \tag{4–62}$$
$$x_1[n] = x[2n+1], \tag{4–63}$$

and the 'unsplit' as

$$x[n] = \begin{cases} x_0[n/2], & n \text{ is even}, \\ x_1[(n-1)/2], & n \text{ is odd}. \end{cases} \tag{4–64}$$

The signal $x_0[n]$ is called the even part and $x_1[n]$ the odd part. Blocks labeled as 'P_i' and 'U_i', for $i \in \{0, \ldots, N\}$, are arbitrary functions for which the range matches the set in which $x[n]$ is defined. In the context of this thesis, it is assumed that $x[n] \in \mathbb{Z}$ and thus $P_i \colon \mathbb{Z}^n \to \mathbb{Z}$ and $U_i \colon \mathbb{Z}^n \to \mathbb{Z}$. In other words, functions P_i and U_i operate on n samples of their input signal and they output a single sample. The structure in figure (4–13) guarantees losslessness. This is easy to see when considering a system with just two distinct blocks P_0 and U_0, where analysis is defined as

$$y_1[n] = x_1[n] + P_0\{x_0\}, \tag{4–65}$$
$$y_0[n] = x_0[n] + U_0\{y_1\}, \tag{4–66}$$

and the synthesis as

$$x_0[n] = y_0[n] - U_0\{y_1\}, \tag{4–67}$$
$$x_1[n] = y_1[n] - P_0\{x_0\}. \tag{4–68}$$

Summation is guaranteed to be lossless. Thus it is easy to verify by substition that the original $x_0[n]$ and $x_1[n]$ are recovered with arbitrary choice of P_0 and U_0. Losslessness of larger systems can be proven by induction.

In the case of wavelet transform P_i and U_i are filters and the step where P_i is applied to the signal is called the predict step while the step where U_i is applied is called the update step. Then $y_1[n]$ corresponds to the detail part and $y_0[n]$ to the

approximation part at the given level. Roughly it could be said that the predict step tries to predict the odd part based on the even part. Further levels of the wavelet decomposition can be evaluated by branching from the approximation part (at the dash line in figure (4–13)) just as was done with the traditional filterbank approach (see figure (4–8)).

If P_i and U_i are LTI filters and the lazy wavelet is used as a starting point, the lifting scheme guarantees that the resulting system is, at least, biorthogonal and perfect reconstruction. Furthermore, in (Daubechies and Sweldens, 1998) it is proven that any QMF filterbank can be implemented with a finite number of alternating predict and update steps. Thus from the wavelet design point of view, one can start with a known wavelet and modify it by adding new filters P_i and U_i to the scheme.

In the case of lossless compression the output of a filter has to be quantized in order to satisfy the requirement that the range of P_i or U_i has to be the set in which the signal is defined in. Thus the first predict step could be implemented as:

$$y_1[n] = x_1[n] - \left\lfloor \sum_{m=M_0}^{M_1} p_0[m] x_0[n-m] \right\rfloor, \tag{4–69}$$

and similarly the update step as:

$$y_0[n] = x_0[n] - \left\lfloor \sum_{l=L_0}^{L_1} u_0[l] y_1[n-l] \right\rfloor, \tag{4–70}$$

where $p_0[m]$ and $u_0[l]$ are arbitrary coefficient sequences.

From the implementation point of view lifting is interesting as it is easy to guarantee losslessness, as has been discussed. Moreover, lifting results almost always in reduced computational time and sometimes the required operations per sample can even be halved. Lifting is also a natural starting point for non-linear wavelet transforms. In fact, integer to integer mapping wavelet transform (as defined by equations (4–69) and (4–70)) is already nonlinear because of the rounding operation.

5 Design of the proposed lossless compression algorithm

Some requirements for the proposed lossless image compression algorithm have been discussed already in section (1.1). Perhaps the central philosophy can be distilled into a common phrase: keep it simple. The target is to reach sufficiently high compression ratio with low complexity coding and decoding, while satisfying the border requirements. These requirements are: lossless compression, fast spatial and spectral access and scalability in resolution. Often modern compression algorithms strive towards maximum compression ratio. Unfortunately it seems that with current technology the compression ratio cannot be improved much over what has already been achieved without the use of disproportional amount of computational resources. Therefore, it is natural to search for algorithms that provide sufficiently high compression ratio and minimize the use of computational resources. Here sufficiently high is around 3:1.

5.1 General structure of the algorithm

The proposed algorithm consists of two preprocessing stages and an entropy coding stage. All three stages are present in both compression and decompression as can be seen from figure (5–1). Detailed description of the individual stages and the reasoning behind choosing these particular techniques are given in sections (5.2), (5.3), and (5.4).

In figure (5–1) the letter 'A' denotes the original image with N bands. The individual bands of the image are wavelet transformed. The wavelet transformed data is denoted as 'B'. Linear prediction is performed between the bands of the image in the wavelet transform domain. Filter coefficients for the linear predictor are adapted for every wavelet transform level of the given band. Letter 'C' denotes the prediction error (i.e., residual). Prediction errors are entropy coded adaptively with Golomb codes. The final bit stream is denoted as 'D'. Details such as tiling and file format are discussed in the following sections.

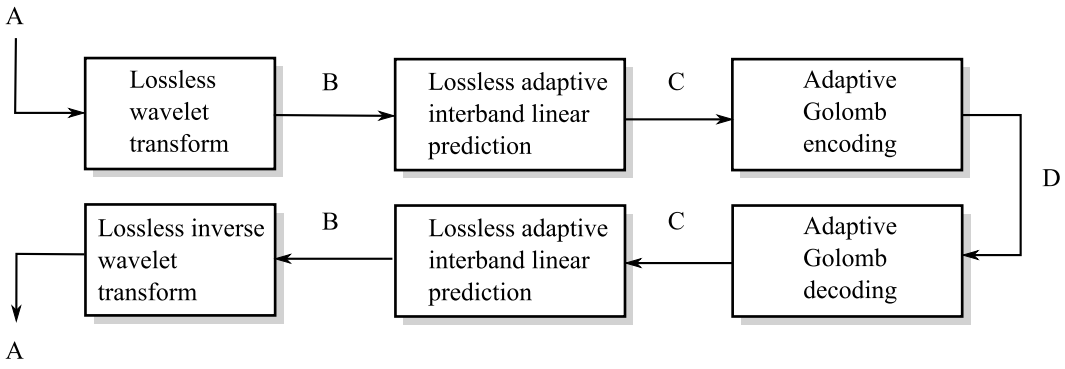

Figure 5–1. High level diagram of the compression and the decompression.

5.2 Intra-band redundancy

Hyper- and multispectral images consist of many bands (or channels). Every individual band can be handled as a single gray scale image. In this section pixel to pixel dependencies inside a single band are considered and how this redundancy can be exploited in compression. Considering intra-band redundancy and inter-band redundancy separately assumes certain level of independence between the two. This assumption probably results in slight underperforming in terms of compression ratios but, on the other hand, computational complexity is significantly reduced.

Often a linear prediction scheme is chosen because of its simplicity and good performance. Intra-band linear predictors have been used extensively, for example, in LOCO-I (Weinberger et al., 2000) and also in CALIC (Wu and Memon, 1996). Both are general lossless image compression algorithms (i.e. not designed specifically for aerial images). Also the new lossless compression algorithm, introduced in (Kubasova and Toivanen, 2004) and designed specifically for hyperspectral images, used linear prediction for intra-band decorrelation.

Unfortunately linear prediction does not easily allow for scalability in resolution – one of the key requirements for the proposed algorithm. Wavelet transform, on the other hand, is very well suited for scalability in resolution. Wavelet transform has been very successful in the area of lossy image compression where subjective image quality is important. Because of the increased flexibility offered by the wavelet transform it has been adopted also for lossless compression. Wavelet transform is used, for example, in JPEG2000 (Taubman and Marcellin, 2002) for both lossy and lossless compression. However, in (Taubman and Marcellin, 2002) it is noted that the achieved compression ratio is often somewhat inferior when compared to JPEG-LS (JPEG-Lossless). JPEG-LS uses LOCO-I as explained in (Weinberger et al., 1999).

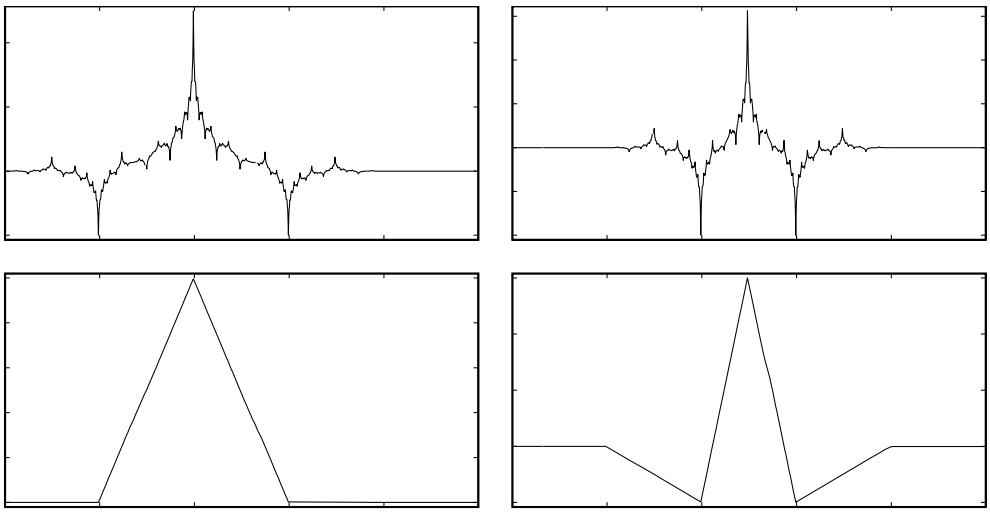

Figure 5–2. **a** **b** / **c** **d** a) Analysis scaling function b) Analysis wavelet function c) Synthesis scaling function d) Synthesis wavelet function

Wavelet transform was chosen for the proposed algorithm. The most important reason for this decision was the provided multiresolution representation. Because the compression is required to be lossless, a reversible wavelet transform that maps

integers to integers is required. In (Adams and Kossentini, 2000) several integer to integer mapping wavelet transforms were compared. Transforms were compared in the context of lossy and lossless compression. Computational complexity was also compared. One of the findings in the paper suggests that interpolating transforms (i.e., transforms requiring only two lifting steps) are the most promising for lossless compression. One such transform named 5/3 in the paper was found to have the lowest computational complexity of all the tested transforms while still performing extremely well (see table VI in (Adams and Kossentini, 2000)). Therefore, this transform was chosen for the proposed algorithm. Scaling and wavelet functions of the 5/3 transform are shown in figure (5–2).

The chosen transform belongs to biorthogonal family and is called biorthogonal 2,2 in (Daubechies, 1992). This name comes from the fact that the transform has 2 vanishing moments on the analysis side as well as on the synthesis side. The system functions for the analysis and synthesis scaling filters are indicated in equations (5–1) and (5–3) respectively (Daubechies, 1992, p. 277). The system functions of the wavelet filters as indicated in equations (5–2) and (5–4), are derived from the scaling filters by flipping signs.

$$\tilde{h}(z) = -\frac{1}{8}z^{-2}+\frac{1}{4}z^{-1}+\frac{3}{4}+\frac{1}{4}z-\frac{1}{8}z^{2} \quad (5\text{–}1)$$

$$\tilde{g}(z) = \frac{1}{4}z^{-1}-\frac{1}{2}+\frac{1}{4}z \quad (5\text{–}2)$$

$$h(z) = \frac{1}{4}z^{-1}+\frac{1}{2}+\frac{1}{4}z \quad (5\text{–}3)$$

$$g(z) = \frac{1}{8}z^{-2}+\frac{1}{4}z^{-1}-\frac{3}{4}+\frac{1}{4}z+\frac{1}{8}z^{2} \quad (5\text{–}4)$$

The transform has 5 coefficients in the scaling filter and 3 coefficients in the wavelet filter (at the analysis side), thus the name 5/3 in the (Adams and Kossentini, 2000). The corresponding filter coefficients should have a normalization factors but they are omitted in reversible transforms.

The lifting implementation of the 5/3 wavelet transform can be written (Adams and Kossentini, 2000):

$$d_{J-1}[n] = d_J[n]-\left\lfloor \frac{1}{2}(a_J[n+1]+a_J[n])+\frac{1}{2}\right\rfloor \quad (5\text{–}5)$$

$$a_{J-1}[n] = a_J[n]+\left\lfloor \frac{1}{4}(d_{J-1}[n]+d_{J-1}[n-1])+\frac{1}{2}\right\rfloor, \quad (5\text{–}6)$$

where equation (5–5) is the predict step in lifting terminology and equation (5–6) is the update step. It is the same transform used for lossless compression in JPEG2000 (Taubman and Marcellin, 2002). Note that even though equations (5–5) and (5–6) are indexed with one variable (i.e., n) $a_J[n]$ and $d_J[n]$ still represent 2D signals. Only one index is used because the transforms are applied separately for rows and columns. It is worth noting that the required operations per sample are halved when the transform is implemented via lifting (compare equations (5–3) and (5–4) to (5–5)

and (5–6)). Furthermore, no multipliers are needed in lifting implementation.

Problems may arise when part of the filter (scaling or wavelet) extend beyond the boundaries of the image. The most conservative way of handling the situation would consider the image undefined outside its borders. However, this would render part of the output also undefined. Symmetric extension is one technique to reduce the border effects caused by the finite size of the image. In symmetric extension the image is extended by mirroring it outwards from the borders. Another possibility is to use zero padding. However, zero padding, while being simple, causes strong artifacts to the data if the reconstruction (synthesis) is not lossless (e.g., when reconstructing lower resolution version of the image). Perhaps the best way would be to use second generation wavelets for which the filters are adapted for the boundaries as discussed in (Sweldens, 1997). Still, in the proposed algorithm symmetric extension is used as illustrated by figure (5–3) because of its simplicity and effectiveness.

In figure (5–3), the horizontal pass of the 2D separable wavelet transform is illustrated. One level of transform is composed of both horizontal and vertical pass. In figure the algorithm advances in clockwise order. The predict step applies the transform indicated in equation (5–5) to the odd and even parts, where a row of even part corresponds to $a_J[n]$ and a row of odd part to $d_J[n]$. Update step applies the transform indicated in equation (5–6) to even and detail part, where $d_{J-1}[n]$ corresponds to the detail part. In horizontal pass the parameter n in equations (5–5) and (5–6) refers to the column and in vertical pass the parameter refers to the row.

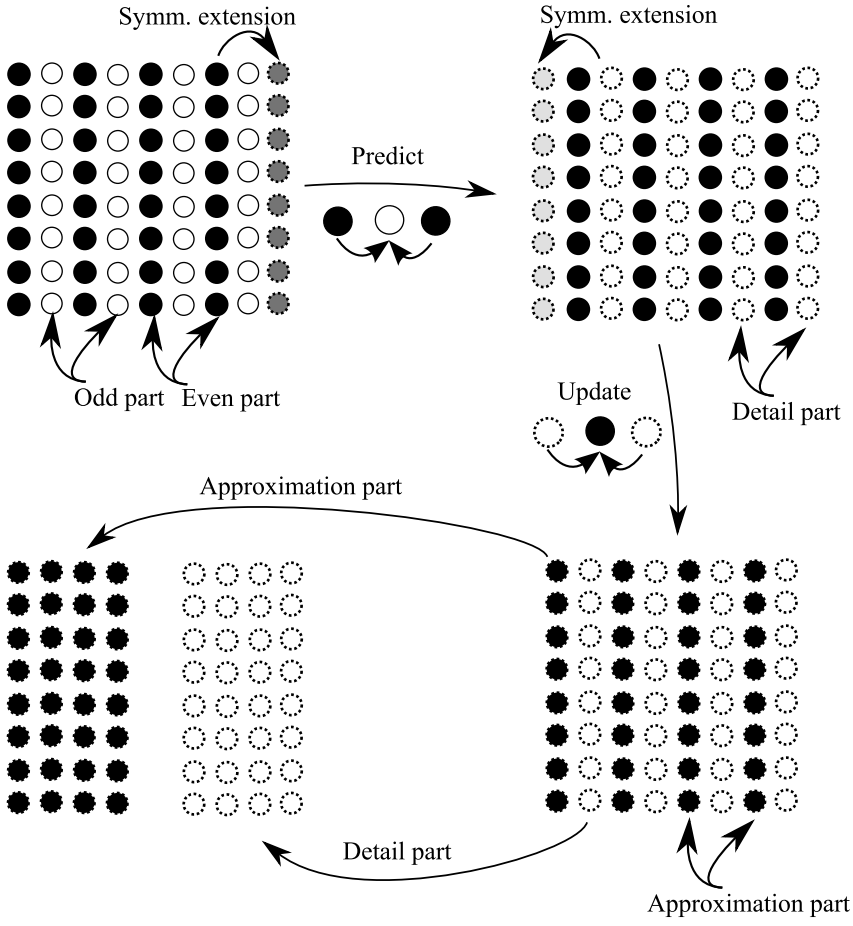

Figure 5–3. Horizontal pass of the 5/3 wavelet transform.

After detail and approximation parts (which are sometimes referred to as bands) are calculated, the algorithm proceeds by processing the obtained parts vertically (i.e., divides both parts further vertically into even and odd parts). At this point the first level of the wavelet transform is fully calculated and the algorithm starts over from the approximation part (i.e., the part which has been transformed by equation (5–6) both horizontally and vertically) in order to calculate the second level.

The 5/3 wavelet transform is not performed for the image as a whole. The image is tiled into tiles of size $L \times L$ pixels. Tiling is performed in image space (i.e., before applying the wavelet transform) in order to reduce computational complexity and memory requirements (as opposed to tiling in transform domain). Unfortunately, tiling in image space can cause some artifacts at the borders of the tiles in case of lossy reconstruction, however, symmetric extensions helps to reduce those artifacts. If the image size (width or height) is not multiple of L, tiles at the edge are truncated to fit the image. Tiling provides fast access to different parts of the image without the need to decompress the whole image (see section 5.5). After tiling every tile is transformed as described earlier. Every tile is treateated as an individual image while transforming them.

5.3 Inter-band redundancy

Hyperspectral images have significant correlation between different bands of the image. This correlation, or redundancy, can be exploited in many ways to achieve higher compression ratios. For example, in JPEG2000 it is possible to use lossless transforms to decorrelate bands (Taubman and Marcellin, 2002). In (Kubasova and Toivanen, 2004) an algorithm was presented to find a better ordering for the bands of a hyperspectral image. Better here means that two adjacent bands would have, roughly speaking, higher correlation in the new ordering. After reordering linear prediction is performed between the bands. The method was found to perform very well. Also wavelet transform has been used in inter-band decorrelation, for example, in ICER-3D (Kiely et al., 2006). However, the lossless performance is not found to be as good as in (Kubasova and Toivanen, 2004).

In the proposed algorithm *adaptive* linear prediction is applied between adjacent bands of the image. Performing inter-band decorrelation after intra-band decorrelation has a few advantages. Firstly and the most importantly, the wavelet transform in its quantized implementation is only approximately linear, and decoding a single band at certain coarse resolution would require decompression of all wavelet decomposition levels if decorrelations are not applied in this order. Secondly, as the prediction is done in the transform domain, it is possible to choose filter coefficients for every decomposition level independently. This provides additional flexibility and possibly higher compression ratio.

Linear prediction is implemented with FIR filters and thus the inverse filters are IIR. IIR filters form a dependence between the current band and all its causal neighbors in decoding. Here causal neighbor refers to band already available at the decoder.

Because of dependence, bands have to be decompressed in the same order as they have been compressed. In order to minimize the impact of this dependence, bands are divided into band packs. For an image with N bands, $\lceil N/K \rceil$ band packs are generated, where K is the number of bands in a single band pack. Linear prediction is applied only inside a band pack thus eliminating the dependence on the bands in other band packs. This way a single band pack can be decoded independent of other band packs.

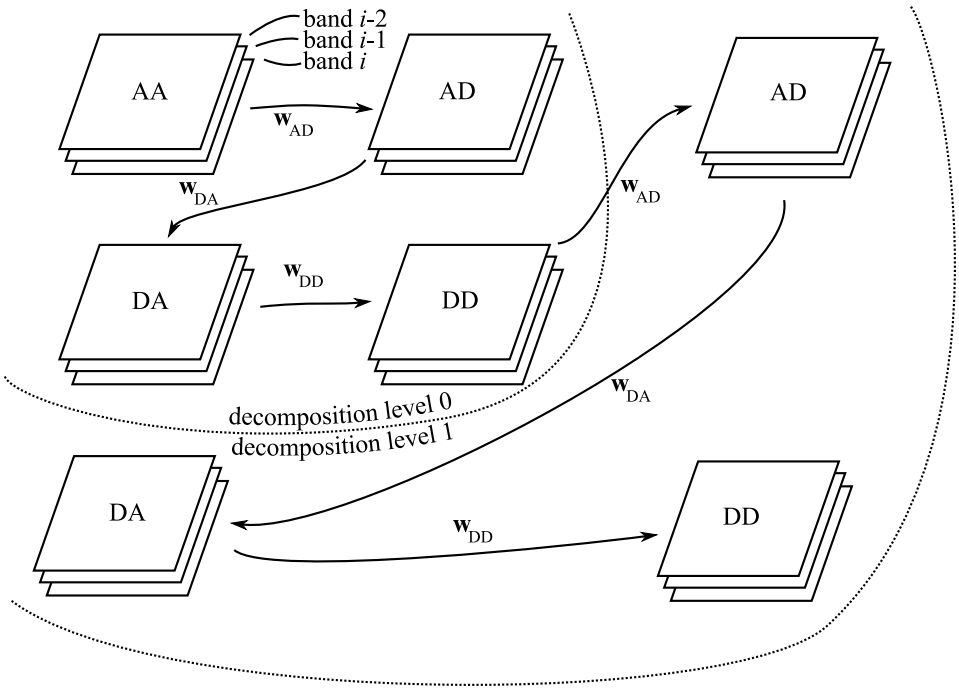

Figure 5–4. Adaptation of the predictor filter coefficients. The tail of the arrow points to part which is used in adaptation and the head of the arrow points to part which is predicted using the adapted coefficients.

Predictor coefficients are adapted for every band and for every decomposition level based on the information currently available at the decoder as shown in figure (5–4). This eliminates the need to store the coefficients, which would cause overhead, as the coefficients can be deduced in the decoder without extra information. In figure (5–4), band i is predicted from bands i-2 and i-1. In order to minimize the computational complexity, the predictor is limited to two taps. One tap predictor is used in the leading edge of the band pack (i.e., when predicting the second band of a band pack from the first band of the band pack) and the first band of the band pack is left as is.

For the approximation part of decomposition level 0 the following fixed polynomial predictors

$$e_{\mathrm{AA}}[n,m,1] = x_{\mathrm{AA}}[n,m,1] - x_{\mathrm{AA}}[n,m,0], \quad \forall n,m \tag{5–7}$$

$$e_{\mathrm{AA}}[n,m,i] = x_{\mathrm{AA}}[n,m,i] - 2x_{\mathrm{AA}}[n,m,i-1] + x_{\mathrm{AA}}[n,m,i-2], \quad \forall n,m;\, i>2 \tag{5–8}$$

are used. $x_{\mathrm{AA}}[m,n,p]$ denotes the wavelet transformed data at decomposition level 0 for part AA and pth band of a band pack. Filter coefficient vectors, $\boldsymbol{w}_{\mathrm{AD}}$, $\boldsymbol{w}_{\mathrm{DA}}$

and $\boldsymbol{w}_{\mathrm{DD}}$, are adapted for all other parts, AD, DA and DD respectively (at all decomposition levels). Adaptation is based on the assumption that the MSE optimal filter coefficients are relatively similar between different decomposition levels. This assumption seems to hold well in practice as illustrated by the coefficients in table (5–1).

Table 5–1. Example of MSE optimal predictor coefficients, for a band in one particular hyperspectral image.

Part	Level 0	Level 1	Level 2	Level 3
$\boldsymbol{w}_{\mathrm{AA}}$	(2.2770 -1.3250)	–	–	–
$\boldsymbol{w}_{\mathrm{AD}}$	(2.5358 -1.6271)	(2.5724 -1.6687)	(2.4664 -1.5531)	(2.0439 -1.0733)
$\boldsymbol{w}_{\mathrm{DA}}$	(2.5355 -1.6268)	(2.5664 -1.6633)	(2.4623 -1.5467)	(2.0985 -1.1344)
$\boldsymbol{w}_{\mathrm{DD}}$	(2.5572 -1.6527)	(2.5551 -1.6503)	(2.3404 -1.4094)	(1.6552 -0.6316)

In the decomposition level 0 coefficients are calculated from the approximation part AA as follows:

First the correlation matrix is calculated as

$$\boldsymbol{R}_{\mathrm{AA}}[i;0] = \boldsymbol{X}_{\mathrm{AA}}[i;0]^T \boldsymbol{X}_{\mathrm{AA}}[i;0], \tag{5–9}$$

where $\boldsymbol{X}_{\mathrm{AA}}[i;0]$ is a $M \times 2$ matrix with columns composed of all the samples of the approximation parts of the respective bands, $i-1$ and $i-2$. In the case of one tap predictor $\boldsymbol{X}_{\mathrm{AA}}[i;0]$ reduces to a vector and $\boldsymbol{R}_{\mathrm{AA}}[i;0]$ is just the square sum. Next

$$\boldsymbol{r}_{\mathrm{AA}}[i;0] = \boldsymbol{X}_{\mathrm{AA}}[i;0]^T \boldsymbol{x}_{\mathrm{AA}}[i;0], \tag{5–10}$$

is calculated where $\boldsymbol{x}_{\mathrm{AA}}[i;\,0]$ is a column vector containing the samples from the approximation part (AA) of the band i. From these the filter coefficients for the detail part AD of decomposition level 0 are calculated as

$$\boldsymbol{w}_{\mathrm{AD}}[i;0] = \boldsymbol{R}_{\mathrm{AA}}[i;0]^{-1} \boldsymbol{r}_{\mathrm{AA}}[i;0]. \tag{5–11}$$

Coefficients in equation (5–11) are MSE sense optimal for predicting the next band in approximation part (i.e., AA). They are not generally optimal for predicting any other parts. However, they serve as an estimate for the optimal coefficients. Decomposition level 0 is a special case. Generally the coefficients are calculated as

$$\begin{aligned}
\boldsymbol{w}_{\mathrm{AD}}[i;y] &= \left(\boldsymbol{X}_{\mathrm{DD}}[i;y-1]^T \boldsymbol{X}_{\mathrm{DD}}[i;y-1]^T\right)^{-1} \boldsymbol{X}_{\mathrm{DD}}[i;y-1]^T \boldsymbol{x}_{\mathrm{DD}}[i;y-1], \\
\boldsymbol{w}_{\mathrm{DA}}[i;y] &= \left(\boldsymbol{X}_{\mathrm{AD}}[i;y]^T \boldsymbol{X}_{\mathrm{AD}}[i;y]^T\right)^{-1} \boldsymbol{X}_{\mathrm{AD}}[i;y]^T \boldsymbol{x}_{\mathrm{AD}}[i;y], \\
\boldsymbol{w}_{\mathrm{DD}}[i;y] &= \left(\boldsymbol{X}_{\mathrm{DA}}[i;y]^T \boldsymbol{X}_{\mathrm{DA}}[i;y]^T\right)^{-1} \boldsymbol{X}_{\mathrm{DA}}[i;y]^T \boldsymbol{x}_{\mathrm{DA}}[i;y],
\end{aligned}$$

where y refers to the decomposition level (see figure 5–4). The prediction is per-

formed as

$$e_P[n,m,i;y] = x_P[n,m,i;y] - \left\lfloor \boldsymbol{w}_P^T[i;y] \begin{pmatrix} x_P[n,m,i-1;y] \\ x_P[n,m,i-2;y] \end{pmatrix} \right\rfloor, \tag{5–12}$$

where indices n and m refer to the location of the sample, i to the band, y to the decomposition level and P is one of AD, DA or DD. In the case of the one tap predictor, vectors obviously reduce to scalars.

Care must be taken when the coefficients are calculated. The adaptation must be bit by bit accurate on all platforms. Floating-point calculations may have platform dependent differences that may render the algorithm lossy. Implementing floating-point calculations in software guarantees the bit accuracy, but performance may be unacceptable. Good solution would be to implement all calculations as fixed-point operations with guaranteed rounding policies. However, the implementation of the proposed algorithm in this thesis uses hardware floating-point calculations because testing takes place only on a certain platform.

5.4 Entropy coding

The actual compression happens at the entropy coder. Entropy coder takes advantage of the statistics of the residuals in order to find more compact representation for the data. Coding was discussed in more detail in section (3.3).

In JPEG2000, context dependent binary arithmetic coder is used (Christopoulos et al., 2000). This means that the arithmetic coder operates on two symbols and the used statistics depend on finite number of contexts. In JPEG-LS it is possible to use modified Golomb coding, which has an advantage in lighter computational requirements. Golomb codes are used also in the proposed algorithm.

Linear prediction residuals are adaptively entropy coded with Golomb codes. First prediction residuals are one-to-one mapped to non-negative integers:

$$r[n,m] = \begin{cases} 2e[n,m], & e[n,m] \geqslant 0 \\ -2e[n,m]-1, & e[n,m] < 0 \end{cases} \tag{5–13}$$

where $e[n,\ m]$ are the prediction residuals. Band, part, and decomposition level identifiers are omitted for convenience.

Adaptation is necessary as the mean of the mapped residuals may not be stationary. For example, the variance of the residuals is probably lower when they correspond to smooth areas in the image (such as water) and higher when they correspond to rougher areas. Because of the mapping, non-stationary variance results in practice as non-stationary mean. In section (3.3) the estimate for the optimal parameter b_{opt} of the Golomb-Rice codes was found to be

$$\hat{b_{\text{opt}}} = \log_2(\ln 2) + \log_2 \hat{\mu}\,, \tag{5–14}$$

where $\hat{\mu}$ is the average of the data. This definition lends itself easily to adaptation.

Adaptation is performed by replacing the average $\hat{\mu}$ by moving average. Moving average could be calculated by averaging N previous samples (this is called boxcar filtering). It is possible to implement boxcar computationally efficiently, however memory requirements depend on the filter length. Therefore, an IIR filter

$$H(z) = \frac{k}{1 + (k-1)\, z^{-1}}, \tag{5–15}$$

where $k \in (0, 1]$, is more suitable as the memory and computational requirements are constant. Parameter k affects the adaptation speed. In figure (5–5) one sees the results of applying $H(z)$ to uniform noise with non-stationary mean and variance. It is clearly visible that larger variance results in larger error in the estimate. Moreover, the estimate of the mean is slightly delayed when compared to the actual mean. The delay reaches its maximum at frequency 0, where it is $\frac{1}{k} - 1$ samples. Thus in the case of the example in figure (5–5) the estimated mean has a maximum delay of 9 samples when compared to the actual mean. Delay is not a large problem when the variance changes slowly compared to the delay (i.e., when variance is approximately constant under d sample window, where d is the delay) as the impact on the accurace of the estimate will be small.

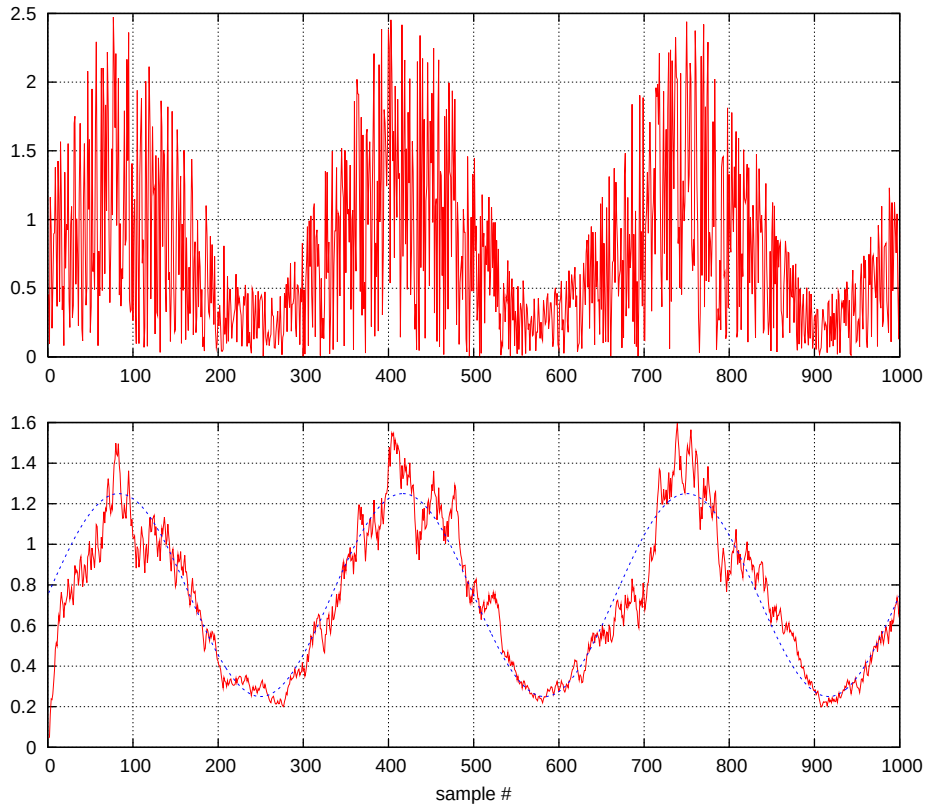

Figure 5–5. **a** / **b** a) Uniform noise with non-stationary mean and variance b) Blue dashed line is the actual mean and red line is the estimated mean (i.e., moving average). $k = 1/10$

This "problem" (i.e., delay of the estimate) is common to all moving average estimators in the case of lossless coding. This is because, in lossless coding one has to use only causal filters. Delays could be completely eliminated with non-causal filters. In the decoding end only some of the samples are present and the order in which the samples arrive at the decoder defines whether a given filter is causal or not.

In two dimensions the moving average is calculated separately as

$$z[n,m] = z[n,m-1] + \lfloor k(d[n,m] - z[n,m-1]) \rfloor, \tag{5–16}$$

which performs the filtering first in the direction of columns, where $z[n, m]$ is the output of the 1D moving average along the direction of columns, and

$$\hat{\mu}[n,m] = \hat{\mu}[n-1,m] + \lfloor k(z[n,m] - \hat{\mu}[n-1,m]) \rfloor, \tag{5–17}$$

which performs the filtering in the direction of rows. Thus, only the previous column and the previous sample have to be kept in memory. In equation (5–16) $d[n, m]$ are the mapped residual values and in equation (5–17) $\hat{\mu}[n, m]$ is the output of the moving average. Now the estimate of the optimal non-stationary parameter can be defined as

$$\hat{b_{\text{opt}}}[n,m] = \log_2(\ln 2) + \log_2 \hat{\mu}[n,m]. \tag{5–18}$$

There is one problem associated with the estimate. When $\hat{\mu}$ is close to 0, $\hat{b_{\text{opt}}}$ will be negative. Negative values could be mapped to 0 as a special case. Moreover, b can be calculated in its rounded form b_r as:

$$b_r = \lfloor \log_2(\hat{\mu}[n,m] + 1) \rfloor, \tag{5–19}$$

which works equally well and there is no need for special cases. Note that as the value of $\log_2(\ln 2) \approx -0.53$, leaving it out of equation (5–19) actually results in rounding $\hat{b_{\text{opt}}}$ towards nearest integer with slight bias towards positive infinity.

5.5 File format

The file format describes how the Golomb coded data is stored in a file. The layout is divided into three logical units: *file*, *tile* and *band pack* (see figure (5–6)). Every logical unit contains a header and other logical units. *File* contains all the necessary information to decompress the original image losslessly. File is further divided into tiles. *Tile* contains the information needed to decompress a particular spatial portion of the image. Division into tiles is done before applying the wavelet transform as

described in section (5.2). Tiles are stored in row order as shown in figure (5–7) for the case of 3×3 tiling.

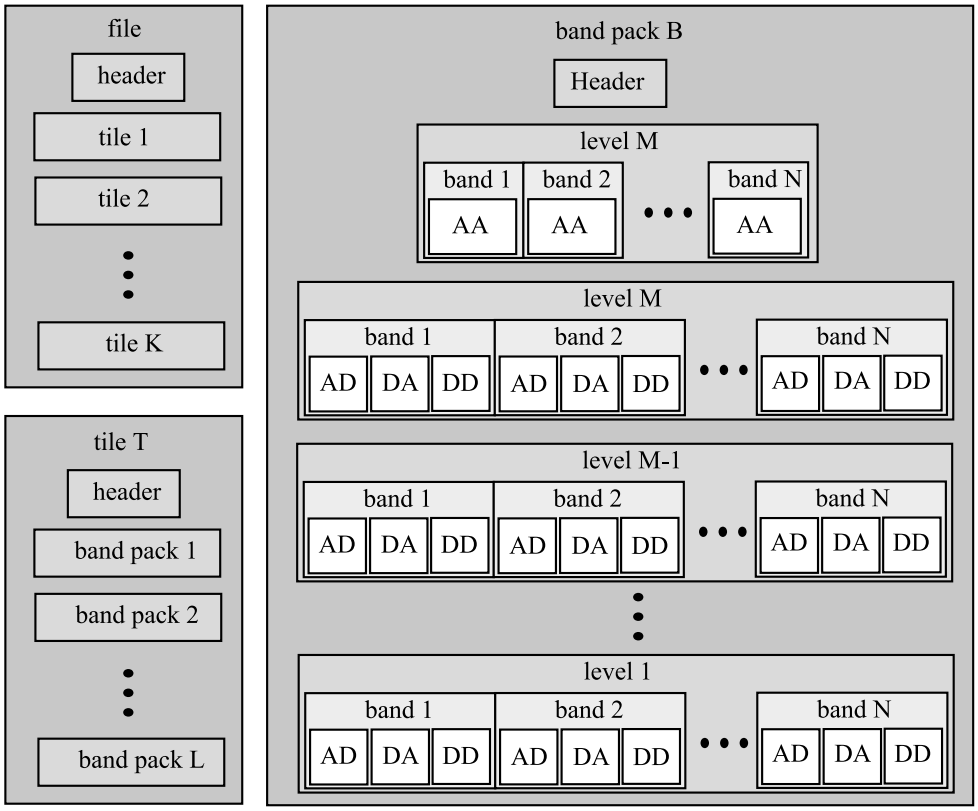

Figure 5–6. File format

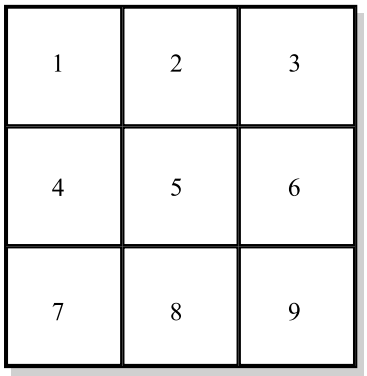

Figure 5–7. Example of tile ordering.

Every tile is divided into band packs. *Band packs* are constructed as described in section (5.3). Band pack contains the information needed to losslessly decode certain number of bands of a certain tile.

Every logical unit has a header associated with it. The specific structure of the

headers can be seen in tables (5–2), (5–3) and (5–4). All data is in unsigned little-endian format unless otherwise mentioned in the comment column.

Table 5–2. File header

Description	Bits	Comments
Width	32	
Height	32	
Bands	16	
Bands in band pack	8	$0 = 1$ band,..., $255 = 256$ bands
Tile size	16	In pixels
Decomposition levels	8	
Initial Golomb parameter b	8	
Adaptation parameter k	16	Divide by 2^{16}
Size of tile 1	32	In bytes
...	32	
Size of tile $K-1$	32	K is the number of tiles in total

Table 5–3. Tile header

Description	Bits	Comments
Size of band pack 1	32	In bytes
...	32	
Size of band pack $L-1$	32	L is the number of band packs in total

Table 5–4. Band pack header

Description	Bits	Comments
Size of decomposition level M	32	In bytes (only part AA)
Size of decomposition level M	32	parts AD, DA, and DD
Size of decomposition level $M-1$	32	
...	32	
Size of decomposition level 2	32	

Band pack differs from other units as it contains several blocks which do not have headers. Although the structure of the band pack may seem unintuitive at first, it maximizes the size of continuous blocks read from storage media or transfered over network and minimize the need to read unnecessary information.

The first band pack is divided into wavelet decomposition levels. They are in the order from the coarsest level to the finest level. Of those blocks, the first one is special as its level number is actually the same level number as that of the second block. The first level block contains only the approximation parts. This way, if only very coarse approximation of the image is required, it can be reconstructed by reading only the first block. From there on, level blocks contain detail parts: AD, DA and DD, for several bands. The image can be subsequently refined by reading and decompressing more level blocks. After all level blocks are read, lossless result is achieved.

Every level block is divided into band blocks. Band blocks correspond to individual bands of the image. However, bands inside a band pack cannot be individually decompressed as band N depends on previous bands $N-1$ and $N-2$. Band blocks contain an approximation parts of the wavelet transform (first block) or 3 detail parts of the wavelet transform. These are stored as the Golomb codes for the residuals.

Example

Consider an image of size 1228×512 pixels with 224 bands. Let the tile size be 256×256. Thus the image will be divided into 5×2 tiles. Two tiles at the left edge of the image will be truncated to size 204×256 in order to fit the image. Let the number of bands in a band pack be 12. Thus there will be 19 band packs of which the last one will have only 8 bands. And lastly, let the number of decomposition levels be 5.

Now consider that one wants to access pixel (1000,300) at band 45. Decompression would start by reading the file header. Firstly the size of the image is determined from the first two fields. Secondly, the tile size is determined and, based on the location of the pixel, the right tile number is calculated. In this case the pixel (1000,300) belongs to tile 10. Based on 3rd and 4th field (band counts) of the file header it is easy to determine that band 45 is found in band pack number 4.

Tile 10 is the last tile so the location of the tile in the file can be determined by summing the sizes of the first 9 tiles and interpreting the result as an offset from the end of the file header. Resulting index points to tile header of the 10th tile. Similarly, band pack number 4 can be found by summing the sizes of the 3 first band packs and treating it as an offset from the end of the tile header.

Band 45 is the 9th band of the band pack. Thus if only coarse approximation is required, it is sufficient to read 9 first bands of the approximation part located right after the band pack header. If more accurate approximation is needed the band pack header can be used to calculate the starting points of the other levels (similarly as was done with tiles and band packs). Note that it is not possible to jump directly to certain band as the given band depends on the preceding bands inside a band pack. However, it is not necessary to perform the inverse wavelet transform for the bands that are of no interest.

6 Comparison of compression algorithms

In this section the parameters of the proposed algorithm are discussed. The results of empirical tests of the parameter values are presented. Furthermore, the proposed algorithm is compared to other algorithms, such as, JPEG2000 and JPEG-LS (JPEG-Lossless), both feature wise and compression ratio wise.

All the reported compression ratios are calculated by dividing the original file size by the size of the compressed file. Exception to this are cited results, which are mentioned separately.

Three publicly available and free hyperspectral images from (NASA AVIRIS website, 2006) were used in the tests. These images have 224 bands which span the EM spectrum from 400 nm to 2500 nm. The data is signed and has bitdepth of 16 bits. All the three images are radiance data (as opposed to reflectance data). The images are distributed as scenes (i.e., smaller regions of the larger image). Prior the compression these scenes were combined into the three images described in table (6–1).

Table 6–1. Three hyperspectral images used for the tests

Name	Width	Height	Bands	File size [kb]	Notes
Jasper Ridge	614	2586	224	694664	vegetation
Cuprite	614	2206	224	592587	geological features
Low altitude	614	3689	224	990958	high spatial resolution

6.1 Implementation and parameters of the proposed algorithm

The proposed algorithm was implemented in Matlab^{TM} R2006a environment with the entropy coder written in C as a MEX-file. Most of the operations can be considered as integer to integer mappings. Notable exception is the adaptive predictor for which the coefficients are calculated using floating point operations.

In table (5–2) some parameters of the proposed algorithm were listed. The most

important ones are: the adaptation speed (or adaptation parameter k), the tile size, the number of wavelet decomposition levels and the number of bands in a band pack. These parameters have potentially significant effect on the compression ratio and especially the number of bands in the band pack affects the speed of the decompression. The following parameter values were used in the tests unless mentioned otherwise:

$$\begin{aligned} \text{initial Golomb parameter } b &= 6, \\ \text{adaptation parameter } k &= 0.06, \\ \text{number of decomposition levels} &= 5, \\ \text{number of bands in a band pack} &= 16, \\ \text{tile size} &= 256 \times 256. \end{aligned}$$

The optimal adaptation speed depends on the properties of the image. For most hyperspectral images the optimal value of the adaptation parameter seems to be between 0.04 and 0.11. This can be seen also in figure (6–1). The plot in the figure is graphed as continuous because there is a reason to assume that compression ratio is a continuous and relatively smooth function of the adaptation speed.

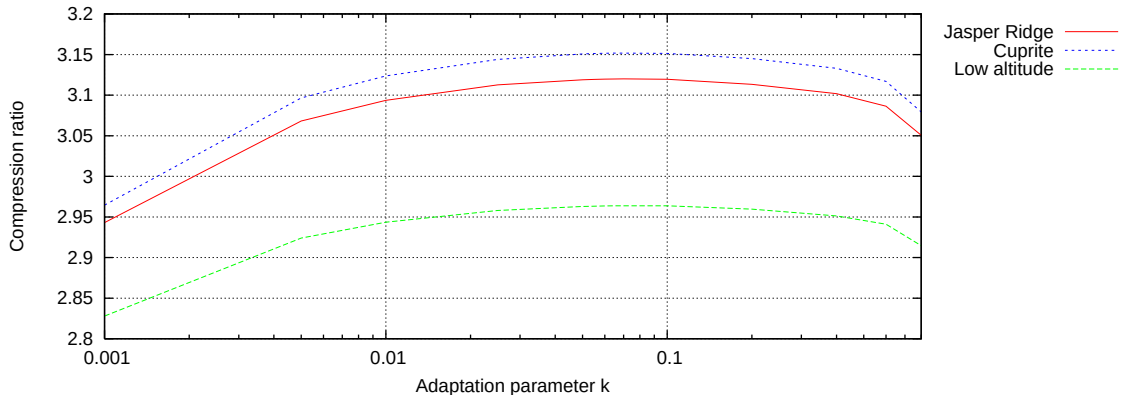

Figure 6–1. Effect of the adaptation speed to compression ratio

It can be seen from figure (6–2) that there is little or no advantage of using more than four decomposition levels with these particular test images. However, when comparing the results for 'Jasper Ridge' and 'Cuprite' it is clear that the effect of the number of decomposition levels depends on the image. Thus it is possible that for some images it is useful to decompose the image further than to four levels. The effect on the speed of decompression is not significant. In fact, if the decomposition is continued ad infinitum the total number of required operations is at most twice of that required for computing the first level.

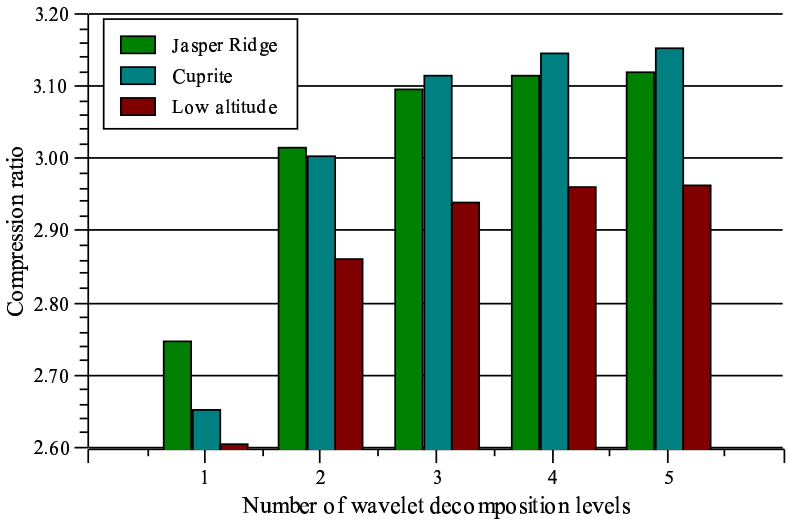

Figure 6–2. Effect of the decomposition depth to compression ratio

Tile size effects mostly the flexibility to fetch random spatial regions of the image. Smaller tiles allow to fetch more accurately only the regions of interest. However, smaller tile size hinders the performance of the wavelet transform and increases the amount of tiles per pixel. Increased number of tiles results in increased number of tile and band pack headers. That is, overhead per pixel increases. Thus, it is beneficial to keep the tile size as high as is convenient from the application point of view. Based on the bar chart in figure (6–3), 256×256 pixels seems to be a good tile size in the sense that the compression ratio does not substantially increase by using larger tile size.

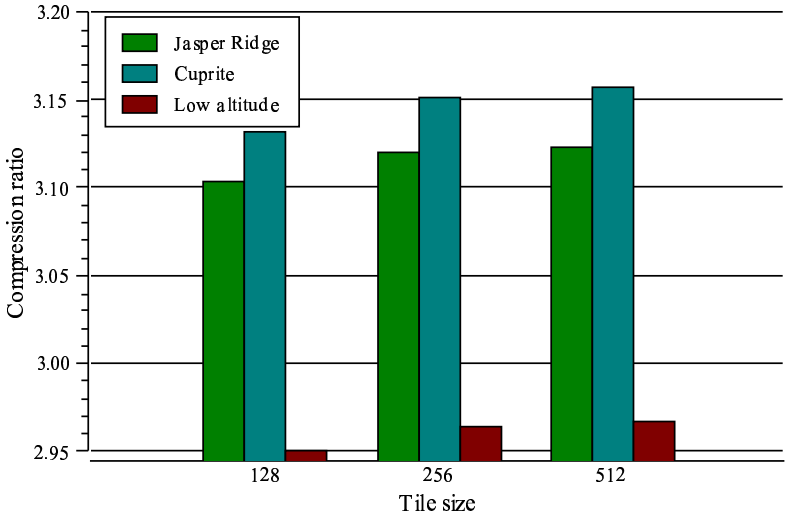

Figure 6–3. Effect of the tile size to compression ratio

Number of bands in a band pack has a significant effect on loading times. If there are N bands in a band pack the expected number of bands required to decode a

single band is $N/2$. When the compression ratio is approximately 3:1, 6 bands per band pack would result, on the average, in the same number of bytes loaded from the disk, or over the network, than would be required to load a single band of raw uncompressed data. However, using more than 6 bands per band pack increases the compression ratio (see figure (6–4)). The average number of bytes required to load a single band can be reduced by using smart caching techniques.

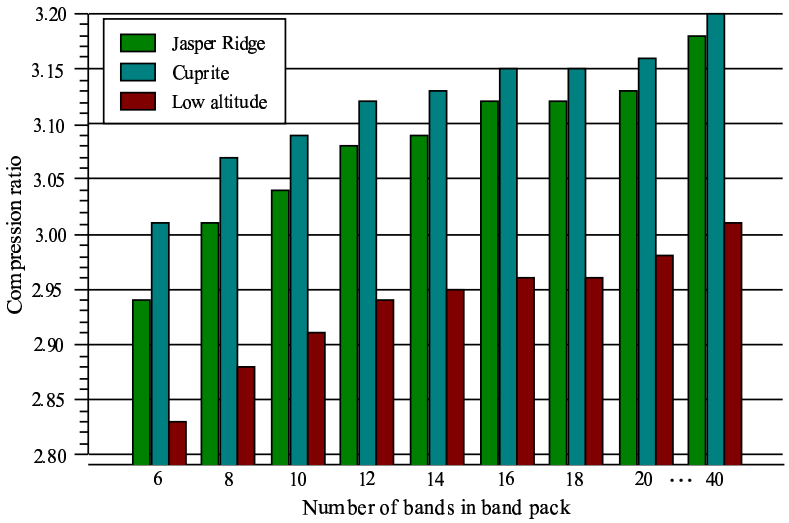

Figure 6–4. Effect of the band pack size on the compression ratio.

6.2 Comparison with other algorithms

JPEG2000

JPEG2000 is a standard for lossy and lossless image compression (Taubman and Marcellin, 2002). It provides highly scalable data stream with progressive lossy to lossless decompression within a single data stream. JPEG2000 is both resolution and distortion scalable. The former is a property which is supported also by the proposed algorithm. That is, losslessly compressed image can be decompressed into lossy lower resolution version of the image. JPEG2000 lossless compression employs the same integer to integer mapping wavelet transform as the proposed algorithm, namely 5/3 transform.

Distortion scalability is a property not supported by the proposed algorithm. JPEG2000 is able to decompress full resolution version of the image with lower signal-to-noise ratio. This obviously results in a lower quality version of the image, but, on the other hand, only part of the data has to be transfer and decompressed. Distortion scalability is achieved by an algorithm based on EBCOT or Embedded Block Coding with Optimized Truncation (Christopoulos et al., 2000).

Some implementations of JPEG2000 are able to decorrelate multicomponent (i.e., multiband) images. One such implementation is available from (Kakadu JPEG2000 implementation website, 2006). This implementation

was used in the following tests. Further information about JPEG2000 standard is available from (Kakadu JPEG2000 implementation website, 2006), (Taubman and Marcellin, 2002) and (Christopoulos et al., 2000).

JPEG-LS

JPEG-LS is based on LOCO-I (LOw COmplexity LOssless COmpression for Images). LOCO uses context based intra-band prediction scheme and modified Golomb coding for the residuals (Weinberger et al., 2000). It does not provide good support for inter-band decorrelation of hyperspectral images. LOCO-I does not allow for scalability and it is not particularly flexible when compared to JPEG2000. For more information about JPEG-LS see (Weinberger et al., 2000).

Fast Lossless

In (Klimesh, 2005) a novel adaptive predictive technique for compression of hyperspectral imagery was introduced. The adaptive prediction is a variant of the LMS algorithm. The prediction neighborhood contains 3 pixels from the 3 preceding bands and 3 pixels from the current band. The prediction is performed independently for fixed size regions of the image. The prediction residuals are coded with Golomb-Rice codes similarly as is done in the proposed algorithm.

Fast Lossless algorithm is similar to the proposed algorithm. However, the proposed algorithm implements the adaptivity in the transform domain and does not employ the LMS algorithm. Furthermore the proposed algorithm uses only two preceding bands and thus the Fast Lossless algorithm has a slight advantage. It should be noted, however, that the Fast Lossless algorithm does not provide scalability in resolution while the proposed algorithm does. Due to fixed size regions (or tiles) the Fast Lossless algorithm does provide fast random access spatially.

ICER-3D

In (Kiely et al., 2006) ICER-3D was introduced. ICER-3D is an extension to ICER which was used onboard of the Mars Exploration Rovers. ICER-3D, as ICER and JPEG2000, compresses progressively allowing for scalability in distortion. Being a wavelet transform based compressor it provides also scalability in resolution, similarly as the proposed algorithm.

Because of the special requirements of interplanetary data transmission ICER-3D uses error containment segments in order to minimize the effects of data loss. This accounts as slight overhead in compression ratio. In contrast, no error resiliency is guaranteed by the proposed algorithm.

Comparison with JPEG2000, JPEG-LS, ICER-3D and Fast Lossless

The tests of the proposed algorithm were performed using the three hyperspectral images described in table (6–1). Results for ICER-3D, Fast Lossless and JPEG-LS are adopted from table 7 of (Kiely et al., 2006). It is not known whether the results in (Kiely et al., 2006) are achieved for the combined hyperspectral images or for every scene independently. However, difference in compression ratio should be minimal between the two cases. Compression ratios for ICER-3D, Fast Lossless and JPEG-LS are calculated from the given average bitdepth by dividing 16 with it.

Table 6–2. Comparison of the proposed algorithm with ICER-3D, Fast Lossless and JPEG-LS. Algorithm 'A' denotes the proposed algorithm with 16 bands in a band pack and 'B' is the proposed algorithm with 40 bands in a band pack. The highest compression ratios per image are denoted as bold face numbers.

Algorithm\Image	A	B	ICER-3D	Fast Lossless	JPEG-LS
Jasper Ridge	3.12	**3.18**	2.95	3.17	2.06
Cuprite	3.15	3.20	3.07	**3.23**	2.21
Low altitude	2.96	**3.01**	2.84	3.00	2.09

The proposed algorithm performs well compared to the other three algorithms. With the parameter set 'B' it has the best compression ratio of the all tested algorithm for two images, 'Jasper Ridge' and 'Low altitude'. For the image 'Cuprite', the Fast Lossless algorithm performs slightly better than the proposed algorithm. However, even with the 'A' parameter set, the proposed algorithm performs consistently better than ICER-3D and JPEG-LS. It should be mentioned that algorithm described in (Kubasova and Toivanen, 2004) performs even better than Fast Lossless on 'Jasper Ridge image'. This algorithm was not included in comparison as the results were available only for 'Jasper Ridge' and the implementation was not readily available.

In the following tests only the first scenes of the previously used images are compressed. This choice was made because of the technical difficulties with large images and the chosen JPEG2000 implementation. In table (6–3) the dimensions of the images and some information about the contents are listed.

Table 6–3. First scenes of the three hyperspectral images used for the tests

Name	Width	Height	Bands	File size [kb]	Notes
Jasper Ridge sc01	614	512	224	137536	vegetation
Cuprite sc01	614	512	224	137536	geological features
Low altitude sc01	614	512	224	137536	high spatial resolution

Before compressing with JPEG2000, the big-endian band interleaved by pixel AVIRIS data was transformed into little-endian band sequential data. Following option string was used for compressing the first scenes of the three images:

```
-i jasperridge.raw*224@628736 -o jasperridge.jpx Creversible=yes Sdims={512,614}
Sprecision=16 Ssigned=yes Clayers=16 Mcomponents=224 Msigned=yes Mprecision=16
```

```
Mvector_size:I4=224 Mvector_coeffs:I4=2048 Mstage_inputs:I25={0,223}
Mstage_outputs:I25={0,223} Mstage_collections:I25={224,224}
Mstage_xforms:I25={DWT,1,4,3,0} Mnum_stages=1 Mstages=25
```

Options were based on the usage example Ai from the demonstration distribution for Win32 (see (Kakadu JPEG2000 implementation website, 2006)). The interband (or multicomponent) decorrelation was performed by three level wavelet transform.

Table 6–4. Comparison of the proposed algorithm and JPEG2000. Algorithm 'A' denotes the proposed algorithm with 16 bands in a band pack and 'B' is the proposed algorithm with 40 bands in a band pack. The highest compression ratios per image are denoted as bold face numbers.

Algorithm\Image	Jasper Ridge sc01	Cuprite sc01	Low altitude sc01
A	3.14	3.19	2.95
B	**3.19**	**3.24**	**3.00**
JPEG2000	2.67	2.81	2.49

The proposed algorithm performs consistently better than the JPEG2000 with both tested parameter sets, 'A' and 'B', and with all the three images.

7 Conclusions

The purpose of this thesis was to develop a new lossless compression algorithm for hyperspectral imagery. The starting point was a set of requirements. Requirements such as, fast spatial and spectral access and scalability in resolution, were chosen to guarantee better usability in GIS applications as discussed in chapter (2). Fast spatial and spectral access were implemented by use of spatial tiling as described in section (5.2) and by use of band packs as described in section (5.3). Furthermore, losslessness guarantees that every bit of information contained in the image is preserved while the size of the file reduces to approximately 1/3 of the original.

Compression is achieved first by decorrelating the hyperspectral data followed by entropy coding the residuals. Decorrelation is performed in two stages. First, spatial decorrelation is done by intra-band decorrelator. Wavelet transform was chosen as intra-band decorrelator as it provides scalability in resolution (one of the requirements). More specifically, 5/3 integer to integer mapping wavelet transform was chosen because of its remarkable qualities: no need for multipliers and good decorrelation efficiency (see section (5.2)).

Second stage of decorrelation performs spectral decorrelation, that is, decorrelation between the bands of the image or inter-band decorrelation. For this, novel adaptive transform domain prediction scheme was developed. In the scheme, bands are predicted from preceding bands with a linear predictor. Prediction is performed and coefficients are adapted in transform domain. The technique for adaptation is based on assumption that the MSE optimal prediction coefficients do not differ significantly between different resolutions of the image. Thus, coefficients of lower resolution image are used as an estimate for the next higher resolution image. For details of the implementation see section (5.3).

The current implementation of inter-band decorrelation may, in some cases, result in slightly slower image loading when compared to raw data because the linear predictor causes dependence between current band and all the preceding bands within the same band pack. That is, the load time is usually I/O bound and dependence to preceding bands forces the algorithm to load not only the current band but also the preceding bands. This dependency is limited by the use of band packs as discussed in section (5.3). It was also noted that for the I/O transfers to match that of raw data the band pack should consist of 6 bands. That is, on average only 3 bands has to be read. With 3:1 compression ratio this correspond to 1 uncompressed band.

However, 6 bands per band pack is not enough to achieve the full potential of the algorithm. It can be seen from figure (6–4) that the increase in compression ratio is "fast" around 6 bands per band pack. That is, increasing number of bands in a band

pack increases significantly compression ratio. 16 bands per band pack was chosen for the tests as the compression ratio does not increase significantly from there on.

This result suggests to swap I/O time for processor time. That is, to use a more complex predictor in order to achieve the same compression ratio with a smaller band pack. The order of the predictor should be increased from 2. However, if one aims to keep average 3:1 compression ratio and maximum of 6 bands per band pack, then the predictor order needs not to be more than 5.

There is room for more research on the adaptation algorithm. Currently adaptation algorithm uses only the next coarser version or the data available from the current resolution (that is one of the detail parts). There is no reason why the coefficients should be based solely on single part. For example, correlation matrices of previously used detail parts could be linearly combined, similarly as in recursive least squares algorithm. Furthermore, the prediction coefficients could also be adapted spatially. However, spatial adaptation may lead to inaccurate estimate for the coefficients, especially with small tile sizes.

Entropy coding is the final stage. 2D adaptive Golomb-Rice entropy coder was chosen because of its computational simplicity and relatively good performance for task at hand. Adaptivity was achieved with combination of closed form estimate of the Golomb parameter and a 2D moving average filter (leaky integrator). For details see section (5.4).

One of the requirements was a possibility for a lossy preview. This is provided by wavelet transform. For example, one may load the coarsest level of the first band of a band pack very efficiently. The coarsest level serves as a preview. However, technique used in JPEG2000 can provide similar preview in full resolution but with lower signal-to-noise ratio. Possibility for incorporating similar functionality without sacrificing low complexity should be researched. This could be possible with slight modification of the predictor coefficient adaptation algorithm and Golomb parameter adaptation algorithm. However, the impact on compression ratio is not clear.

When compared to other state of the art lossless compressors the proposed algorithm performed well. Out of all 5 tested algorithms (JPEG2000, JPEG-LS, ICER-3D, Fast Lossless and the proposed algorithm) the proposed algorithm gave the best results except for one particular hyperspectral image. For that image Fast Lossless, an algorithm based on adaptive linear prediction, performed slightly better in terms of compression ratio than the proposed algorithm. For other images Fast Lossless resulted as second. However, one has to keep in mind that the proposed algorithm is more flexible than Fast Lossless (e.g., the proposed algorithm provides scalability in resolution), although it is not as flexible or "transmission safe" as ICER-3D. ICER-3D came consistently on 3rd place in terms of compression ratio. Without doubt, JPEG-LS performs worst of all tested algorithms because it does not provide adequate spectral decorrelation. For details see section (6.2).

The Matlab implementation proved to be inadequate for real life use, as expected. The overhead of scripting language and the practically uncontrollable use of dynamic memory slows the algorithm down. In future version the implementation of the algorithm should be coded in a language in which the memory usage can be better

controlled and where the language itself does not pose large overhead over the algorithm. Better use of multicore or multiprocessor systems and possibly the use of GPU to assist in computations should be considered.

As a conclusion the proposed algorithm filled all the requirements posed to it. Algorithmic complexity was kept at minimum and competitive compression ratio was achieved. However, next version of the algorithm should address the issue of I/O-operation minimization and provide more flexible preview possibilities with minimal loading times.

Bibliography

Adams, M. and Kossentini, F. (2000). Reversible integer-to-integer wavelet transforms for image compression: performance evaluation and analysis. *IEEE Transactions on Image Processing*, 9:1010–1024.

Akansu, A. and Haddad, R. (2001). *Multiresolution Signal Decomposition: Transforms, Subbands, Wavelets*. Academic Press.

Burt, P. and Adelson, E. (1983). The Laplacian Pyramid as a Compact Image Code. *IEEE Transactions on Communications*, COM-31:532–540.

Calderbank, R., Daubechies, I., Sweldens, W., and Yeo, B. (1998). Wavelet transforms that map integers to integers. *Appl. Comput. Harmon. Anal.*, 5(3):332–369.

Christopoulos, C., Skodras, A., and Ebrahimi, T. (2000). The JPEG2000 still image coding system: an overview. *IEEE Transactions on Consumer Electronics*, 46:1103–1127.

Daubechies, I. (1992). *Ten Lectures on Wavelets*. Society for Industrial and Applied Mathematics.

Daubechies, I. and Sweldens, W. (1998). Factoring Wavelet Transforms into Lifting Steps. *J. Fourier Anal. Appl.*, 4(3):245–267.

Fliege, N. (2000). *Multirate Digital Signal Processing*. John Wiley and Sons.

Golomb, S. (1966). Run-length encodings. *IEEE Transactions on Information Theory*, pages 399–401.

Gonzalez, R. and Woods, R. (2002). *Digital Image Processing*. Prentice Hall Inc., international 2nd ed. edition.

Hankerson, D., Harris, G., and Johnson, P. (1998). *Introduction to Information Theory and Data Compression*. CRC Press.

Huffman, D. (1952). A method for the construction of minimum-redundancy codes. *Proceedings of the I.R.E.*, pages 1098–1102.

Jain, A. (1989). *Fundamentals of Digital Image Processing*. Prentice Hall.

Jawerth, B. and Sweldens, W. (1994). An overview of wavelet based multiresolution analyses. *SIAM Rev.*, 36:377–412.

Kakadu JPEG2000 implementation website (2006). http://www.kakadusoftware.com.

Kiely, A., Klimesh, M., Xie, H., and Aranki, N. (2006). ICER-3D: A Progressive Wavelet-Based Compressor for Hyperspectral Images. *The Interplanetary Network Progress Report*, 42-164.

Klimesh, M. (2005). Low-Complexity Lossless Compression of Hyperspectral Imagery via Adaptive Filtering. *The Interplanetary Network Progress Report*, 42-163.

Kubasova, O. and Toivanen, P. (2004). Lossless compression methods for hyperspectral images. In *Proceedings of the 17th International Conference on Pattern Recognition*, volume 2, pages 803–806.

Landgrebe, D. (2003). *Signal Theory Methods in Multispectral Remote Sensing*. Wiley-Interscience.

Mallat, S. (1989). A Theory for Multiresolution Signal Decomposition: The Wavelet Representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11:674–693.

NASA AVIRIS website (2006). http://aviris.jpl.nasa.gov.

Oppenheim, A., Schafer, R., and Buck, J. (1999). *Discrete-Time Signal Processing 2nd ed.* Prentice Hall.

Quayle, B., Sohlberg, R., and Descloitres, J. (2004). Operational remote sensing technologies for wildfire assessment. *Geoscience and Remote Sensing Symposium, Proceedings. IEEE International*, 3:2245–2247.

Scott, J. (1997). *Remote Sensing: The image chain approach*. Oxford University Press.

Shannon, C. (1948). A Mathematical Theory of Communication. *The Bell System Technical Journal*, 27:379–423, 623–656.

Strang, G. and Nguyen, T. (1996). *Wavelets and Filter Banks*. Wellesley-Cambridge Press.

Sweldens, W. (1996). The lifting scheme: A custom-design construction of biorthogonal wavelets. *Appl. Comput. Harmon. Anal.*, 3(2):186–200.

Sweldens, W. (1997). The lifting scheme: A construction of second generation wavelets. *SIAM J. Math. Anal.*, 29(2):511–546.

Taubman, D. and Marcellin, M. (2002). JPEG2000: Standard for Interactive Imaging. *Proceedings of the IEEE*, 90:1336–1357.

Wang, Y., Ostermann, J., and Zhang, Y.-Q. (2001). *Video Processing and Communications*.

Weinberger, J., Seroussi, G., and Sapiro, G. (1999). From LOCO-I to the JPEG-LS Standard. In *Proceedings of the 1999 International Conference on Image Processing*, volume 4, pages 68–72.

Weinberger, M., Seroussi, G., and Sapiro, G. (2000). The LOCO-I Lossless Image Compression Algorithm: Principles and Standardization into JPEG-LS. *IEEE Transactions on Image Processing*, 9:1309–1324.

Wu, X. and Memon, N. (1996). CALIC - A Context Based Adaptice Lossless Image Codec.