Wavelets and Vector Quantization for Image Compression

door Pieter Ganseman

Promotor: prof. dr. ir. G. De Mey
Promotor: prof. E. Smigiel

Afstudeerwerk ingediend tot het behalen van de graad van
Burgerlijk Elektrotechnisch Ingenieur

Academiejaar 2004-2005
The completion of this thesis would have not been possible without the support and cooperation of many people. I am very grateful to all of them.

I wish to thank my parents José and Monique for giving me the opportunity to pursue my studies. My fiancée Christelle, I am thankfull for the love and support you gave me during this project.

I am very grateful to Prof. G. De Mey from UGent and Prof. R. Troester from INSA for creating the bilateral Erasmus agreement that allowed me to study at INSA. I spent a wonderful year in Strasbourg that I will remember for the rest of my life.

Many thanks to prof. E. Smigiel for providing me the possibility for this research and the freedom to explore my research ideas. I greatly enjoyed our collaboration.
Toelating tot bruikleen
De auteur geeft de toelating dit afstudeerwerk voor consultatie beschikbaar te stellen en delen van het afstudeerwerk te copiëren voor persoonlijk gebruik. Elk ander gebruik valt onder de beperkingen van het auteursrecht, in het bijzonder met betrekking tot de verplichting de bron uitdrukkelijk te vermelden bij het aanhalen van resultaten uit dit afstudeerwerk.

Pieter Ganseman 21 juni 2005
Wavelets and Vector Quantization for Image Compression

door
Pieter Ganseman

Afstudeerwerk ingediend tot het behalen van de graad van
Burgerlijk Elektrotechnisch Ingenieur
Academiejaar 2004-2005
Universiteit Gent
Faculteit Toegepaste Wetenschappen
Vakgroep Elektronica en Informatiesystemen
Voorzitter: prof. dr. ir. J. Van Campenhout
Promotor: prof. dr. ir. G. De Mey
Promotor: prof. E. Smigiel

Summary

In this thesis lossy image compression systems are studied. A theoretical description of
the steps in the image transform paradigm is given. Much attention is given to DCT and
wavelet transforms. The JPEG and JPEG2000 standards are discussed and compared to
each other. It will be illustrated that JPEG2000 outperforms JPEG in compression rate
and flexibility, but is more complex.

The principles of vector quantization are explained and it is shown that competitive neural
networks can be used for vector quantization. Self-organizing maps and growing neural
gas are competitive neural networks that, in addition to performing vector quantization,
introduce a topology in their network.

Image compression schemes based on vector quantization and self-organizing maps are
reviewed and implemented. An error in a trendsetting publication on self-organizing
maps is outlined and it is shown and explained why self-organizing maps, contrary to
what was believed, do not perform as well as traditional vector quantization methods.

Next the value of vector quantization methods for image compression is discussed. In
the category of DCT based methods vector quantization yields slightly better results
at low-bit rates than JPEG. In the category of wavelet transform based methods the
introduction of vector quantization is somewhat artificial and results are not as good as
JPEG2000. In both cases vector quantization removes flexibility from the compression
system. Hence, vector quantization or self-organizing maps cannot improve upon today’s
image compression standards.

Index terms: Wavelets, DCT, vector quantization, self-organizing maps, image com-
pression, JPEG, JPEG2000
## Contents

1 Introduction .......................... 1
   1.1 Motivation and Goals .................. 1
   1.2 Outline ................................ 2
   1.3 Sources of Compression ................ 3
   1.4 Image Transform Paradigm ............. 5
   1.5 Performance Evaluation ............... 7

2 Image Transforms ..................... 9
   2.1 Definitions .......................... 10
   2.2 Reasons for Transforming ............. 13
   2.3 Karhunen-Loève Transform (KLT) ...... 14
   2.4 DFT and DCT .......................... 15
   2.5 Discrete-time Filters ................. 19
       2.5.1 Z-Transform ...................... 21
       2.5.2 Downsampling ..................... 22
       2.5.3 Upsampling ....................... 23
   2.6 Subband Filtering ..................... 24
   2.7 Perfect Reconstruction Filter Banks ... 27
   2.8 Orthonormal Discrete Wavelet Transform 29
   2.9 Multiresolution Analysis .............. 30
   2.10 Biorthogonal Wavelets ............... 36
   2.11 Haar Wavelet ......................... 38
   2.12 Choice of Wavelets .................. 39
   2.13 Daubechies wavelets .................. 41
   2.14 CDF Biorthogonal Wavelets .......... 43
   2.15 Subband Coding of Images ............ 45
3 Scalar Quantization and Entropy Coding
  3.1 Scalar Quantization .............................................. 47
  3.2 Entropy Coding .................................................. 48
  3.3 Variable-Rate Entropy Coding .................................. 48
    3.3.1 Huffman Coding ............................................ 49
    3.3.2 Arithmetic Coding ........................................ 50
  3.4 Run-Length Encoding ............................................ 50
  3.5 DPCM ................................................................. 51

4 JPEG and JPEG2000 .................................................. 52
  4.1 JPEG ................................................................. 52
    4.1.1 Image Transform ........................................... 52
    4.1.2 Scalar Quantization ....................................... 53
    4.1.3 Entropy Coding ............................................ 55
    4.1.4 Performance ................................................ 55
  4.2 JPEG2000 ............................................................ 58
    4.2.1 Motivation .................................................. 58
    4.2.2 Structure of JPEG2000 .................................... 58
    4.2.3 Performance ................................................ 59
    4.2.4 Breakthrough ? ............................................... 60

5 Vector Quantization .................................................. 62
  5.1 Definitions ...................................................... 62
  5.2 LBG and k-means Algorithms .................................... 65
  5.3 Neural Networks .................................................. 67
  5.4 Self-Organizing Maps .......................................... 71
  5.5 Growing Neural Gas .............................................. 75

6 VQ in image compression ............................................ 79
  6.1 VQ paradigm ...................................................... 79
  6.2 DCT/VQ ............................................................. 82
  6.3 Classified VQ .................................................... 83
  6.4 Finite-State VQ .................................................. 85
  6.5 DCT/SOM ............................................................ 87
7 SOM and image compression 89
  7.1 Tools ................................................................. 89
  7.2 DCT/VQ ............................................................... 90
  7.3 DCT/SOM ............................................................. 91
  7.4 Other SOM methods ............................................... 93

8 Evaluation 96
  8.1 Classified VQ of Kim & Lee ...................................... 96
  8.2 Comparison to JPEG, JPEG2000 ................................ 99

9 DWT/VQ 102
  9.1 DWT/VQ ............................................................. 102
  9.2 DWT/SQ ............................................................. 105

10 Conclusions 109
  10.1 Summary .......................................................... 109
  10.2 Future Work ...................................................... 111

Bibliography 115
List of Figures

1.1 Contrast Sensitivity Function[30] ........................................... 4
1.2 Image Transform Paradigm ..................................................... 5
1.3 USC database images: (a) Lena (b) Tiffany (c) Lake (d) Peppers (e) Elaine
(f) Boat (g) House ............................................................... 8

2.1 Examples of possible basis vectors for the two-dimensional space (a) Orthonormal case (b) Biorthogonal case .................. 13
2.2 cosine transform symmetry .................................................... 17
2.3 (a) Gray-scale values of original image block (b) Image block in the DFT transform domain (absolute values) (c) Image block in DCT domain. Note that to preserve a good scale, the DC component is omitted. ............ 18
2.4 Notation of a downsampling by two operation .......................... 22
2.5 (a) Original spectrum (b) intermediate spectrum (c) Spectrum after downsampling by two (d) Spectrum after upsampling by two .......... 23
2.6 Notation of an upsampling operation ....................................... 24
2.7 Scheme of a 2-channel Filter Bank .......................................... 25
2.8 Ideal 2-channel filter bank: (a) Original signal (b)-(c) Spectra after analysis filtering (d)-(e) Spectra after downsampling (f)-(g) Spectra after upsampling (h)-(i) Spectra after synthesis filtering (j) Reconstructed Spectrum .... 26
2.9 Scheme for a 3-level multiresolution decomposition $a^{(0)}[n] = a^{(3)}[n] + d^{(3)}[n] + d^{(2)}[n] + d^{(1)}[n]$ ................................. 30
2.10 3-stage analysis octave-band filter bank ................................. 34
2.11 3-stage synthesis octave-band filter bank ............................... 34
2.12 (a) Haar scaling function $\phi(x)$ (b) Fourier transform $\Phi(e^{j\omega})$ (c) Haar mother wavelet $\psi(x)$ (d) Fourier transform $\Psi(e^{j\omega})$ ................. 38
2.13 Daubechies scaling functions $\phi(x)$ and mother wavelets $\psi(x)$ in space domain and in Fourier domain $\Phi(e^{j\omega})$ and $\Psi(e^{j\omega})$ for different filter lengths
4-taps filter: (a) $\phi(x)$ (b) $\psi(x)$ (c) $\Phi(e^{j\omega})$ (d) $\Phi(e^{j\omega})$
6-taps filter: (e) $\phi(x)$ (f) $\psi(x)$ (g) $\Phi(e^{j\omega})$ (h) $\Phi(e^{j\omega})$
8-taps filter: (i) $\phi(x)$ (j) $\psi(x)$ (k) $\Phi(e^{j\omega})$ (l) $\Phi(e^{j\omega})$
12-taps filter: (m) $\phi(x)$ (n) $\psi(x)$ (o) $\Phi(e^{j\omega})$ (p) $\Phi(e^{j\omega})$

2.14 9/7 CDF scaling functions $\phi(x)$ and mother wavelets $\psi(x)$ in space domain and in Fourier domain $\Phi(e^{j\omega})$ and $\Psi(e^{j\omega})$
Analysis filter: (a) $\phi(x)$ (b) $\psi(x)$ (c) $\Phi(e^{j\omega})$ (d) $\Phi(e^{j\omega})$
Synthesis filter: (e) $\phi(x)$ (f) $\psi(x)$ (g) $\Phi(e^{j\omega})$ (h) $\Phi(e^{j\omega})$

2.15 (a) Positioning of the subbands in a one-scale decomposed image (b) Example of a one-scale decomposed image
2.16 (a) Positioning of the subbands in a 2-scale decomposed image. (b) 2-scale decomposed Lena image. The high-pass signals are enhanced.

3.1 Construction of Huffman code by a binary tree
3.2 Example of arithmetic entropy coder

4.1 Example of JPEG DCT, quantization and reconstruction [49]
4.2 Ordering of DCT coefficients before entropy coding
4.3 PSNR after JPEG Compression for Lena
4.4 Zoom of Lena’s face after JPEG compression (a) at 0.20 bpp (b) at 0.30 bpp (c) at 0.40 bpp (d) at 0.50 bpp
4.5 PSNR after JPEG2000 and JPEG Compression for Lena
4.6 Zoom of Lena’s face after JPEG2000 compression (a) at 0.40 bpp (b) at 0.20 bpp (c) at 0.10 bpp (d) at 0.04 bpp

5.1 A vector quantizer as the cascade of an encoder and a decoder
5.2 (a) Locally optimal codebook vectors $y_1, y_2$ for input $x_1, x_2, x_3, x_4$ (b) Globally optimal codebook vectors $y_1, y_2$ for input $x_1, x_2, x_3, x_4$
5.3 (a) Neuron model (b) Example of a two-layered feed-forward neural network
5.4 Competitive Neural Network Architecture
5.5 (a) topology of two-dimensional SOM with rectangular grid (b) Neighborhood $H_c[t]$ as a function of time: $t_1 < t_2 < t_3$
5.6 Training ”video” for SOM with input vectors randomly chosen in the unit square.
5.7 A ”welfare” SOM [22]
5.8 Weight vectors for inputs from a two-dimensional normal distribution (a) found by $k$-means algorithm (b) found by SOM
5.9  Training "video" for growing neural gas for an example input distribution.
   (a) 2 neurons (b) 3 neurons (c) 4 neurons (d) 5 neurons (e) 8 neurons (f) 12 neurons .................................................. 78
5.10  $4 \times 4$ SOM obtained by training the input set with SOM batch algorithm . 78

6.1  Example of a codebook $\{y_i\}$ for the case of clustered input vectors . . . . 80
6.2  Basic VQ: Lena at 0.5 bpp ........................................... 81
6.3  Classified Vector Quantization ........................................ 83
6.4  (a) Different horizontal edges (b) How to classify this block? .......... . 83
6.5  (a) DCT areas corresponding to edges (b) subvector partition [18] .... 84
6.6  (a) Notation of DCT coefficients (b) Signs of DCT coefficients for horizontal edges .................................................. 85
6.7  Side-Match VQ: the state is determined by $u$ and $v$ ...................... 86
6.8  First-Order Differential Coder ......................................... 87

7.1  Comparison of results of JPEG and DCT/SOM compression on Lena. (a)
   by Amerijckx et al. [1] (b) correct results .............................. 93

8.1  (a) Classes ordered in descending order (b) Classes ordered for best entropy coding .................................................. 97
8.2  Lena’s shoulder at 0.30 bpp (a) DCT/VQ (b) Classified DCT/VQ ...... 99
8.3  Comparison of VQ methods to JPEG,JPEG2000 for Lena .............. 100
8.4  Lena at 0.20 bpp (a) JPEG (b) JPEG2000 (c) basic DCT/VQ (d) classified DCT/VQ .................................................... 101

9.1  Intra-band vectors ...................................................... 103
9.2  Cross-band vectors ..................................................... 103
9.3  Bit plane coding ......................................................... 106
9.4  Spatial Orientation Tree ................................................ 107
List of Tables

2.1 CDF 9/7 Analysis Filter Coefficients ........................................ 43
3.1 Example of Huffman entropy coding ........................................... 49
7.1 DCT/VQ on Lena ................................................................. 90
7.2 Results of DCT/SOM compression on Lena. ............................... 91
7.3 results of Amerijckx et al. [1] on Lena. .................................... 92
8.1 Results for Lena compressed with different classified DCT/VQ configura-
tions .......................................................................................... 98
Chapter 1

Introduction

1.1 Motivation and Goals

Our digital age is one of information interchange and images are very important sources of information. Whether the interchange of images takes place via floppy disks, the internet, the mobile network or any other channel, transmission and storage resources are always scarce and should not be wasted. But raw digital image files are large and take up a lot of these scarce resources.

Luckily, efficient algorithms have been developed to diminish the space occupied by digital image files. Everybody is familiar with worldwide image compression formats as GIF and JPEG. GIF is mostly used for computer-drawn images, while JPEG is popular for complex natural images. We can safely say that the compression quality achieved by these standards suffices for broadband internet applications.

But the recent developments in mobile phone technology have created a further need for image compression. The electromagnetic spectrum is a precious natural source and data transmission by this medium is very costly. This creates a new need for compression formats that achieve moderate to good quality at very low bit-rates.

At the end of the eighties methods based on vector quantization were reported to yield good results in this application field. But this results were first overshadowed by the huge success of JPEG, later by promising new techniques based on wavelets. These wavelet techniques lead to the development of the new JPEG2000 standard. In the meantime, neural network approaches of vector quantization became well-known and this opened up
new perspectives. Self-organizing maps were shown to be promising neural networks for image compression.

In view of the research activities of the Laboratoire en Informatique et Intelligence Artificielle (LIIA) at INSA Strasbourg, where this research took place, the focus of this thesis was set on the application of vector quantization and self-organizing maps for lossy image compression. The following topics are researched in this thesis:

- What is the structure of successful image compression standards JPEG and JPEG2000?
- How can vector quantization be used in image compression?
- Is it possible to combine techniques used in JPEG and JPEG2000 with vector quantization to yield better compression?
- How do methods based on vector quantization compare to existing standards JPEG and JPEG2000?
- Can existing vector quantization methods be improved by self-organizing maps or other neural networks?
- What is the future of vector quantization in image compression?

1.2 Outline

The first part of this thesis is consecrated to a profound study on the basic steps of the image transform paradigm. I felt that this study was necessary to learn the keys points of successful image compression schemes and to provide the thorough understanding needed to identify these areas where further improvement can be made.

To introduce the subject I describe the sources of image compression, the structure of the transform paradigm and the common way to evaluate compression schemes. In the second chapter image transforms are scrutinized. The reasons for transforming are explained and the discrete cosine transform is presented. The discrete wavelet transform is another important transform. Because it is a recently discovered transform which uses interesting concepts, its theory is developed with much attention for the mathematical background.
The third chapter treats scalar quantization and entropy coding. Huffman coding and arithmetic coding are popular variable-rate entropy coders. Run-length coders are entropy coders that yield good results when long strings of the same symbol occur in its input. DPCM is a prediction technique that results in additional compression when used together with an entropy coder.

The fourth chapter focuses on the standards JPEG and JPEG2000. I present the motivations behind their development and explain their structure. They are compared to each other.

In the fifth chapter vector quantization is defined. The problem of finding an optimal vector quantizer is raised and the important LBG and $k$-means algorithms are presented as solutions for this problem.

Artificial Neural Networks are introduced. A link between neural networks and vector quantization is established by showing that the LBG algorithm fits in the theory of competitive neural networks. Self-organizing maps and growing neural gas networks are presented as competitive neural networks with interesting characteristics.

In the sixth chapter I present a literature review on the use of vector quantization in image compression. I explain the problems of basic vector quantization and some interesting solutions for these problems: DCT/VQ, classified VQ and finite-state VQ. A scheme based on DCT and SOM is explained.

In chapter seven I present results of simulations on compression schemes. I explain why self-organizing maps, contrary to what other authors have claimed, are not suited for image compression.

Chapter eight presents the results of simulations of a classified DCT/VQ scheme. This results are analyzed and compared to results from JPEG and JPEG2000. Chapter nine explains the problems related to wavelet/VQ combinations. Finally chapter ten gives the conclusion and discusses future research possibilities.

1.3 Sources of Compression

Image compression literature distinguishes two types of compression. In lossless compression the original image can be reconstructed exactly from the encoded image, while in lossy compression this is not possible. This means that lossy compression will introduce errors in the reconstructed image. On the other hand, the compression rate (the size of
the encoded image divided by the size of the original image) that can be obtained with lossy compression is far superior to the compression rate that can be obtained with lossless compression.

For images where absolutely no quality degradation can be allowed, for example certain medical or satellite images, lossless compression is needed, but the maximum compression rate will not be very high. For other image categories, for example computer generated graphics with large areas of the same color, lossless compression schemes lead to high compression rates. For natural images however, which are continuous-tone with complex shapes and shading, lossy compression techniques are required to achieve high compression rates.

Well-designed lossy compression systems are capable of encoding images in such a way that the errors introduced are not or hardly visible, and this for much higher compression rates than can be obtained with lossless compression. Reed [38] identified four primary mechanisms to exploit in the design of lossy image compression systems.

- *Local spatial correlation* between pixels in the image is the principal source of compression. Otherwise said: "If a pixel has color $C$, the probability that its neighbor pixels have a color resembling $C$ is high." All image compression systems use this property.

- Because of the limited number of photo receptors in the eye, the human visual system cannot distinguish patterns with high spatial frequencies. The frequency
response of the human visual system is characterized by the contrast sensitivity function (CSF). Fig. 1.1 shows the CSF as found by Mannos and Sakrison [30]. This effect is widely used, particularly in the formulation of quantization rules.

- **Visual masking** considers the reduced visual sensitivity in the vicinity of edges. This property of the visual system, although well known, is not effectively used in most existing coding techniques.

- **The oblique effect** of human vision states that human visual acuity is greater in the vertical and horizontal directions. Therefore, humans are less sensitive to diagonal edges than to horizontal or vertical ones. This property is successfully used in wavelet transform-based compression systems.

### 1.4 Image Transform Paradigm

An important class of lossy image compression schemes is based on a structure referred to as the image transform paradigm (fig. 1.2). The standards JPEG and JPEG2000 are based on this structure. It consists of the following operations:

**Image transform** The image is transformed such that most of the image's energy is concentrated in just a few of the transform coefficients. The transform has an inverse, so this step is lossless and the image could be reconstructed exactly after this step.

**Quantization** This operation will map a large number of possible inputs onto a smaller set of outputs. Mostly inputs and outputs are scalars and this operation is called scalar quantization. It is an irreversible operation and introduces loss. Because the preceding transform step typically concentrates the signal energy in few transform
coefficients, many input values will be near zero and will be mapped to the value of zero. Also, not all transform coefficients are equally important for the visual quality of the image and those coefficients that are less important can be quantized more coarsely.

**Entropy coding** This is the final and lossless coding step. Based on the statistics of the input (output of the quantizer), frequently occurring input symbols will be mapped to short-length bit sequences, while rare inputs will be mapped to longer-length bit sequences. This results in overall data compression.

The image transform paradigm is always applied to monochromatic images. It considers images simply as matrices of numbers. This is not a problem because color images can be considered the superposition of three monochromatic images. On computer screens color images are represented as a superposition of a red, a green and a blue image. We say that on computer screens images are represented in the *RGB color space*. A straightforward way of employing the image transform on a color image could be to apply it separately to the red, green and blue image.

However, the RGB color space is not very good for color image compression. There exists correlation between the red, green and blue components and by coding these color components apart, this correlation is not exploited. It is better to work in a color space where the three components are less correlated.

The YCbCr color space is such a color space. It has been shown that this color space yields excellent results compared to other color spaces [32]. The components Y,Cb,Cr that determine a color can be obtained from the components R, G, B from the RGB color space by

\[
\begin{bmatrix}
Y \\
Cb \\
Cr
\end{bmatrix} =
\begin{bmatrix}
+0.299 & +0.587 & +0.114 \\
-0.168736 & -0.331264 & +0.5 \\
+0.5 & -0.418688 & -0.081312
\end{bmatrix}
\begin{bmatrix}
R \\
G \\
B
\end{bmatrix}
\]  

(1.1)

A characteristic of this color space is that the Y or *luminance* component is much more important to the quality of the image then the Cb and Cr or *chrominance* components. Hence, the chrominance components can be encoded with fewer bits without affecting the image quality much. The YCbCr color space is used in both JPEG and JPEG2000 standards.
1.5 Performance Evaluation

To compare image compression schemes, they should be tried out on the same images. Most researchers use test images form the University of South California (USC) image database\(^1\). Fig. 1.3 shows some images that I refer to in this thesis. The size of these images is 512 × 512 pixels. Lena is the most frequently used test image. It exists in different versions. The monochromatic version I used is the luminance component of the color version in the USC database after YCbCr transform.

To assess the performance of an image compression scheme a mathematical quality measure is applied to the test image: peak-signal-to-noise ratio (PSNR). For a monochromatic test image of size \(N \times N\) it is defined by

\[
PSNR = 10 \log \frac{255^2}{\frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} \| \hat{P}_{ij} - P_{ij} \|^2} \quad (dB)
\]

(1.2)

with \(P_{ij}\) the gray-scale values of the original test image and \(\hat{P}_{ij}\) the gray-scale values of the compressed image. PSNR values are typically between 25 and 35 dB. It should be noted that PSNR is not a very good performance measure. It does not detect very well certain artifacts to which the human visual system is sensitive, for example block artifacts: distortions of the image characterized by the visibility of an underlying block encoding.

Advanced quality metrics have been developed in recent years to take into account the correct influence of these and other artifacts on visual image quality. They are called perceptual quality metrics [11]. However, most authors still prefer the simple PSNR measure. Together with a detailed visual inspection of test images, it suffices for evaluation of a compression scheme.

\(^1\)http://sipi.usc.edu/database
Figure 1.3: USC database images: (a) Lena (b) Tiffany (c) Lake (d) Peppers (e) Elaine (f) Boat (g) House
Chapter 2

Image Transforms

Image transforms are an essential part of all state-of-the-art lossy image compression standards. Our focus in this first chapter will be directed to the mathematical background of this image transform step. The reader will discover the reasons for transforming images and learn to understand why certain transforms are so effective.

I will start by defining images and image transforms. Separable, orthonormal and biorthogonal transforms will be presented as important special cases of the general transform. The goals and possible advantages of a transform step will be explained. After these generalities, we look in detail at the mathematics, advantages and disadvantages of some popular transforms.

The Karhunen-Loève transform is interesting from a theoretical point of view, because it can be shown that it minimizes the mean squared error between compressed and original image for a given number of transform coefficients. However, for reasons that will be explained, it is not possible to use the Karhunen-Loève transform in practical image compression schemes.

Other candidates are the discrete Fourier transform (DFT) and the discrete cosine transform (DCT). These transforms perform a frequency analysis of a signal. The DCT and the DFT are related and can both be considered an approximation of the Karhunen-Loève transform, but the DCT has several advantages.

A disadvantage of the DCT is the infinite-length of the basis functions. Because images are finite-length signals, a transform that expands a signal as a linear combination of finite-length basis functions was developed. This transform is called the Discrete Wavelet Transform (DWT). To understand its working, discrete-time filter concepts need to be
introduced.
We will review FIR filters and discuss filter implementation by means of linear and circular convolution. The upsampling and downsampling operations and their properties are explained. These basic elements can be used to construct the two-channel filter bank and obtain a subband filtering scheme. The interest of subband filtering will be made clear and perfect reconstruction filter banks will be presented as a practical implementation, together with some examples.

Next we will define wavelets and the orthonormal discrete wavelet transform. We will establish multiresolution analysis, a theory that gives rise to a fast implementation algorithm for the DWT by means of an octave-band filter bank. Next the concepts introduced for the orthonormal transform will be generalized to define the biorthogonal discrete wavelet transform. I will explain how to take the DWT of an image in practice. Finally we will discuss which properties good wavelets for image compression should have, take a look at some examples and explain their interest for image compression.

2.1 Definitions

In this thesis images are defined from a computer scientist’s point of view. Monochromatic images or images are defined as two-dimensional finite-length discrete-time signals \( x[m, n] \). Color images are the superposition of three monochromatic images. For mathematical convenience all images will be square: \( m, n \in [0, N - 1] \). The images will be represented with 8 bit-precision gray levels, that is, \( x[m, n] \in \mathbb{N} \) and \( 0 \leq x[m, n] \leq 255 \).

An image transform will transform an image of \( N \times N \) coefficients \( x[m, n] \) into \( N \times N \) other coefficients \( X[u, v] \). To be useful the coefficients \( x[m, n] \) have to be re-obtainable from the coefficients \( X[u, v] \): the transform is reversible.

Series expansions provide the mathematical framework for image transforms. To construct a series expansion of a signal \( x[m, n] \), a complete set of basis functions \( \{\phi(u, v, m, n)\} \) needs to be available so that we can write all \( x[m, n] \) as a linear combination of these basis functions:

\[
x[m, n] = \sum_{u=1}^{N} \sum_{v=1}^{N} X[u, v] \tilde{\phi}[u, v, m, n]
\]  

(2.1)

Because this equation permits to obtain the image \( x[m, n] \) from its transform coefficients \( X[u, v] \) it is called the inverse transform. The equation that finds the transform coefficients...
$X[u, v]$ is called the forward transform or *transform* and looks like

\[
X[u, v] = \sum_{m=1}^{N} \sum_{n=1}^{N} x[m, n] \phi[u, v, m, n]
\]  

(2.2)

A transform is said to be *separable* if the basis functions $\phi$ and $\tilde{\phi}$ can be written in the form [36]

\[
\phi[u, v, m, n] = \varphi_C[u, m]\varphi_R[v, n]
\]  

(2.3)

\[
\tilde{\phi}[u, v, m, n] = \tilde{\varphi}_C[u, m]\tilde{\varphi}_R[v, n]
\]  

(2.4)

where the subscripts $R$ and $C$ indicate row and column one-dimensional transform operations. A separable two-dimensional transform can be computed in 2 steps. First, a one-dimensional transform is taken along each row of the image yielding

\[
P[m, v] = \sum_{n=1}^{N} x[m, n] \phi_R[v, n]
\]  

(2.5)

Next, a second one-dimensional transform is taken along each column of $P[m, v]$

\[
X[u, v] = \sum_{m=1}^{N} P[m, v] \phi_C[u, m]
\]  

(2.6)

All image transforms used in this thesis are separable and can be implemented by one-dimensional transforms. In general a one-dimensional transform and its inverse satisfy

\[
X[u] = \sum_{m=1}^{N} x[m] \phi(m, u)
\]  

(2.7)

\[
x[m] = \sum_{u=1}^{N} X[u] \tilde{\phi}(m, u)
\]  

(2.8)

A first of two important types of transforms are *orthonormal transforms*. In the one-
dimensional case these transforms and their inverse satisfy

\[ X[u] = \langle \tilde{\phi}[m, u], x[m] \rangle = \sum_m \tilde{\phi}[m, u] x[m] \]  \hspace{1cm} (2.9) \]

\[ x[m] = \sum_m X[u] \phi[m, u] \]  \hspace{1cm} (2.10) \]

The set of basis functions \( \{\phi[m, u]\} \) is complete and comprises an orthonormal basis for the \( x[m] \). Hence, it satisfies the orthonormality condition \( \langle \phi[i, m], \phi[j, m] \rangle = \delta_{i,j} \) with \( \delta_{i,j} \) the kronecker-delta. Orthonormal transforms can be considered a special case of the general one-dimensional transform (2.7) and its inverse (2.8) for which \( \tilde{\phi}[m, u] = \phi[m, u] \).

A second important type are **biorthogonal transforms**. For these transforms both \( \tilde{\phi}[m, u] \) and \( \phi[m, u] \) comprise a complete, but not orthonormal, basis for the \( x[m] \). All \( x[m] \) can be written as a linear combination of each of these bases and this leads to two transforms \( X[u] \) and \( \tilde{X}[u] \). Each transform has a corresponding inverse transform. The transforms are

\[ X[u] = \langle \tilde{\phi}[m, u], x[m] \rangle \]  \hspace{1cm} (2.11) \]

\[ \tilde{X}[u] = \langle \phi[m, u], x[m] \rangle \]  \hspace{1cm} (2.12) \]

There inverses are given by

\[ x[m] = \sum_m \tilde{X}[u] \tilde{\phi}[m, u] = \sum_m X[u] \phi[m, u] \]  \hspace{1cm} (2.13) \]

The bases \( \{\tilde{\phi}[m, u]\} \) and \( \{\phi[m, u]\} \) are **dual bases** and satisfy the biorthogonality constraint

\[ \langle \phi[i, u], \tilde{\phi}[j, u] \rangle = \delta_{i,j} \]  \hspace{1cm} (2.14) \]

Fig. 2.1 shows an example of possible orthonormal and biorthogonal basis vectors for the two-dimensional space.
2.2 Reasons for Transforming

In real-life image signals important correlations between neighboring data (pixels) exist. Although this has not been proved, it is widely accepted and confirmed in practice that representing a signal by uncorrelated data leads to better compression. In general, it is not possible to obtain entirely decorrelated data, but an appropriate transform will lead to strongly decorrelated data.

A second advantage is that for well-chosen transforms, a large amount of signal energy will be compacted in a small number of transform coefficients. This permits to compress the signal by minimizing the number of retained transform coefficients while keeping distortion at an acceptable level.

A third advantage is that the new domain is often more appropriate for quantization using perceptual criteria. The transform coefficients are not all equally important for the visual quality of the image and larger errors can be allowed on those coefficients that are less important.

Finally, the advantage of transform coding comes at a low computational price, as fast algorithms exist for all popular transforms.
Chapter 2. Image Transforms

2.3 Karhunen-Loève Transform (KLT)

The idea behind the KLT is to apply a linear transform so that the transform coefficients are entirely decorrelated. We divide an image in \( n \) blocks of \( d \) pixels. From each block we create a \( d \)-dimensional DC-leveled column vector \( X \). (This means that the vectors must have zero-mean.) Then there are \( n \) vectors \( X_i = [x_i(1), x_i(2), \ldots x_i(d)]^T \) (\( i = 1 \ldots n \)).

To have decorrelated transform coefficients, the covariance matrix should be diagonal. I recall that the covariance \( \text{cov}(i, j) \) between 2 dimensions \( i \) and \( j \) is a measure for the variation from the mean with respect to each other:

\[
\text{cov}(i, j) = \frac{\sum_{k=1}^{d}(x_k(i) - \overline{x(i)})(x_k(j) - \overline{x(j)})}{n - 1} \tag{2.15}
\]

with \( \overline{x(i)} \) and \( \overline{x(j)} \) the mean value of dimensions \( i \) and \( j \). The covariance matrix is defined by

\[
R_X = \begin{bmatrix}
\text{cov}(1, 1) & \text{cov}(1, 2) & \ldots & \text{cov}(1, d) \\
\text{cov}(2, 1) & \ldots & \ldots & \ldots \\
\ldots & \ldots & \ldots & \ldots \\
\text{cov}(d, 1) & \ldots & \ldots & \text{cov}(d, d)
\end{bmatrix} \tag{2.16}
\]

Statistically we can consider the vectors \( X_i \) random vectors with jointly Gaussian distributed elements \( x_i(k) \). For such vector \( X \) the covariance matrix becomes \( R_X = E[XX^T] \) with \( E \) the expectation value[10]. Our goal is to find a matrix \( A \) such that the components of the transformed vector \( Y = AX \) will be uncorrelated. The covariance matrix for \( Y \) has to be diagonal and is given by \( R_Y = E[(AX)(AX)^T] = AR_XA^T \). The matrix \( R_X \) is symmetric and positive semidefinite, hence the Karhunen-Loève Transform is determined by the matrix \( A \) whose rows are the eigenvectors of \( R_X \). We order the rows of \( A \) so that \( R_Y = \text{diag}(\lambda_0, \lambda_1, \ldots, \lambda_{d-1}) \), where \( \lambda_0 \geq \lambda_1 \geq \ldots \geq \lambda_{d-1} \).

The following theorem is about the optimality of the KL-transform [10]:

**Theorem 2.3.1** Given a random vector \( X \), a linear transform matrix \( U \) and its inverse \( V \). We form the vector \( Y = UX \), the vector \( \hat{Y} \) that contains only the first \( m \) components of \( Y \) and the vector \( \hat{X} = V \hat{Y} \), then the squared error \( E[\|X - \hat{X}\|^2] \) is a minimum when the matrices \( U \) and \( V \) are the Karhunen-Loève Transform and its inverse.

This theorem tells us that the KLT is the ideal transform: it completely decorrelates the image in the transform domain, minimizes the mean squared error in image compression.
and packs the most energy in the fewest number of transform coefficients. However, there are some problems with the KLT, which make this transform not suited for practical image compression[41]:

1. The KLT is image dependent, since it depends on the covariance matrix.

2. The KLT is computationally complex, requiring \( O(N^4) \) operations to apply to an \( N \times N \) image.

3. In image compression we cannot afford to keep signal components exactly. Typically, one or more components of the transformed signal are quantized.

On the other hand, the Karhunen-Loève transform has proved to be a valuable for statistical feature extraction. In this context it is often called principal component analysis. It is used to reduce the dimensionality of problems and to transform interdependent coordinates into significant and independent ones. This application relates the KLT to self-organizing feature maps, that can also be used to this end (see section 5.4).

### 2.4 DFT and DCT

Most authors define the two-dimensional *discrete Fourier transform* (DFT) of an image \( x[m,n] \) as [36]

\[
X[u,v] = \frac{1}{N} \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} x[m,n] e^{-j \frac{2\pi}{N} (um+vn)}
\] (2.17)

where \( j = \sqrt{-1} \) and its inverse is given by

\[
x[m,n] = \frac{1}{N} \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} X[u,v] e^{j \frac{2\pi}{N} (um+vn)}
\] (2.18)

In analogy to electrical engineering, the \( X[1,1] \) coefficient is sometimes called the *DC component* and the other DCT coefficients are then referred to as *AC components*. The DC component always equals the average gray-scale value of the block in the spatial domain.

Because the basis functions are separable, the two-dimensional DFT and its inverse can be found by applying the one-dimensional DFT sequentially on the rows and columns.
This one-dimensional DFT and its inverse are most often defined as

\[ X[u] = \sum_{m=0}^{N-1} x[m] e^{-j\frac{2\pi m u}{N}} \quad (2.19) \]

\[ x[m] = \frac{1}{N} \sum_{u=0}^{N-1} X[u] e^{j\frac{2\pi m u}{N}} \quad (2.20) \]

In computational algorithms, one will prefer to implement the two-dimensional DFT by two one-dimensional transforms. This permits the use of fast algorithms (Fast Fourier Transform). The interest of the DFT for image compression can be explained as follows. In most images, large surfaces of more or less the same color exist. So if an image is divided in relatively small blocks (for example 8 × 8 pixels), we can expect strong inter-pixel correlations in each block. If such a block is transformed into the frequency domain, most of the information will be packed in the low-frequency components. Only edges blocks should contain important higher-frequency components. This means that many high-frequency components can be discarded or loosely quantized without significant loss in image quality.

However because of two major drawbacks the DFT is not used for image compression [17].

- The DFT is a complex transform. Real gray-scale values are transformed into complex transform coefficients. Complex number calculations are computationally heavier and the complex transform coefficients need more space to be stored than real transform coefficients.

- The DFT is equivalent to the discrete-time Fourier transform (2.36) of a periodically extended version of the block of data under consideration. When there are sharp discontinuities between the right and left side, and between the top and bottom of a block, there will be substantial high-frequency transform content. When the high-frequency DFT transform coefficients are discarded or loosely quantized, the Gibbs phenomenon [3] causes the boundary points to take on erroneous values, which appear in an image as block artifacts: the boundaries between adjacent subimages become visible because the boundary pixels of the blocks take on the mean values of the discontinuities formed at the boundary points.

These problems can be avoided by forcing symmetry as shown in Fig. 2.2.
By using this symmetry and normalizing the result, it can be shown that the DFT of this $2N \times 2N$ image reduces to [36]:

$$X[u, v] = \frac{2}{N} C(u)C(v) \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} x[m, n] \cos \left[ \frac{(2m + 1)\pi u}{2N} \right] \cos \left[ \frac{(2n + 1)\pi v}{2N} \right]$$ (2.21)

where $C(0) = \frac{2}{\sqrt{N}}$ and $C(w) = 1$ for $w = 1, 2, \ldots, N - 1$. This is the forward two-dimensional discrete cosine transform. Its inverse is

$$x[m, n] = \frac{2}{N} \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} X[u, v] C(u)C(v) \cos \left[ \frac{(2m + 1)\pi u}{2N} \right] \cos \left[ \frac{(2n + 1)\pi v}{2N} \right]$$ (2.22)

The two-dimensional DCT again is a separable transform. Its one-dimensional version and inverse are given by

$$X[u] = \sqrt{\frac{2}{N}} C(u) \sum_{m=0}^{N-1} x[m] \cos \left[ \frac{(2m + 1)\pi u}{2N} \right]$$ (2.23)

$$x[m] = \sqrt{\frac{2}{N}} \sum_{u=0}^{N-1} X[u] C(u) \cos \left[ \frac{(2m + 1)\pi u}{2N} \right]$$ (2.24)

Like the DFT, the DCT tends to concentrate the image energy toward the lower spatial frequencies. It has the advantage that it is a real-valued transform that generates real coefficients from real-valued data. It is also equivalent to the DTFT of a symmetrically extended version of the block of data under consideration. This means that the artificial discontinuities and unwanted high-frequency content introduced by the DFT are not introduced by the DCT. These advantages are the reason the DCT has become the transform used in the popular JPEG image compression standard.
Fig. 2.3 compares the DFT (absolute values) and the DCT for the upper left 16 × 16 block in the Lena image. (Note that the colors in the drawing are only for better visibility.) We can see that the DCT transform coefficients are considerably smaller than the DFT transform coefficients, especially near the borders, which makes the DCT coefficients better suited for quantization.

![Diagram of DFT and DCT coefficients](image)

Figure 2.3: (a) Gray-scale values of original image block (b) Image block in the DFT transform domain (absolute values) (c) Image block in DCT domain. Note that to preserve a good scale, the DC component is omitted.

What is more, it can be shown that the DCT (as well as the DFT) diagonalizes approximately the covariation matrix of a Gauss Markov source. A Gauss Markov source defines the elements $x[m]$ by

$$x[m + 1] = \alpha x[m] + w_m$$

(2.25)

where $\alpha$ is a regression coefficient with magnitude less than one and $w_m$ a zero-mean, unit
variance Gaussian source. The statistical properties of Gauss Markov sources resemble those of natural images. The DCT is asymptotically equivalent to the KLT when the block size tends to infinity. It should be noted that even though real images are of course no Gauss Markov processes, the DCT has proved to be a robust approximation to the KLT [41].

The DCT also has shortcomings. The image still has to be divided in blocks. This division is arbitrary and this can generate block artifacts at high compression rates. The division is necessary due to the fundamental fact that the cosine basis functions have infinite length. To avoid this artificial division a category of transforms with finite-length basis functions has been developed, such that block division is no longer necessary. They are called discrete wavelet transforms. To explain their working, some digital filter theory needs to be recalled.

### 2.5 Discrete-time Filters

Filtering is one of the most widely used signal processing operations. In many practical cases the filtering operation is linear shift-invariant (LSI) and may be described by the following relation

\[
y[n] = h[n] \ast x[n]
\]

\[
= \sum_{i=-\infty}^{\infty} h[i]x[n - i]
\]

\[
= \sum_{i=-\infty}^{\infty} x[i]h[n - i]
\]

(2.26)

where \( h[n] \) is the called the impulse response of the filter, \( x[n] \) the input and \( y[n] \) the output of the system. This relation also defines the linear convolution operation \( h[n] \ast x[n] \) for two discrete-time signals \( h[n] \) and \( x[n] \). An important category of discrete-time systems exhibits a finite-length impulse response. They are called FIR filters. They are very important in image processing because of their linear phase property, necessary to guarantee the absence of phase distortion and preserve the locality of edges. It is common to define \( h[n] \) on the interval \([0, N - 1]\). In practice the signal \( x[n] \) is also finite-length and defined on the interval \([0, M - 1]\). Outside these intervals the signals are 0. For these
$h[n]$ and $x[n]$ (2.26) reduces to

$$y[n] = h[n] * x[n]$$
$$= \sum_{i=0}^{N-1} h[i]x[n-i]$$
$$= \sum_{i=0}^{M-1} x[i]h[n-i]$$

(2.27)

The output signal $y[n]$ has length $M+N-1$ compared to the input signal $x[n]$’s length $N$. This coefficient expansion is a problem if we want to use filtering in compression systems, where the goal is to reduce, not increase, the amount of information to be coded.

To solve this problem, we can use circular convolution, rather than linear convolution, to implement the filtering operation. If we assume $M < N$ and we extend $h[n]$ with $h[n] = 0$ for $n = [M, N-1]$, both $h[n]$ and $x[n]$ have length $N$. Circular convolution is now defined by

$$y_c[n] = h[n] \odot x[n]$$
$$= \sum_{i=0}^{N-1} h[i]x[(n-i) \mod N]$$
$$= \sum_{i=0}^{N-1} x[i]h[(n-i) \mod N]$$

(2.28)

Here $y_c[n]$ has length $N$. Of course there is a price to pay: $y_c[n]$ is a distorted version of the correct filter output $y[n]$

$$y_c[n] = \begin{cases} 
    y[n] + y[N + n] & n = [0, M-1] \\
    y[n] & n = [M, N-1]
\end{cases}$$

(2.29)

In order to use circular convolution successfully, this unacceptable distortion should be eliminated. If we create a new input signal $x'[n]$ by symmetric extension of $x[n]$

$$x'[n] = \begin{cases} 
    x[n] & n = [0, N-1] \\
    x[2N-1-n] & n = [N, 2N-1] \\
    0 & otherwise
\end{cases}$$

(2.30)
then it can be shown that filtering $x'[n]$ with $h[n]$ leads to a symmetric output $y'[n]$ when $h[n]$ is symmetric and to an antisymmetric output $y'[n]$ when $h[n]$ is antisymmetric [34]. Such filters $h[n]$ are called linear phase filters. Circular convolution can only be used in combination with linear phase filters.

After all, the symmetric extension doubles the input signal’s length from $N$ to $2N$. Circular convolution of this signal will lead to a $2N$-length output signal $y'[n]$. When this output signal $y'[n]$ is (anti)symmetric, then half of the $2N$ values $y'[n]$ are redundant and can be omitted without loss of information. This results in an $N$-length filtered output that was, remember, obtained from an $N$-length input $x[n]$. When $h[n]$ is a non-linear phase filter, $y'[n]$ is not (anti)symmetric. This means that none of the $2N$ values $y'[n]$ can be omitted without loss of information. In this case circular convolution yields even worse signal expansion than linear convolution.

### 2.5.1 Z-Transform

When working with digital filters, it is sometimes advantageous to transform digital signals into the Z-transform domain (with $z \in \mathbb{C}$):

$$X(z) = \sum_{n=-\infty}^{\infty} x[n] z^{-n} \quad (2.31)$$

The Z-transform has some interesting properties [4].

- Given $x[n]$ with Z-transform $X(z)$, the following are transform pairs:

$$( -1)^n x[n] \longleftrightarrow X(-z) \quad (2.32)$$

$$x[n-k] \longleftrightarrow z^{-k}X(z) \quad (2.33)$$

$$x[-n] \longleftrightarrow X(\frac{1}{z}) \quad (2.34)$$

$$x[n] * h[n] \longleftrightarrow X(z)H(z) \quad (2.35)$$

This last expression tells us that a filter operation can be implemented in the Z-transform domain by multiplication of the Z-transform $X(z)$ of the signal $x[n]$ and Z-transform $H(z)$ of the filter’s impulse response $h[n]$.

- On the unit circle $z = e^{j\omega}$, the Z-transform reduces to a $2\pi$-periodic function: the
Discrete-Time Fourier Transform (DTFT):

\[ X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x[n]e^{-j\omega n} \] (2.36)

Its inverse is given by

\[ x[n] = \frac{1}{2\pi} \int_{-\pi}^{\pi} X(e^{j\omega})e^{j\omega n} d\omega \] (2.37)

### 2.5.2 Downsampling

Sometimes called subsampling or decimation, downsampling by a factor N means keeping only every N-th sample. In this thesis, only downsampling by two is used. Fig. 2.4 shows the typical notation.

![Image of downsampling notation](image)

Figure 2.4: Notation of a downsampling by two operation

This operation maps \( x[n] \) into \( y[n] = x[2n] \), that is, only even indexed samples of \( x[n] \) are kept. In the Z-transform domain we get:

\[ Y(z) = \frac{1}{2}(X(z^{\frac{1}{2}}) + X(-z^{\frac{1}{2}})) \] (2.38)

To understand where this expression comes from, we construct a function \( X' = \frac{1}{2}(X(z) + X(-z)) = \sum_n x[2n]z^{2n} \). Then, noting that only even powers of \( z \) are present, we can write \( Y(z) = X'(z^{\frac{1}{2}}) \). In time domain this gives:

\[
\begin{align*}
x[n] &= \ldots \ x[-2] \ x[-1] \ x[0] \ x[1] \ x[2] \ \ldots \\
x'[n] &= \ldots \ x[-2] \ 0 \ x[0] \ 0 \ x[2] \ \ldots \\
y[n] &= \ldots \ x[-4] \ x[-2] \ x[0] \ x[2] \ x[4] \ \ldots 
\end{align*}
\]

Evaluating (2.38) for \( z = e^{j\omega} \) yields the result of the downsampling by two operation in the Fourier domain:

\[ Y(e^{j\omega}) = \frac{1}{2}(X(e^{j\frac{\omega}{2}}) + X(e^{j(\frac{\omega}{2} - \pi)}) \] (2.39)
Fig. 2.5 (a)-(c) show the Fourier spectra of $X(e^{j\omega}), X'(e^{j\omega}), Y(e^{j\omega})$. We see that down-sampling by two corresponds to stretching the spectrum by two, and adding an aliased version at multiples of $2\pi$.

![Fourier spectra](image)

Figure 2.5: (a) Original spectrum (b) intermediate spectrum (c) Spectrum after downsampling by two (d) Spectrum after upsampling by two

Given an input signal $x[n]$, if we filter $x[n]$ with a filter $H(z)$, followed by downsampling by two, we get the signal $y[n]$:

$$y[n] = \sum_{i=\infty}^{\infty} h[2n - i]x[i] = \sum_{i=\infty}^{\infty} x[2n - i]h[i]$$

(2.40)

By making use of (2.38) and (2.35), we get this result in the Z-transform domain:

$$Y(z) = \frac{1}{2}(H(z^{\frac{1}{2}})X(z^{\frac{1}{2}}) + H(-z^{\frac{1}{2}})X(-z^{\frac{1}{2}}))$$

(2.41)

### 2.5.3 Upsampling

I will only consider the case of upsampling by two. Fig. 2.6 shows the notation of this operation that simply means adding a zero between every 2 input samples.
The output signal is
\[ y[2n] = x[n] \]
\[ y[2n + 1] = 0 \] (2.42)

In the Z-transform and Fourier domains this relation is expressed as:
\[ Y(z) = X(z^2) \] (2.43)
\[ Y(e^{j\omega}) = X(e^{j2\omega}) \] (2.44)

Fig. 2.5 (d) shows that this operation corresponds to a contraction in the Fourier domain.

Given an input signal \( x[n] \), if we upsample this signal by two followed by filtering with \( H(z) \), we get the signal \( y[n] \):
\[ y[n] = \sum_{i=-\infty}^{\infty} h[n - 2i]x[i] \] (2.45)

By making use of (2.43) and (2.35), we get this result in the Z-transform domain:
\[ Y(z) = X(z^2)H(z) \] (2.46)

### 2.6 Subband Filtering

There exist many applications in modern signal processing where it is advantageous to separate a signal into different frequency ranges called subbands, for example in image compression. Most natural images contain far more low-frequency content than high-frequency content. After subband filtering, one could allocate more bits to subbands that contain a lot of information and fewer bits to subbands that contain less information in order to achieve efficient lossy compression.

We can separate a digital signal \( x[n] \) of bandwidth \( f_b \) in a signal \( x_l[n] \) with frequency
range $[0, \frac{f_b}{2}]$ and a signal $x_h[n]$ with frequency range $[\frac{f_b}{2}, f_b]$. The signals $x_l[n]$ and $x_h[n]$ are obtained by filtering with an ideal low-pass filter $H_0(z)$ and an ideal high-pass filter $H_1(z)$ respectively, both with cut-off frequency $\frac{f_b}{2}$. Because the signals $x_l[n]$ and $x_h[n]$ have only half the bandwidth of the signal $x[n]$, they can be downsampled by 2 without loss of information. We call the downsampled signals $y_l[n]$ and $y_h[n]$. It is now possible to reconstruct the signal $x[n]$ from the signals $y_l[n]$ and $y_h[n]$.

By upsamling by two of signals $y_l[n]$ and $y_h[n]$ the signals $q_l[n]$ and $q_h[n]$ are obtained. If we now filter $q_l[n]$ and $q_h[n]$ with an ideal low-pass filter $G_0(z)$ and an ideal high-pass filter $G_1(z)$ respectively, both with cut-off frequency $\frac{f_b}{2}$, we obtain the signals $p_l[n]$ and $p_h[n]$. By summing $p_l[n]$ and $p_h[n]$, we obtain the signal $r[n]$. In this ideal case $r[n] = x[n]$.

Fig. 2.7 shows a symbolic representation of these operations and Fig. 2.8 shows what happens in the Fourier domain. Filters $H_0(z)$ and $H_1(z)$ together are called the analysis filter bank, while filters $G_0(z)$ and $G_1(z)$ together are called the synthesis filter bank.

This presented filter scheme cannot be realized in practice because of the ideal filter assumption. However, it is possible to find other 2-channel perfect reconstruction filter banks (PRFB) for which $r[n] = x[n]$. In the next section, I will explain which conditions the filters $H_0(z)$, $H_1(z)$, $G_0(z)$, $G_1(z)$ should satisfy in order to comprise a PRFB.
Figure 2.8: Ideal 2-channel filter bank: (a) Original signal (b)-(c) Spectra after analysis filtering (d)-(e) Spectra after downsampling (f)-(g) Spectra after upsampling (h)-(i) Spectra after synthesis filtering (j) Reconstructed Spectrum
2.7 Perfect Reconstruction Filter Banks

We will describe the scheme of fig. 2.7 in the Z-transform domain in order to find the conditions for a PRFB. By combining (2.38) and (2.43), we find that the combination of downsampling and upsampling gives the following relation between \( q_l[n] \) and \( x_l[n] \) in the Z-transform domain:

\[
Q_l(z) = \frac{1}{2} (X_l(z) + X_l(-z)) \tag{2.47}
\]

A similar equation holds for \( q_h[n] \) and \( x_h[n] \). We find now for the complete scheme:

\[
R(z) = \frac{1}{2} G_0(z) [H_0(z) X(z) + H_0(-z) X(-z)] + \frac{1}{2} G_1(z) [H_1(z) X(z) + H_1(-z) X(-z)]
\]

(2.48)

We rewrite this:

\[
R(z) = \frac{1}{2} [G_0(z) H_0(z) + G_1(z) H_1(z)] X(z) + \frac{1}{2} [G_0(z) H_0(-z) + G_1(z) H_1(-z)] X(-z)
\]

(2.49)

We now require \( R(z) \) to be a delayed copy of \( X(z) \):

\[
r[n] = x[n - k] \tag{2.50}
\]

or,

\[
R(z) = z^{-k} X(z) \tag{2.51}
\]

From equation (2.49) follows that the following conditions need to hold in order to have perfect reconstruction:

\[
G_0(z) H_0(z) + G_1(z) H_1(z) = 2 z^{-k} \tag{2.52}
\]

\[
G_0(z) H_0(-z) + G_1(z) H_1(-z) = 0 \tag{2.53}
\]

Condition (2.52) is called the distortion condition and condition (2.53) is called the aliasing condition. In the classic Quadrature Mirror Filters (QMF) solution [14], the aliasing is removed by defining

\[
G_0(z) = H_0(z) \tag{2.54}
\]

\[
G_1(z) = -H_0(-z) \tag{2.55}
\]

\[
H_1(z) = H_0(-z) \tag{2.56}
\]
So the QMF approach defines all filters by the low-pass filter $H_0(z)$. The distortion condition (2.52) becomes

$$H_0^2(z) - H_0^2(-z) = 2z^{-k} \quad (2.57)$$

The above relation explains the name QMF. On the unit circle $H_0(-z) = H_0(e^{j(\omega+\pi)})$ is the mirror image of $H_0(z)$ and both mirror images are squared. By (2.54)-(2.56), filters $G_0, G_1, H_1$ satisfy a relation like (2.57) and are QMF as well.

It can be shown that no non-trivial exact FIR filter solutions for equation (2.57) exists, but many good approximative solutions have been designed. A popular class for image compression has been designed by Johnston [20].

Smith and Barnwell [46] designed a class of FIR filters for which perfect reconstruction is possible: the conjugate quadrature filters (CQF). First they removed aliasing by defining the synthesis filters

$$G_0(z) = H_1(-z) \quad (2.58)$$
$$G_1(z) = -H_0(-z) \quad (2.59)$$

Then they defined CQF by the following requirement, where $N$ is odd:

$$H_1(z) = -H_0(-z)z^{-N} \quad (2.60)$$

Then by (2.58) and (2.59) filters $G_0(z)$ and $G_1(z)$ satisfy an equivalent equation and are CQF too. Without restrictions we can take $k = N$ and in this case the distortion condition becomes

$$H_0(z)H_0(z^{-1}) + H_0(-z)H_0(-z^{-1}) = 2 \quad (2.61)$$

In the Fourier domain $z = e^{j\omega}$ we have $H_0(e^{-j\omega}) = \overline{H_0(e^{j\omega})}$, and condition (2.61) becomes

$$|H_0(e^{j\omega})|^2 + |H_0(e^{j(\omega+\pi)})|^2 = 2 \quad (2.62)$$

The problem with the solution proposed in [46] was the somewhat arbitrary choice of the filters $H_0$ that satisfy (2.62).

A filter pair $\{P(e^{j\omega}), Q(e^{j\omega})\}$ is defined to be power complementary if

$$|P(e^{j\omega})|^2 + |Q(e^{j\omega})|^2 = C \quad \text{with } C \text{ a constant} \quad (2.63)$$
From condition (2.62) follows that \( \{ H_0(e^{j\omega}), H_0(e^{j(\omega+\pi)}) \} \) should be a power complementary filter pair.

We will see that the discrete wavelet transform implementation by means of multiresolution analysis will give rise to a power complementary CQF pair and that proper wavelet design will lead to desirable CQF pair characteristics.

## 2.8 Orthonormal Discrete Wavelet Transform

Now that all the necessary concepts from filter bank theory have been introduced, we can bring up wavelets. Let us start by defining what we understand by a wavelet. Wavelets are continuous-time functions in \( L^2(\mathbb{R}) \) \(^1\) obtained by shifting and scaling a mother wavelet \( \psi(x) \). Most wavelets used in practice are dyadic wavelets

\[
\psi_{j,n}(x) = 2^{-j/2} \psi(2^{-j}x - n) \tag{2.64}
\]

Wavelets are constructed so that \( \{ \psi_{j,n}(x) \mid n \in \mathbb{Z} \} \) (i.e., the set of all shifted wavelets at a fixed scale \( j \)), describes a particular level of detail in the signal. As \( j \) decreases (\( j \to -\infty \)), the wavelets become more compact and the level of detail increases. This feature makes wavelets good candidates for image compression basis functions, as they permit ”to see the forest and the trees”.

It is possible to find complete orthonormal bases of wavelets \( \psi_{j,n}(x) \) for the space \( L^2(\mathbb{R}) \). These bases satisfy the orthonormality condition

\[
\langle \psi_{j,n}(x), \psi_{i,m}(x) \rangle = \delta_{i,j} \delta_{n,m} \tag{2.65}
\]

and can be used to expand a function \( f(x) \in L^2(\mathbb{R}) \). This expansion defines the orthonormal discrete wavelet transform (ODWT) and its inverse:

\[
d^{(j)}[n] = \langle \psi_{j,n}(x), f(x) \rangle = \int_{-\infty}^{\infty} \psi_{j,n}(x)f(x)dx \tag{2.66}
\]

\[
f(x) = \sum_{j=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} d^{(j)}[n] \psi_{j,n}(x) \tag{2.67}
\]

At this point we are confronted with two problems

\(^1f \in L^2(\mathbb{R}) \iff \int_{-\infty}^{\infty} |f(x)|^2dx < \infty\)
1. How to find complete orthonormal bases of wavelets $\psi_{j,n}(x)$?

2. Is it possible to find a fast algorithm to calculate the transform coefficients $d^{(j)}[n]$, that is, without calculating continuous-time integrals?

The answers to these questions are given by the *multiresolution analysis* theory [29] that provides a mathematical framework for the construction of wavelet bases, gives a fast algorithm for calculation of the transform coefficients and links wavelet theory to the field of subband coding.

### 2.9 Multiresolution Analysis

The mathematical theory of multiresolution analysis will permit us to analyze the decomposition of a signal into a sequence of coarser approximations and detail signals that are mutually orthonormal. This orthonormality ensures that there is no redundancy in the representation of the signal. Every level of decomposition results in one approximation $a^{(j)}[n]$ and one detail signal $d^{(j)}[n]$. The obtained approximation signal $a^{(j)}[n]$ is further decomposed in a coarser approximation $a^{(j+1)}[n]$ and detail signal $d^{(j+1)}[n]$. This scheme can be iterated. The reconstruction of the original signal is done by successively adding finer detail signals to the coarsest approximation signal and every reconstruction level improves resolution of the signal by a factor of 2. Fig. 2.9 shows a schematic representation of a multiresolution decomposition.

![Figure 2.9: Scheme for a 3-level multiresolution decomposition](image)

We begin the theory with a formal definition:

**Definition** A *multiresolution analysis* in $L^2(\mathbb{R})$ consists of a sequence of successive approximation spaces $V_j$. These closed subspaces $V_j$ satisfy the following conditions:
Chapter 2. Image Transforms

1. (Increasing)
\[ \cdots V_2 \subset V_1 \subset V_0 \subset V_{-1} \subset V_2 \cdots \quad (2.68) \]

2. (Density)
\[ \bigcup_{j \in \mathbb{Z}} V_j = L^2(\mathbb{R}) \quad (2.69) \]

3. (Separation)
\[ \bigcap_{j \in \mathbb{Z}} V_j = \{0\} \quad (2.70) \]

4. (Scale Invariance)
\[ f(x) \in V_j \iff f(2^j x) \in V_0 \quad (2.71) \]

5. (Shift Invariance)
\[ f(x) \in V_0 \iff f(x - n) \in V_0, \text{ for all } n \in \mathbb{Z} \quad (2.72) \]

6. (Existence of a Basis)

There exists a function \( \phi(x) \in V_0 \), such that \( \{\phi(x - n) \mid n \in \mathbb{Z}\} \) is an orthonormal basis for \( V_0 \). (2.73)

This function \( \phi \) is called the scaling function

Starting from this definition, some interesting properties can be derived: By combining (2.71) - (2.73), we find:

\[ \{\phi_{j,n} = 2^{-j/2}\phi(2^{-j}x - n) \mid n \in \mathbb{Z}\} \text{ is an orthonormal basis for } V_j. \quad (2.74) \]

Now, since \( \phi(x) \in V_0 \) and \( V_0 \subset V_{-1} \), we have \( \phi(x) \in V_{-1} \). The scale invariance property (2.71) implies that \( \phi(2^{-1}x) \in V_0 \). The existence of a basis (2.73) guarantees that we can expand:

\[ \phi(2^{-1}x) = \sum_{n=-\infty}^{\infty} c_n \phi(x - n) \quad (2.75) \]

We can rewrite this in the form:

\[ \phi(x) = \sqrt{2} \sum_{n=-\infty}^{\infty} g_0[n] \phi(2x - n) \quad (2.76) \]
This important equation is called the refinement equation or two-scale equation. We can now state a fundamental theorem:

**Theorem 2.9.1** If the sequence of successive spaces \((V_j)_{j \in \mathbb{Z}}\) is a multiresolution analysis with scaling function \(\phi(x)\), then there exists an orthonormal wavelet basis \(\{\psi_{j,n} \mid j, n \in \mathbb{Z}\}\) for \(L^2(\mathbb{R})\). This basis is given by:

\[
\psi_{j,n}(x) = 2^{-j/2} \psi(2^{-j}x - n), \quad j, n \in \mathbb{Z} 
\]

(2.77)

with

\[
\psi(x) = \sqrt{2} \sum_{n=-\infty}^{\infty} (-1)^n g_0[-n + 1] \phi(2x - n) 
\]

(2.78)

where the \(g_0[-n + 1]\) are the coefficients from the refinement equation (2.76). This basis is such that \(\{\psi_{j,n} \mid n \in \mathbb{Z}\}\) is an orthonormal basis for \(W_j\), where \(W_j\) is the orthonormal complement of \(V_j\) in \(V_{j-1}\). We write: \(V_{j-1} = V_j \oplus W_j\).

A complete proof of this theorem can be found in [9]. It is interesting to explain where equation (2.78) comes from. Because \(V_{-1} = V_0 \oplus W_0\), an orthonormal set of basis functions \(\{\psi(x - n)\}\) for \(W_0\) satisfies \(\psi(x) \in V_{-1}\). With the same argumentation used to find the refinement equation (2.76) we obtain

\[
\psi(x) = \sqrt{2} \sum_{n=-\infty}^{\infty} g_1[n] \phi(2x - n) 
\]

(2.79)

One can proof that the choice

\[
g_1[n] = (-1)^n g_0[-1 + n] 
\]

(2.80)

satisfies the orthonormality condition between \(V_0\) and \(W_0\) to obtain equation (2.78).

Iterating the theorem learns us that a signal in \(V_0\) can be orthonormally decomposed into a signal in the approximation space \(V_j\) and \(j\) signals in detail spaces \(W_j, \ldots, W_1\).

\[
V_0 = V_j \oplus W_j \oplus W_{j-1} \oplus \ldots \oplus W_2 \oplus W_1 
\]

(2.81)

A function \(f(x)\) in \(L^2(\mathbb{R})\) can be projected on \(V_0\) to obtain \(\tilde{f}(x) \in V_0\)

\[
\tilde{f}(x) = \sum_{n=-\infty}^{\infty} a^{(0)}[n] \phi(x - n), \quad \text{where } a^{(0)}[n] = \langle \phi(x - n), f(x) \rangle
\]

(2.82)
Chapter 2. Image Transforms

Hence, since \( \{ \phi_{j,n} \mid n \in \mathbb{Z} \} \) is an orthonormal basis for \( V_j \) and the \( \{ \psi_{k,n} \mid n \in \mathbb{Z} \} \) are orthonormal bases for the \( W_k \), this function \( \tilde{f}(x) \in V_0 \) can be decomposed as

\[
\tilde{f}(x) = \sum_{n=-\infty}^{\infty} a^{(j)}[n] \phi_{j,n}(x) + \sum_{k=1}^{j} \sum_{n=-\infty}^{\infty} d^{(k)}[n] \psi_{k,n}(n)
\]

(2.83)

with

\[
a^{(j)}[n] = \langle f(x), \phi_{j,n}(x) \rangle \quad \text{and} \quad d^{(k)}[n] = \langle f(x), \psi_{k,n}(x) \rangle
\]

(2.84)

The coefficients \( a^{(j)}[n] \) of the projection onto \( V_j \) are called the discrete approximation of \( \tilde{f}(x) \) at the resolution \( 2^j \) and the coefficients \( d^{(j)}[n] \) of the projection onto \( W_k \) are called the detail coefficients at resolution \( 2^k \).

We can see that the orthonormal discrete wavelet transform as defined in (2.66) is a limit case for \( f(x) \in L^2(\mathbb{R}) \), that is, without approximation space and with an infinite number of detail spaces \( W_k \). We can write this as

\[
L^2(\mathbb{R}) = \bigoplus_{k \in \mathbb{Z}} W_k
\]

(2.85)

Normally, all (2.84) should be calculated by inner products involving continuous-time integration. Luckily, by using properties of the multiresolution analysis, this won’t be necessary. Complete calculation can be done by using a discrete-time algorithm based on the appropriate filter bank structure. This algorithm is called Mallat’s algorithm [29].

By combining the two-scale equation (2.76) with equation (2.82) and by taking \( h_0[n] = g_0[-n] \) we obtain

\[
a^{(j+1)}[n] = \sum_{k=-\infty}^{\infty} h_0[2n - k] a^{(j)}[k]
\]

(2.86)

By comparing this equation with (2.40) we see that the discrete approximation coefficients \( a^{(j)}[n] \) can be obtained by filtering \( a^{(j-1)}[n] \) with \( h_0[n] \) followed by downsampling by two. This is the beauty of Mallat’s algorithm: higher level (smaller resolution) coefficients are obtained from the previous level coefficients by an easy filter bank operation.

By using (2.79) and (2.82) and taking \( h_1[n] = g_1[-n] \), we can find a similar formula for coefficients of the projection onto the detail space \( W_j \):

\[
d^{(j+1)}[n] = \sum_{k=-\infty}^{\infty} h_1[2n - k] a^{(j)}[k]
\]

(2.87)
This operation too can be implemented by a simple cascade of downsampling by two and filtering, but this time with \( h_1[n] \). The equivalent structure that implements this algorithm is shown in fig. 2.10. It is called an analysis octave-band filter bank.

\[
\begin{align*}
\text{Figure 2.10: 3-stage analysis octave-band filter bank}
\end{align*}
\]

By taking the inner product of both sides of (2.83) with \( \phi_{j-1,n} \) one can show that it is possible to obtain the approximation coefficients \( a^{(j-1)}[n] \) from \( a^{(j)}[n] \) and \( d^{(j)}[n] \) by

\[
a^{(j-1)}[n] = \sum_{k=-\infty}^{\infty} \left( g_0[n - 2k]a^{(j)}[k] + g_1[n - 2k]d^{(j)}[k] \right) \tag{2.88}
\]

Comparison with (2.42) shows that this formula can be implemented a filter bank that sums two signals: a signal obtained by upsampling \( a^{(j)}[k] \) by two followed by filtering with \( g_0[n] \) and a signal obtained by upsampling \( d^{(j)}[k] \) by two followed by filtering with \( g_1[n] \). By iteration of (2.88), we can reconstruct the original \( a^{(0)}[n] \). The whole mechanism is implemented in practice by the synthesis octave-band filter bank of fig. 2.11.

\[
\begin{align*}
\text{Figure 2.11: 3-stage synthesis octave-band filter bank}
\end{align*}
\]

I will now give some interesting properties of orthonormal wavelets and scaling functions
The orthonormality of the family \( \{ \phi(x - n) \mid n \in \mathbb{Z} \} \) is equivalent to the following in the Fourier domain
\[
\sum_{n=-\infty}^{\infty} |\Phi(e^{j(\omega+2n\pi)})|^2 = 1 \tag{2.89}
\]
Expressing the two-scale equation (2.76) in the Fourier domain, with \( \Phi \) and \( G_0 \) the DTFT of \( \phi \) and \( g_0 \), leads to
\[
\Phi(e^{j\omega}) = \frac{1}{\sqrt{2}} G_0(e^{j\omega/2}) \Phi(e^{j\omega/2}) \tag{2.90}
\]
By combining (2.89) and (2.90), the following equation can be found
\[
|G_0(e^{j\omega})|^2 + |G_0(e^{j(\omega+\pi)})|^2 = 2 \tag{2.91}
\]
This equation learns us that \( \{ G_0(e^{j\omega}), G_0(e^{j(\omega+\pi)}) \} \) is a power complementary filter pair.

Let’s recall now the relationships between the filter coefficients \( h_0[n], h_1[n], g_0[n] \) and \( g_1[n] \) that were established in the above multiresolution analysis.
\[
g_0[n] = h_0[-n] \tag{2.92}
\]
\[
g_1[n] = h_1[-n] \tag{2.93}
\]
\[
g_1[n] = (-1)^n g_0[-n+1] \tag{2.94}
\]
By using properties (2.32)-(2.34) we can write \( G_0, G_1, H_1 \) in function of \( H_0 \) in the Z-transform domain:
\[
G_0(z) = H_0(z^{-1}) \tag{2.95}
\]
\[
G_1(z) = \sum_{n=-\infty}^{\infty} (-1)^n g_0[-n+1] z^{-n}
\]
\[
= \sum_{n=-\infty}^{\infty} (-1)^n h_0[n-1] z^{-n}
\]
\[
= -z^{-1} \sum_{k=-\infty}^{\infty} (-1)^k h_0[k] z^{-k} \quad \text{with } k = n - 1
\]
\[
= -z^{-1} H_0(-z) \tag{2.96}
\]
\[
H_1(z) = G_1(z^{-1}) = -z H_0(-z^{-1}) \tag{2.97}
\]
We now define a new two-channel filter bank with filters

\[
\begin{align*}
\tilde{H}_0(z) &= G_0(z) = H_0(z^{-1}) \\
\tilde{H}_1(z) &= G_1(z) = -z^{-1}H_0(-z) \\
\tilde{G}_0(z) &= H_0(z) \\
\tilde{G}_1(z) &= H_1(z) = -zH_0(-z) 
\end{align*}
\]  

(2.98) \hspace{2cm} (2.99) \hspace{2cm} (2.100) \hspace{2cm} (2.101)

This filter bank satisfies the aliasing condition (2.53). \( \tilde{H}_0, \tilde{H}_1, \tilde{G}_0, \tilde{G}_1 \) are CQF pairs (2.60) and from (2.98) and (2.62) it follows that \( \tilde{H}_0(z) \) is a power complementary filter pair. This means that \( \tilde{H}_0, \tilde{H}_1, \tilde{G}_0, \tilde{G}_1 \) comprises a perfect reconstruction filter bank. This remark completes the link between filter bank theory and wavelets.

### 2.10 Biorthogonal Wavelets

Biorthogonal wavelets are a generalization of orthonormal wavelets. Multiresolution analysis can be extended and characterized by two scaling functions: \( \phi(x) \) and its dual \( \tilde{\phi}(x) \), and two mother wavelets: \( \psi(x) \) and its dual \( \tilde{\psi}(x) \). These functions are such that for two hierarchies of approximation subspaces

\[
\begin{align*}
\cdots & \subset V_2 \subset V_1 \subset V_0 \subset V_{-1} \subset V_{-2} \subset \cdots \\
\cdots & \subset \tilde{V}_2 \subset \tilde{V}_1 \subset \tilde{V}_0 \subset \tilde{V}_{-1} \subset \tilde{V}_{-2} \subset \cdots
\end{align*}
\]  

(2.102) \hspace{2cm} (2.103)

the \( V_j \) are spanned by \( \{ \phi_{j,n} = 2^{-j}\phi(2^{-j}x-n) \mid n \in \mathbb{Z} \} \) and the \( \tilde{V}_j \) by \( \{ \tilde{\phi}_{j,n} = 2^{-j}\tilde{\phi}(2^{-j}x-n) \mid n \in \mathbb{Z} \} \). The \( \{ \psi_{j,n} = 2^{-j}\psi(2^{-j}x-n) \mid n \in \mathbb{Z} \} \) form a basis for \( W_j \) and the \( \{ \tilde{\psi}_{j,n} = 2^{-j}\tilde{\psi}(2^{-j}x-n) \mid n \in \mathbb{Z} \} \) form a basis for \( \tilde{W}_j \) such that \( V_{j-1} = \tilde{V}_j \oplus W_j \) and \( V_{j-1} = V_j \oplus \tilde{W}_j \).

Any function \( f(x) \in V_0 \) can be expanded as

\[
\begin{align*}
f(x) &= \sum_{n=-\infty}^{\infty} \tilde{a}^{(j)}[n] \tilde{\phi}_{j,n}(x) + \sum_{k=1}^{j} \sum_{n=-\infty}^{\infty} d^{(k)}[n] \psi_{k,n}(n) \\
&= \sum_{n=-\infty}^{\infty} a^{(j)}[n] \phi_{j,n}(x) + \sum_{k=1}^{j} \sum_{n=-\infty}^{\infty} \tilde{d}^{(k)}[n] \tilde{\psi}_{k,n}(n)
\end{align*}
\]  

(2.104) \hspace{2cm} (2.105)
where

\[
\tilde{a}^{(j)}[n] = \langle f(x), \tilde{\phi}_{j,n}(x) \rangle \quad \text{and} \quad \tilde{d}^{(k)}[n] = \langle f(x), \tilde{\psi}_{k,n}(x) \rangle
\]  

\[
a^{(j)}[n] = \langle f(x), \phi_{j,n}(x) \rangle \quad \text{and} \quad d^{(k)}[n] = \langle f(x), \psi_{k,n}(x) \rangle
\]  

Discrete iterated filters are introduced by the two-scale equations

\[
\phi(x) = \sqrt{2} \sum_{n=-\infty}^{\infty} c_0[n] \phi(2x - n)
\]  

\[
\tilde{\phi}(x) = \sqrt{2} \sum_{n=-\infty}^{\infty} \tilde{c}_0[n] \tilde{\phi}(2x - n)
\]  

If we define

\[
h_0[n] = c_0[-n]
\]  

\[
g_0[n] = \tilde{c}_0[-n]
\]  

\[
h_1 = (-1)^{n+1} g_0[-n + 1]
\]  

\[
g_1 = (-1)^{n+1} h_0[-n + 1]
\]

then Mallat’s algorithm can be to find the approximation and detail coefficients:

\[
a^{(j)}[n] = \sum_{k=-\infty}^{\infty} h_0[2n - k] a^{(j-1)}[k]
\]  

\[
d^{(j)}[n] = \sum_{k=-\infty}^{\infty} h_1[2n - k] a^{(j-1)}[k]
\]

and to reconstruct the signal again

\[
a^{(j-1)}[n] = \sum_{k=-\infty}^{\infty} (g_0[n - 2k] a^{(j)}[k] + g_1[n - 2k] d^{(j)}[k])
\]

This algorithm gives rise to an octave-band filter bank implementation for the biorthogonal DWT, completely equivalent to the octave-band filter bank implementation for the orthonormal DWT.
2.11 Haar Wavelet

The simplest wavelet is the Haar wavelet. Here $\psi$ equals

$$
\psi(x) = \begin{cases} 
1, & 0 \leq x < 0.5 \\
-1, & 0.5 \leq x < 1 \\
0, & \text{otherwise}
\end{cases}
$$

(2.117)

and $\phi$ equals

$$
\phi(x) = \begin{cases} 
1, & 0 \leq x \leq 1 \\
0, & \text{otherwise}
\end{cases}
$$

(2.118)

and these functions are shown together with their DTFT in fig. 2.12

![Figure 2.12](image)

(a) Haar scaling function $\phi(x)$ (b) Fourier transform $\Phi(e^{j \omega})$ (c) Haar mother wavelet $\psi(x)$ (d) Fourier transform $\Psi(e^{j \omega})$

It is easy to show that this $\phi(x)$ and $\psi(x)$ define a multiresolution analysis for the subspaces $V_j$ of piecewise continuous functions at intervals $[k2^j, (k+1)2^j]$. They are orthonormal and satisfy both a two-scale equation:

$$
\phi(x) = \phi(2x) + \phi(2x - 1) = \sqrt{2} \left( \frac{1}{\sqrt{2}} \phi(2x) + \frac{1}{\sqrt{2}} \phi(2x - 1) \right)
$$

(2.119)

$$
\psi(x) = \phi(2x) - \phi(2x - 1) = \sqrt{2} \left( \frac{1}{\sqrt{2}} \phi(2x) - \frac{1}{\sqrt{2}} \phi(2x - 1) \right)
$$

(2.120)

The filters defined by the multiresolution analysis become

$$
g_0[n] = \left[ \frac{1}{\sqrt{2}} \right] \quad g_1[n] = \left[ \frac{1}{\sqrt{2}} \right] \quad h_0[n] = \left[ \frac{-1}{\sqrt{2}} \right] \quad h_1[n] = \left[ \frac{1}{\sqrt{2}} \right]
$$

(2.121)
2.12 Choice of Wavelets

Before looking at some further examples of wavelets, we will discuss which properties 'good' wavelets for image compression should possess.

Filter Length

A first condition is that the wavelets should be compactly supported: there exists an $L \in \mathbb{R}$ such that $\psi(x) = 0$ for $|x| > L$. This condition follows from the fact that the two-scale equation should give rise to finite-length FIR filters. It should be noted that long wavelets result in long filters and short wavelets result in short filters.

Knowing that wavelets are basis functions, they should be able to approximate image signals well. Image signals are typically smoothly varying signals with occasional discontinuities at edges. To approximate the signal near this discontinuities, it is advantageous to have basis functions that are well localized in space, that is, short basis functions. This way, the value at the edge is determined by just a few shifted basis functions and this allows for fast convergence of the wavelet expansion at the edge. We say that short wavelets have good space resolution.

On the other hand, to approximate the smooth parts of the signal, it is advantageous to have basis functions that are well localized in frequency (fast decaying), that is, long basis functions. This way, the signal values are determined by just a few scaled basis functions and this allows for fast convergence of the wavelet expansion where the signal is smooth. We say that long wavelets have good frequency resolution.

This discussion learns that good frequency resolution and space resolution are not compatible, a fundamental property related to the Heisenberg uncertainty principle. It is interesting to remark that the usual image representation is a limit case without frequency resolution and the DFT transform a limit case without space resolution. Good wavelets should find a compromise between long and short basis functions. The Haar wavelet, which is the shortest existing wavelet, will not yield good results in image compression.

In the mathematical wavelet theory, smoothness is characterized by a rigorously defined value called regularity. Rioul [39] experimentally verified the influence of regularity and concluded that some regularity is desired and that higher regularity further improves compression, but not much. A good compromise is found for filter lengths in the order of 8 - 10.

Regularity is related (but not equivalent!) to the wavelet’s number of vanishing moments.
We say that a wavelet $\psi_{k,n}(x)$ has $N$ vanishing moments when
\[ \int_{-\infty}^{\infty} \psi_{k,n}(x)x^Ldx = 0 \quad \forall L \in [0, N] \quad (2.122) \]

If a wavelet has $N$ vanishing moments, then polynomials of degree less than $N$ can be represented as a linear combination of translates of the same resolution scaling function $\psi_{k,n}(x)$. This means that where the signal can be approximated very well locally by its Taylor polynomial of order $N - 1$, many wavelet coefficients $\langle f(x), \psi_{k,n}(x) \rangle$ will be near zero and this leads to efficient coding.

**Orthonormality**

The DWT should be an orthonormal DWT, so that the wavelet basis functions (2.77) form an orthonormal basis for $L^2(\mathbb{R})$. Orthonormal transforms satisfy an energy conservation theorem, analogous to Parseval’s theorem in Fourier analysis, that can be written in terms of the multiresolution analysis approximation and detail coefficients as
\[ \sum_n (a^{(j-1)}[n])^2 = \sum_n (a^{(j)}[n])^2 + \sum_n (d^{(j)}[n])^2 \quad (2.123) \]

This energy conservation property is convenient for compression system design since the mean squared distortion introduced by quantizing the transformed coefficients equals the mean squared distortion in the reconstructed signal. Hence, the energy conservation property simplifies designing the coder since the quantizer design can be carried out completely in the transform domain and optimal bit allocation formulas can be used. [48]

**Linear Phase**

In section 2.5 I explained that filter operations are defined by linear convolutions. For an input signal $x[n]$ of length $N$ and a filter $h[n]$ of length $M$, this results in an output signal $y[n]$ of length $N + M - 1$. This signal extension cannot be allowed in compression systems. We saw that the linear convolution can be replaced by a circular convolution, resulting in an output $y_c[n]$ of length $N$, only when the filter $h[n]$ is linear phase.

This means that the analysis filters $h_0[n], h_1[n]$ and synthesis filters $g_0[n], g_1[n]$ of the octave-band filter bank that implements the discrete wavelet transform should be symmetrical $h[n] = h[M - n]$ or antisymmetrical $h[n] = -h[M - n]$. Consequently, the mother wavelet $\psi(x)$ and scaling $\phi(x)$ of the multiresolution analysis that gives rise to these filters should be symmetric or antisymmetric too.
Another reason to use linear phase filters, well known in image processing, is the phase distortion introduced by non-linear phase filters. This phase distortion causes edges to smear out and reduces sharpness of the image. One could argue that because of the perfect reconstruction filters this doesn’t matter, but because the image is typically quantized after the analysis filter pass, the reconstruction will not be perfect and linear phase is important. A final argument in favor of linear phase filters is that the human eye is less sensitive to symmetric distortion around edges than it is to asymmetric distortion.

**Orthonormality vs. Linear phase**

In the previous paragraphs I explained that the ideal wavelets for image compression purposes should be compactly supported, smooth, (anti)symmetric and form an orthonormal basis for $L^2(\mathbb{R})$. However, Daubechies [9] has shown that the only compactly-supported symmetrical or antisymmetrical wavelets that comprise an orthonormal basis for $L^2(\mathbb{R})$ are the Haar wavelets. In practice, this problem is overcome by using biorthogonal wavelets that are close to orthonormal wavelets, so that (2.123) is closely approximated.

### 2.13 Daubechies wavelets

The Daubechies wavelets provided a breakthrough in wavelet theory in 1988 [8]. They are a family of orthonormal, compactly supported wavelets that have the maximum regularity for a given length of the resulting filters. The mathematical details concerning their construction are outside the scope of this thesis.

It is however interesting to remark that a closed form expression for these wavelets does not exist. The mathematical theory will only give us the corresponding filter coefficients. By iterating the so-called cascade algorithm [9], better and better approximations for these wavelets can be found.

Fig. 2.13 shows the Daubechies scaling functions and mother wavelets, obtained by the cascade algorithm, and their Fourier domain representation for different filter lengths. This figure confirms what was said in the previous section. For longer filters, we get smoother wavelets, the scaling function’s Fourier transform becomes a better low-pass filter and the mother wavelet’s Fourier transform a better band-pass filter. The wavelets do not have a symmetry-axis, so the filters are not linear phase and are not well suited for image compression applications.
Chapter 2. Image Transforms

Figure 2.13: Daubechies scaling functions $\phi(x)$ and mother wavelets $\psi(x)$ in space domain and in Fourier domain $\Phi(e^{j\omega})$ and $\Psi(e^{j\omega})$ for different filter lengths

4-taps filter: (a) $\phi(x)$ (b) $\psi(x)$ (c) $\Phi(e^{j\omega})$ (d) $\Phi(e^{j\omega})$

6-taps filter: (e) $\phi(x)$ (f) $\psi(x)$ (g) $\Phi(e^{j\omega})$ (h) $\Phi(e^{j\omega})$

8-taps filter: (i) $\phi(x)$ (j) $\psi(x)$ (k) $\Phi(e^{j\omega})$ (l) $\Phi(e^{j\omega})$

12-taps filter: (m) $\phi(x)$ (n) $\psi(x)$ (o) $\Phi(e^{j\omega})$ (p) $\Phi(e^{j\omega})$
2.14 CDF Biorthogonal Wavelets

The Cohen-Daubechies-Fauveau (CDF) wavelets [6] are a class of biorthogonal wavelets that are close to orthonormal wavelets. The most popular wavelet for image compression is the 9/7 CDF wavelet. This wavelet has been adopted by the JPEG2000 image compression standard. The analysis filter coefficients are given in table 2.1. The synthesis filter coefficients follow immediately by $g_0[n] = (-1)^n h_1[1]$ and $g_1[n] = (-1)^n h_0[1]$.

<table>
<thead>
<tr>
<th>$i$</th>
<th>low-pass filter $h_0[n]$</th>
<th>high-pass filter $h_1[n]$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.6029490182363579</td>
<td>1.115087052456994</td>
</tr>
<tr>
<td>±1</td>
<td>0.2668641184428723</td>
<td>-0.5912717631142470</td>
</tr>
<tr>
<td>±2</td>
<td>-0.07822326652898785</td>
<td>-0.0575435262284957</td>
</tr>
<tr>
<td>±3</td>
<td>-0.01686411844287495</td>
<td>0.09127176311424948</td>
</tr>
<tr>
<td>±4</td>
<td>0.02674875741080976</td>
<td></td>
</tr>
</tbody>
</table>

Table 2.1: CDF 9/7 Analysis Filter Coefficients

Fig. 2.14 shows the 9/7 CDF scaling functions and mother wavelets and their Fourier domain representation. We can see that the synthesis scaling function and wavelet are much more regular than the analysis scaling function and wavelet. This is no coincidence. We explained in section 2.1 that in biorthogonal expansions two complete bases $\psi(x)$ and $\tilde{\psi}(x)$ exist. The two formulas

$$f(x) = \sum_{k,n} \langle f(x), \psi_{k,n}(x) \rangle \tilde{\psi}(x)$$ \hspace{1cm} (2.124)
$$= \sum_{k,n} \langle f(x), \tilde{\psi}_{k,n}(x) \rangle \psi(x)$$ \hspace{1cm} (2.125)

are both equally valid. In filter terms, this means that synthesis and analysis filters can be interchanged. However, when $\tilde{\psi}(x)$ is much more regular than $\psi(x)$ then both configurations do not yield the same results in compression. Equation 2.124, corresponding to more regular synthesis filters "builds" $f(x)$ from smoother basis functions and will yield better results than 2.125 in image compression.
Figure 2.14: 9/7 CDF scaling functions $\phi(x)$ and mother wavelets $\psi(x)$ in space domain and in fourier domain $\Phi(e^{j\omega})$ and $\Psi(e^{j\omega})$

Analysis filter: (a) $\phi(x)$ (b) $\psi(x)$ (c) $\Phi(e^{j\omega})$ (d) $\Phi(e^{j\omega})$

Synthesis filter: (e) $\phi(x)$ (f) $\psi(x)$ (g) $\Phi(e^{j\omega})$ (h) $\Phi(e^{j\omega})$
2.15 Subband Coding of Images

In the previous section I explained that a continuous-time one-dimensional signal \( f(x) \) can be projected onto an approximation space \( V_0 \) to obtain a discrete approximation \( a^{(0)}[n] \). For a discrete-time signal \( x[n] \), we will simply define \( a^{(0)}[n] = x[n] \).

The next-resolution discrete approximation \( a^{(1)}[n] \) and detail coefficients \( a^{(1)}[n] \) could then be obtained by a two-channel analysis filter bank scheme with filters \( h_0[n] \) and \( h_1[n] \) and a downsampler by two in each channel. Because of this, we speak often about low-pass signals instead of discrete approximations and high-pass signals instead of detail coefficients.

The generalization of this one-dimensional subband decomposition to two-dimension image signals \( x[m, n] \) is straight-forward. We will only discuss the separable case where the low-pass signal and high-pass signals can be found by a cascade of two one-dimensional operations.

For a \( 2^n \times 2^n \) size monochromatic image this consists in considering each of the \( 2^n \) rows as a \( 2^n \)-length 1D-signal and pass them through the two-channel analysis filter bank with filters \( h_0[n] \) and \( h_1[n] \). This will result in \( 2^n 2^{n-1} \)-length low-pass signals and \( 2^n 2^{n-1} \)-length high-pass signals\(^2\).

We now create \( 2^n \) new \( 2^n \)-length row vectors with the low-pass signal at the first \( 2^n-1 \) positions and the high-pass signal at the last \( 2^n-1 \) positions. Together these row vectors comprise a new \( 2^n \times 2^n \) size image.

We can now repeat the filtering procedure described above for the columns of this new image. This results in a new image that consists of 4 subbands. We call the subband that has passed 2 low-pass filters the low-pass approximation \( (LL) \), the subband that was first low-passed, then high-passed the horizontal details \( (HL) \), the subband that was first high-passed then low-passed the vertical details \( (LH) \) and the subband that was 2 times high-passed the diagonal details \( (HH) \).

Figure 2.15 (a) shows the positioning of these subbands in the image while Fig. 2.15 (b) shows an example that makes it clear why these subbands are called the way they are.

We can now apply the same filtering scheme to the low-pass approximation. This two-scale subband decomposition is shown for the Lena image in fig. 2.16. For a \( 2^n \times 2^n \) size image, this procedure can be repeated maximum \( n \) times.

\(^2\)Only for linear phase FIR filters. See section 2.5
Figure 2.15: (a) Positioning of the subbands in a one-scale decomposed image (b) Example of a one-scale decomposed image

Figure 2.16: (a) Positioning of the subbands in a 2-scale decomposed image. (b) 2-scale decomposed Lena image. The high-pass signals are enhanced.
Chapter 3

Scalar Quantization and Entropy Coding

The transform coding paradigm for image compression consists of three basic steps: image transform, scalar quantization and entropy coding. In this chapter, a brief overview of scalar quantization and entropy coding is presented.

3.1 Scalar Quantization

**Definition** An L-point scalar quantizer $Q$ is a mapping $Q : \mathbb{R} \rightarrow C$ where $\mathbb{R}$ is the set of real values and $C = \{y_1, y_2, \ldots, y_L\} \subset \mathbb{R}$ is the output set or codebook.

Electrical engineers are familiar with the principles of scalar quantization from the theory of analog-to-digital conversion, where a mapping from a real value from a continuous subset of $\mathbb{R}$ to natural value from a discrete subset of $\mathbb{N}$ is defined. A classic example is the mapping of a real-valued voltage in the interval $0 - 5$ Volt to a 10-bit natural value in the range of $0 - 1023$.

Scalar quantization is an irreversible step and cannot be used in lossless image compression. Because there are fewer possible outputs than possible inputs, fewer bits are needed to code the outputs and this leads to compression.

When a large number of input values is mapped onto a small number of output values we speak about coarse quantization. When the number of output values is high we speak about fine quantization.
Recall that we represent images \( x[m, n] \) by \( N \times N \)-sized matrices where the elements are values between 0 and \( 2^8 - 1 \) that correspond to the pixels' grayscale values. A first, primitive idea to use scalar quantization for image compression could be to map the \( 2^8 \) possible grayscale values to a smaller set of, for example, \( 2^4 \) grayscale values. It is evident that this method is not at all a good compression method.

A much better method would be to precede the scalar quantization by an image transform. In fact, as explained in section 2.2, one of the principal reasons of existence of image transforms is exactly that they lead to much more efficient scalar quantization. In the transform domain all coefficients do not have the same importance for the visual quality of the image and coefficients that are not important can be quantized more coarsely (mapped to a smaller set of output values) than coefficients that have great importance for the image quality. An important example of how to implement scalar quantization can be found in the JPEG standard and is explained in detail in the next chapter.

### 3.2 Entropy Coding

Entropy coding is lossless image coding and removes redundancy from the representation of data. A good image compression system will always have a transform and quantizer design that maximizes the compression obtained by the entropy coding step.

To define "entropy", start from a data sequence of \( N \) symbols \( S_1, \ldots, S_N \). Let \( M \) be the number of distinct symbols occurring in this data sequence: \( M \leq N \). The entropy \( H \) is defined as the fundamental lower bound of the average number of bits needed to encode each symbol in the sequence and satisfies [44]

\[
H = \sum_{i=1}^{N} p(S_i) \log_2 \frac{1}{p(S_i)} \quad (3.1)
\]

with \( p(S_i) \) the probability of occurrence of symbol \( S_i \) in the data sequence. The quality of an entropy coder is determined by how close it approaches the entropy.

### 3.3 Variable-Rate Entropy Coding

Variable-rate entropy coding is a collective term for all entropy coding schemes that use variable bit lengths for encoding different symbols. Compression is obtained by represent-
ing frequently occurring symbols by short bit sequences and rare symbols by longer bit sequences. The two most popular variable-rate entropy coding techniques are Huffman coding and arithmetic coding.

### 3.3.1 Huffman Coding

Huffman coding is a simple, effective and popular entropy coding scheme. We will illustrate it with an easy example. Suppose that the data sequence to encode consist of 6 symbols $S_i$. Table 3.1 shows the probability of occurrence of these symbols and the Huffman code. The best way to explain the construction of Huffman code is by a binary tree (fig. 3.1). Start with a list of the probabilities of the symbols. Then, take the two least probable symbols, assign them a 0 and a 1 resp. and take them together to form a new symbol. The new symbol has a probability which is the sum of the probabilities of the merged symbols. The new list of symbols is now shorter by one. Iterate until only one symbol is left. The codewords can now be read off along the branches of the binary tree. For this example the expectation value of the number of bits per symbol is 2.35 bits/symbol, very close the the entropy of 2.28 bits/symbol. More on Huffman coding.

![Figure 3.1: Construction of Huffman code by a binary tree](image-url)

<table>
<thead>
<tr>
<th>$S_i$</th>
<th>$p(S_i)$</th>
<th>$b_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0.40</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>0.20</td>
<td>100</td>
</tr>
<tr>
<td>C</td>
<td>0.15</td>
<td>101</td>
</tr>
<tr>
<td>D</td>
<td>0.10</td>
<td>110</td>
</tr>
<tr>
<td>E</td>
<td>0.10</td>
<td>1110</td>
</tr>
<tr>
<td>F</td>
<td>0.05</td>
<td>1111</td>
</tr>
</tbody>
</table>

Table 3.1: Example of Huffman entropy coding
can be found in [36] or [17].

### 3.3.2 Arithmetic Coding

In this scheme, symbols are ordered on an interval \([0, 1]\) such that the probability of a symbol is equal to the length of the subinterval to which it is assigned. A sequence of symbols is then encoded together by assignment of a binary floating point number. This floating point is found by iterative splitting of the subintervals.

This is illustrated in fig. 3.2 for the example of table 3.1 and a simple sequence \(BC\). Contrary to Huffman coding, arithmetic coding does not have to use an integral number of bits to encode a symbol and leads often to greater efficiency. More on arithmetic coding can be found in [36] or [17].

### 3.4 Run-Length Encoding

Run-length encoding (RLE) is an entropy coding technique for data sequences containing long stretches of the same symbol. Consider for example the following binary sequence:

\[
000000110000100000000
\]

A RLE solution could encode this by

\[(0, 6); (1, 2); (1, 1); (0, 8)\]
This RLE technique would completely fail for a data sequence with many alternating zeros and ones. Variations of RLE solve this problem.

3.5 DPCM

Differential Pulse Code Modulation (DPCM) is not really an entropy coding technique on its own, but rather a technique to use in combination with classical entropy coders. In DPCM, the actual symbol value is predicted from the values of a subset of previously encoded symbols.

This prediction is then used to compute a difference measure with the actual symbol and this difference is encoded instead of the actual symbol. For sources with large correlation between subsequent symbols, the differences will typically be a lot smaller than actual symbols and this causes additional compression after entropy coding.

An interesting DPCM implementation is discussed in [1]. It is explained later in this work.
Chapter 4

JPEG and JPEG2000

4.1 JPEG

The JPEG compression standard has emerged as the most popular universal image compression standard. In this section I explain how the concepts from the previous chapters can be put together to form the standard. Wallace [49] is a reference article that presents an overview of the basic features of this standard, while Pennebaker and Mitchell [35] provides a detailed discussion.

At the time of its development the following criteria were put first [49]:

- State-of-the-art compression for a wide range of compression rates and image qualities. The encoder should be parameterizable, so that the user can set the desired compression/quality trade-off.
- Applicable to practically any kind of continuous-tone digital image.
- Tractable computational complexity.

The standard proposes four operation modes and one or more codecs for each operation mode. The most widespread codec is the baseline sequential codec. In what follows I simply refer to this codec as ”JPEG”.

4.1.1 Image Transform

In JPEG an RGB image is transformed into the YCbCr color space. This yields three monochromatic images that are processed apart. They are divided in blocks of $8 \times 8$
pixels and these blocks are also processed apart. Each block is transformed into the spatial frequency domain by means of a two-dimensional discrete cosine transform. This yields $8 \times 8$ blocks of transform coefficients $X(u, v)$.

The advantages of this transform were discussed in detail in section 2.4. I briefly recall that the transformed blocks contain low spatial frequency components in the upper left corner while higher frequency components tend toward the lower right corner. Because of the local spatial correlation typically found in natural images, most $8 \times 8$ blocks will contain little variation. This means that the DCT concentrates most signal energy in the low spatial frequency components.

### 4.1.2 Scalar Quantization

The theory of the contrast sensitivity function, section 1.3, tells us that the human visual system is much more sensitive at low spatial frequencies than at high spatial frequencies. This means that, in order to preserve a desired image quality, the high frequency components do not need to be represented with the same precision as the low frequency components.

The scalar quantization technique in JPEG uses a predefined quantization table $Q(u, v)$. A frequently used choice for this quantization table, based on psychovisual experiments is shown in fig. 4.1 (c).

Notice that coefficients corresponding to low frequencies (upper left corner) have small quantization values compared to high quantization values for high frequency components (lower right corner). The quantization table is not symmetric. This is because the human eye is not equally sensitive in horizontal and vertical directions.

A quantization matrix $P(u, v)$ is defined from this quantization table $Q(u, v)$:

$$P(u, v) = \text{Int Round} \left( Q(u, v) \frac{50}{q} \right) \quad (4.1)$$

where $q$ (quality) is a user-defined number between 1 and 100 that determines the quality and bit-rate by which the image will be encoded. If $q = 50$ then $P(u, v) = Q(u, v)$ holds.

The quantization is now defined as a division of each DCT coefficient $X(u, v)$ by its
corresponding quantizer step size $P(u, v)$, followed by rounding to the nearest integer

$$X_Q(u, v) = \text{Int Round} \left( \frac{X(u, v)}{P(u, v)} \right) \quad (4.2)$$

For lower quality $q$ we get higher values in the quantization matrix $P(u, v)$ and hence a more coarsely quantized matrix $X_Q(u, v)$. This means that $X_Q(u, v)$ contains lower coefficients (and more zeros) and this will result in a lower bit-rate after entropy coding.

Reconstruction is straightforward. The numbers in $X_Q(u, v)$ are multiplied by the corresponding number from $P(u, v)$ and an inverse DCT is applied. Fig. 4.1 (a)-(f) illustrates the process of DCT, quantization and reconstruction.
4.1.3 Entropy Coding

At the entropy coding stage all blocks $X_Q(u,v)$ are received. The DC coefficients $X_Q(1,1)$ are treated separately from the 63 AC coefficients. A DC coefficient is equal to the average value of the 64 image samples in a block. Because there is usually strong correlation between the DC coefficients of adjacent $8 \times 8$ blocks, the quantized DC coefficient is encoded as the difference from the DC term of the previous block.

All of the quantized AC coefficients are ordered into a zig-zag sequence. This ordering helps to facilitate entropy coding by placing low-frequency coefficients (which are more likely to be nonzero) before high-frequency coefficients. These concepts are illustrated in fig. 4.2.

![Figure 4.2: Ordering of DCT coefficients before entropy coding](image)

The AC components are entropy coded by a variable-length entropy coder. The coder features run-length coding to efficiently encode the frequently occurring long runs of zeros and Huffman or arithmetic coding.

4.1.4 Performance

The compression in JPEG comes from the large amount of zeros and small numbers in the blocks $X_Q(u,v)$. I explained that this large amount of zeros and small numbers is due to two factors:

- Smooth blocks resulting in low high-frequency DCT coefficients
• Lesser sensitivity of the human eye at high DCT frequencies allowing coarser quantization

From this it is easy to point out the weakness of JPEG coding. When there is an edge in the $8 \times 8$ block under consideration, some of the high frequency components do contain significant signal energy, but will be quantized too coarsely. This will spread out errors over the block and causes block boundaries to become visible at relatively modest compression ratios: blocking artifacts.

To illustrate the performance of JPEG I compressed the Lena image at different bit-rates. The JPEG compressed images were calculated by the Matlab implementation of JPEG. Fig. 4.3 shows the PSNR of a JPEG compressed Lena in function of the bit-rate after compression. Fig. 4.4 shows a zoom in on Lena’s face for Lena images compressed at different bit-rates. We can see that stronger blocking artifacts occur for lower bit-rates.

![Figure 4.3: PSNR after JPEG Compression for Lena](image-url)
Figure 4.4: Zoom of Lena’s face after JPEG compression (a) at 0.20 bpp (b) at 0.30 bpp (c) at 0.40 bpp (d) at 0.50 bpp
4.2 JPEG2000

4.2.1 Motivation

JPEG2000 is the recently proposed successor of JPEG [5]. Its goal is not only to improve on compression performance, but also to add more flexibility. In fact, at good quality bit rates for color images of 0.5-1.0 bpp, the performance of JPEG2000 is typically only 20% better than JPEG. At low bit rates however, it does not suffer from blocking artifacts as JPEG and yields much improved results.

Flexibility is added by embedded coding. This means that the bits in the bit stream are generated in order of importance. An embedded coder can terminate the encoding at any target bit rate or target quality, contrary to JPEG where only an approximative quality factor could be specified.

The flexibility of embedded coding can further be illustrated by an example. Suppose we have a losslessly encoded image at 100000 bits, then we can send a low quality version of 5000 bits to somebody by sending the first 5000 bits. If this person later wants a better quality version of say 10000 bits, it suffices to send bits 5001-10000. With a non-embedded coder like JPEG this is not possible: there is no relation between the bits in a low and high quality version of the same image, and all 10000 bits should be transmitted.

JPEG2000 also allows for region-of-interest (ROI) coding: defining parts of an image that should be coded with more bits and less distortion than the rest of the image and for random codestream access: applying processing operations as filtering, rotation, feature extraction to ROI’s.

4.2.2 Structure of JPEG2000

The first operation in JPEG2000 is a color transform into the YCbCr space. Then, the image can be split into large blocks, called tiles, that are processed apart. Tiling always diminishes the quality of the compression and should only be done when memory is limited.

Next each tile is wavelet transformed by the CDF 9/7 wavelet until an arbitrary depth. The resulting wavelet transformed image is scalar quantized and all quantized subbands are processed by Embedded Block Coding with Optimized Truncation of the embedded bit-streams (EBCOT) [47].
The details of this highly complicated system fall outside the scope of this thesis. The general idea is to split the subbands into code-blocks. For each code-block EBCOT generates a separate embedded bit stream. A rate distortion optimization finds how many bits in each code-block’s embedded bit stream should be kept to result in the highest quality image at a given bit rate. The coded data of each code-block is distributed across one or more layers, ordered by quality, in the final bit stream. Adaptive arithmetic coding is incorporated in this system.

### 4.2.3 Performance

To evaluate the performance of JPEG2000 I made use of free JASPER codec of M.D. Adams\(^1\). This is is one of the only two free codecs that exist. The other is the JJ2OOO java implementation\(^2\).

Fig. 4.5 shows the PSNR of a JPEG2000 compressed Lena in function of the bit-rate after compression and compares to JPEG.

![Figure 4.5: PSNR after JPEG2000 and JPEG Compression for Lena](image)

Fig. 4.6 shows a zoom in on Lena’s face for Lena images compressed at different bit-rates. We can see that even at a very low bit-rate of 0.10 bpp (compression rate 80:1) the quality

---

\(^1\)http://www.ece.uvic.ca/~mdadams/jasper/

\(^2\)http://jj2000.epfl.ch/
is acceptable and at an extremely low bit-rate of 0.04 bpp the face is still recognizable. Comparison of the 0.20 bpp image to the 0.20 bpp JPEG image in fig. 4.4 clearly shows the superiority of JPEG2000.

4.2.4 Breakthrough?

Up till now JPEG2000 has not experienced its breakthrough. JPEG is still by far the most widespread lossy image compression standard, even though the improvements of JPEG2000 are obvious. Three primary reasons can be found for this

- Legal problems: Software patents on the mathematics of the compression method. The JPEG committee has agreed with over 20 large companies to allow the use of their intellectual property in the area of the compression method, but cannot guarantee that no other patents exist.

- The complexity of the standard. Wavelet transforms and the concepts used in EBCOT are difficult to understand. Few comprehensive literature on the subject exists.

- The good performance of JPEG for good quality images. (only 20% worse than JPEG2000 for bit rates between 0.5-1.0 bpp).

It is doubtful whether or not five years after its introduction JPEG2000 will still break through. The buzz generated about the standard around the start of this millennium has dropped off and very few research publications or new implementations appear in the literature or on the Internet.
Figure 4.6: Zoom of Lena’s face after JPEG2000 compression (a) at 0.40 bpp (b) at 0.20 bpp (c) at 0.10 bpp (d) at 0.04 bpp
Chapter 5

Vector Quantization

In this chapter vector quantization is defined and an encoder-decoder model is presented. This model permits us to formulate a definition of an optimal vector quantizer. The LBG and \( k \)-means algorithms solve the problem of finding optimal vector quantizers. We will establish a link with neural networks and present self-organizing maps and growing neural gas as interesting solutions for the vector quantizer problem.

5.1 Definitions

I have explained that scalar quantization defines a mapping from one input number to one output number. Vector quantization (VQ) is simply a generalization of this and defines a mapping from a possible set of input vectors, in the sense of ordered sets of numbers, to a smaller set of output vectors.

The standard reference work on vector quantization is Gersho and Gray [16]. Many of the definitions and results presented in this section find their origin in this excellent book. To establish notation I define VQ formally:

**Definition** A vector quantizer \( Q \) of dimension \( k \) and size \( L \) is a mapping from an input vector \( \mathbf{x} \) in \( k \)-dimensional Euclidean space, \( \mathbb{R}^k \), into a finite set \( \mathcal{C} = \{ \mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_L \} \) with \( \mathbf{y}_i \in \mathbb{R}^k \) for each \( i \in \mathcal{J} = \{ 1, 2, \ldots, L \} \).

\[
Q : \mathbb{R}^k \rightarrow \mathcal{C}
\]  

(5.1)
The set \( C \) is called the *codebook*, the \( y_i \) are called the *codebook vectors* and \( J \) is called the *index set*.

A vector quantizer can be decomposed into two components, the vector *encoder* and the vector *decoder*. The encoder \( E \) is the mapping from \( \mathbb{R}^k \) to the index set \( J \) and the decoder \( D \) maps the index set \( J \) into the codebook \( C \):

\[
E : \mathbb{R}^k \rightarrow J \quad \text{and} \quad D : J \rightarrow C
\]  

(5.2)

The task of the encoder is to identify in which of \( L \) geometrically specified regions in \( \mathbb{R}^k \), called partitions, the input vector lies. A first type of encoders makes use of the codebook and bases its decision on the distance between the input vector and the codebook vectors. A second type of encoders does not use the codebook, but instead takes a neural network approach. Important examples of both types of encoders will be presented later. On the other hand, the decoder is simply a table lookup and fully determined by specifying the codebook. This vector quantizer model is shown in fig. 5.1.

An *optimal vector quantizer* is determined by a codebook, specifying the decoder, and a partition in \( \mathbb{R}^k \), specifying the encoder, that maximize an overall measure of performance for a training set \( T \)

\[
T = \{x_j \mid j \in [1, A]\}
\]  

(5.3)

The performance measure is often chosen to be the sum of the inverse of the individual distortion measures \( D(x_j, Q(x_j)) \). An optimal vector quantizer should then minimize this total distortion measure.

Necessary conditions for optimal vector quantizers exist. The *nearest neighbor condition* states that for a given decoder, the encoder should be optimal. Otherwise said, an input vector \( x_j \) should be mapped into this codebook vector \( y_p \) that minimizes the distortion.
measure \( D(x_j, y_i) \)

\[
y_p = Q(x_j) \Leftrightarrow D(x_j, y_p) \leq D(x_j, y_i) \quad \forall i \in J
\] (5.4)

The second necessary condition is the centroid condition. This condition states that for a given encoder, the decoder should be optimal. If we call \( \mathcal{R}_i \) the set of all input vectors \( x_j \in \mathcal{T} \) that are mapped onto the same output vector \( y_i = Q(x_j) \)

\[
\mathcal{R}_i = \{ x_j \in \mathcal{T} : y_i = Q(x_j) \quad j \in [1, A] \} \quad \forall i \in J
\] (5.5)

then the codebook vector \( y_i \) should be the centroid of the set \( \mathcal{R}_i \). The centroid \( cent(\mathcal{R}_i) \) is defined as

\[
y_i = cent(\mathcal{R}_i) \Leftrightarrow \sum_{x_j \in \mathcal{R}_i} D(x_j, y_i) \leq \sum_{x_j \in \mathcal{R}_i} D(x_j, y) \quad \forall y \in \mathbb{R}^k
\] (5.6)

The most frequently used distortion measure is the Euclidean distance

\[
D(x, Q(y)) = \|x - y\| = \sqrt{\sum_{n=1}^{k} (x_n - Q(y_n))^2}
\] (5.7)

For this distortion measure the nearest neighbor condition simply states that input vectors should be mapped to the nearest codebook vector, while the centroid condition states the codebook vectors should be the centroids, with centroid used in the usual sense of center of gravity, of the input vectors mapped onto them. In this case the centroid condition (5.6) becomes

\[
cent(\mathcal{R}_i) = \frac{1}{r} \sum_{x_j \in \mathcal{R}_i} x_j \quad \text{with } r = \text{number of vectors in } \mathcal{R}_i
\] (5.8)

All vector quantizers used in this thesis are optimal vector quantizers with the Euclidean distance as the distortion measure.

Suppose one has a vector quantizer that satisfies the centroid and nearest neighbor conditions. What can we tell about the optimality of this vector quantizer? The total distortion for the entire set of input vectors \( x_j \), minimized by an optimal vector quantizer, is given
by
\[ d = \sum_{j=1}^{A} D(x_j, Q(x_j)) \] (5.9)

A vector quantizer that finds a global minimum for this expression is called *globally optimal* and a vector quantizer that finds a local minimum of this expression is called *locally optimal*. It is intuitively evident that the nearest neighbor and centroid conditions will be sufficient conditions for locally optimal vector quantizers. However, what we are looking for is a globally optimal vector quantizer! We should be aware that locally optimal vector quantizers can, in fact, be very suboptimal.

In fig. 5.2 (a) input vectors \( X_1 \) and \( X_3 \) are mapped onto codebook vector \( Y_1 \) and input vectors \( X_2 \) and \( X_4 \) are mapped onto codebook vector \( Y_2 \). The codebook vectors \( Y_1 \) and \( Y_2 \) are locally optimal because a small change in the position of for example \( Y_1 \) always increases the distance \( |X_1 - Y_1|^2 + |X_3 - Y_1|^2 \). However, it is easy to see that \( Y_1 \) and \( Y_2 \) are not globally optimal. The globally optimal codebook vectors \( Y_1, Y_2 \) are shown in fig. 5.2 (b).

![Figure 5.2](image)

5.2 LBG and \( k \)-means Algorithms

An idea to find the globally optimal vector quantizer could be to solve the nearest neighbor condition (5.4) and the centroid condition (5.6). This would yield the codebooks of all locally optimal vector quantizers. Among these, we would then have to search for the globally optimal vector quantizer. But (5.4) and (5.6) are a highly non-linear coupled set of equations and cannot be solved except in a few trivial cases. This means that we should look for another way to find a globally optimal vector quantizer.

The nearest neighbor and centroid conditions provide the basis for iteratively improving a given vector quantizer. The iteration starts from the given vector quantizer and then finds
a new codebook such that the total distortion is no greater (and usually less) than the original quantizer. The repeated application of this improvement step results in a locally optimal vector quantizer. However, for a well-chosen original vector quantizer, one can assume that this locally optimal vector quantizer will also be the globally optimal vector quantizer. Initialization of the codebook with some random vectors from the training set usually yields the globally optimal vector quantizer.

This iterative codebook improvement step is the heart of the generalized Lloyd algorithm or Linde-Buzo-Gray (LBG) algorithm [27]. The basic steps of this algorithm are

**Step 1.** Begin with an initial codebook $C_1 = \{y_1\}$ with $y_1$ chosen randomly (but mutually different) from the training set $T$. Set $t = 1$.

**Step 2.** Given a codebook $C_t = \{y_i\}$, partition the training set $T$ into cluster sets $R_i$ using the nearest neighbor condition:

$$R_i = \{x_j \in T : D(x_j, y_i) < D(x_j, y_k) \forall k \in J, k \neq i \} \forall i \in J \quad (5.10)$$

**Step 3.** Using the centroid condition (5.6) or (5.8), compute the centroids for the cluster sets $R_i$ just found to obtain a new codebook $C_{t+1} = \{cent(R_i)\}$.

**Step 4.** Compute the total distortion for $C_{t+1}$. If it has changed by a small enough amount since the last iteration, stop. Otherwise set $t + 1 \rightarrow t$ and go to step 2.

The LBG algorithm belongs to the category of batch update algorithms. In batch algorithms all inputs are evaluated first before any adaptation is done. In some situations the training set $T$ is so huge that batch update algorithms become impractical. In other cases the input data comes as a continuous sequence of unlimited length which makes the use of batch algorithms impossible.

**On-line update algorithms** perform an update directly after evaluating each sample and should be used in these situations. An on-line algorithm that provides a solution to the optimal vector quantizer problem is the $k$-means algorithm [28].

**Step 1.** Begin with an initial codebook $C_1 = \{y_1\}$. Set $t = 1$.

**Step 2.** Extract randomly an input vector $x_j$ out of the training set $T$. 
Step 3. Given a codebook $C_t = \{y_i\}$, find the codebook vector $y_p$ that is nearest to the input vector $x_j$

$$y_p : D(x_j, y_p) < D(x_j, y_i) \quad \forall i \in J, \ i \neq p \quad (5.11)$$

Step 3. Adapt the codebook vector $y_p$ toward $x_j$ and set $t = t + 1$

$$y_p = y_p + \epsilon[t](x_j - y_p) \quad (5.12)$$

Step 4. Unless a maximum number of steps $m$ is reached continue with step 2

The function $\epsilon[t]$ is called the learning rate. In the k-means algorithm this function is

$$\epsilon[t] = \frac{1}{t} \quad (5.13)$$

This decaying learning rate forces fast convergence during the first iterations while leading to a stable solution during the later iterations.

Suppose that either by the LBG algorithm or the k-means algorithm we have found the globally optimal vector quantizer. The performance of this globally optimal vector quantizer is determined by the total distortion (5.9). This distortion depends on the number of codebook vectors $L$, their values $y_i$ and the distribution in space of the input vectors.

More codebook vectors means smaller distortion, but more bits needed to code the index $i$. For the same number of codebook vectors, input vectors grouped in clusters in $\mathbb{R}^k$ allow for a smaller distortion, while input vectors randomly distributed in $\mathbb{R}^k$ generally have a large total distortion.

### 5.3 Neural Networks

A conventional von Neumann computer has one central processing unit (CPU). The CPU processes one instruction at the time (serial processing) at an extremely high clock speed and its arithmetic and algorithmic capacities exceed by far the capabilities of the human brain. For other tasks however the human brain is superior to the von Neumann computer, for example in image processing:
• Feature extraction. Any operation that extracts significant information from a larger amount of data. In image processing, these features are often geometric or perceptual characteristics (edges, corners,...) or application dependent ones, e.g., facial features.

• Segmentation. Any operation that partitions an image into regions that are coherent with respect to some criterion.

• Object detection and recognition. Determining the position, orientation and scale of specific objects in an image, and classifying these objects.

• Image understanding. Obtaining high level information of what an image shows.

Moreover, the human brain is capable of performing these tasks in noisy and changing environments. This capability of the human brain is due to the presence of an enormous amount of highly interconnected brain cells that permit massive parallel processing of information. This has incited researchers to try to build computer models of the human brain: artificial neural networks (ANN). ANN have effectively been used to perform the above cited image processing tasks [12, 19].

The basic elements of ANN are neurons. A neuron has $k$ inputs $x_1, \ldots, x_k$ and is characterized by $k$ weights $w_1, \ldots, w_k$ and a bias $\theta$. It is often convenient to consider all inputs together as one input vector $\mathbf{x} = [x_1, \ldots, x_k]$ and the weights associated to neuron together as one weight vector $\mathbf{w} = [w_1, \ldots, w_k]$. In general, the neuron’s output $y$ is determined by an activation function $y = \mathcal{F}(\mathbf{x}, \mathbf{w}, \theta)$.

Neurons can be interconnected to construct ANN. For a given network topology the network’s behavior is completely determined by the weights and biases of the neurons in the network. To obtain a desired behavior from an ANN, its weights and biases should be set correctly. This is done during the learning phase or training, which precedes the actual use of the network. ANN are often divided in two categories, based on the type of algorithm by which the neural net has been trained. [40]

A first class of ANN is trained by means of supervised learning. This denotes a method in which a large enough training set of realistic input examples is collected and presented. The output computed by the network is observed and the deviation from the expected answer is measured. The weights and bias are corrected according to the magnitude of the error in the way defined by a learning rule.
A type of ANN that needs to be trained by supervised learning are multi-layered feed-forward ANN. For this type of neural networks, the activation function \( F(x, w, \theta) \) takes a simplified form:

\[
F(x, w, \theta) = F \left( \sum_{i=1}^{n} w_i x_i + \theta \right)
\]

The total input is the weighted sum of the inputs plus a bias term \( \theta \). The activation function \( F \) is either some kind of threshold function (typically a Heaviside function or a sigmoid) with output either high or low, either the function \( F(x) = x \). Fig. 5.3 (a) shows this basic neuron model. Multi-layered feed-forward ANN are simply constructed by interconnection of inputs and outputs of these basic neurons. Fig. 5.3 (b) shows an example of a two-layered feed-forward ANN. Multi-layered feed-forward networks can be trained by the so-called back-propagation learning rule [42].

Another class of ANN are trained by unsupervised learning. In this case, corrections to the weights and biases are not performed based on a training set. Often we do not even know what solution to expect from the network. The network itself decides what output is best for a given input and reorganizes accordingly. Hence, the network needs to find all relevant information within the set of inputs.

An important type of unsupervised learning is competitive learning. In this method the neurons in the network compete with each other for the “right” to provide the output associated with an input vector. In hard competitive learning, only the weights and bias of the winner of this competition are adapted. In soft competitive learning, the weights and bias of runners-up of the competition are adapted as well.

Competitive learning is of particular interest because it is capable of implementing vector quantization. To show this, consider the neural network architecture of fig. 5.4. The
network consist of a first layer of \( L \) neurons \( N_i \). Each neuron has \( k \) inputs and the inputs \( x_1, \ldots, x_k \) are the same for all neurons. Weights \( w_1, \ldots, w_k \) are associated to each neuron.

It is convenient to consider the inputs all together as one input vector \( x = [x_1, \ldots, x_k] \) and the weights associated to neuron \( N_i \) together as one weight vector \( w_i = [w_1, \ldots, w_k] \).

Hence, there are \( L \) weight vectors \( w_1, \ldots, w_L \) associated to \( L \) neurons.

The goal of the network is to find the neuron \( N_p \) that minimizes the Euclidean distance between its weight vector \( w_p \) and the input vector \( x \). To indicate this winner \( N_p \), the network outputs a vector \( y \) of length \( L \). \( y \) outputs all zeros, except at the position \( p \) of the winner. Here the output is 1.

To implement this behavior, each neuron \( N_i \) measures the distance \( D_i = \|x - w_i\| \) and transmits it to the competitive layer \( C \). Sometimes certain neurons are favored by adding a bias \( \theta_i \) to \( D_i : D'_i = D_i + \theta_i \). The competitive layer searches for the neuron that generates the lowest distance \( D'_i \). Suppose this is \( D'_p \). The competitive layer will then output a vector \( y \) of length \( L \) with all zeros, except at position \( p \).

By considering the weight vectors \( w_1, \ldots, w_L \) as codebook vectors, a vector quantizer decoder is specified. Then, the competitive neural network is completely equivalent to a vector quantizer encoder: it takes an input vector \( x \) and outputs a vector \( y \) that indicates the index of the nearest codebook vector following the nearest neighbor condition.

In addition, the \( k \)-means algorithm, section 5.2, can be considered as an on-line update learning rule for competitive learning. It takes an input vector \( x \), finds the neuron \( N_p \) whose weight vector \( w_p \) is nearest to \( x \) (the winner), and applies the learning rule

\[
 w_p = w_p + \epsilon[t](x - w_p) \tag{5.15}
\]
This learning rule moves the weight vector of the winner closer to the input vector by a distance proportional to the difference between input and weight vector. The proportionality factor $\epsilon[t]$ decays as a function of the number of iterations $t$. Equivalently the LBG algorithm can be considered as a batch update learning rule for competitive networks. Both the $k$-means and LBG algorithms are examples of hard competitive learning rules. These remarks establish neural networks as a powerful paradigm for vector quantization.

5.4 Self-Organizing Maps

Self-Organizing maps (SOM) or Kohonen networks are a special type of competitive networks. The neurons in a SOM are ordered in one-, two-, or multi-dimensional grids. The neurons have lateral neighborhood connections to several of their neighbors. These connections are unweighted and symmetric. The distance on the grid between two neurons is defined as the number of neighborhood connections that need to be passed to go from one neuron to the other. Fig. 5.5 (a) shows a two-dimensional rectangular SOM grid with the lateral connections.

SOM are trained by soft competitive learning and the distance on the grid is used to determine how strongly a neuron $i$ is adapted when neuron $c$ is the winner. Mostly the choice is made to adapt all neurons within a certain distance $r[t]$ of the winner $N_c$ and all others not. This critical distance $r[t]$ is made smaller as a function of time. To put this in a mathematical expression we define the neighborhood $H_c[t]$ as the set of neurons at a distance smaller or equal to $r[t]$ from $N_c$. Fig. 5.5 (b) shows the decay of this neighborhood as a function of time for the case of a two-dimensional SOM with hexagonal grid.

Then, a neighborhood function $h_{ci}[t]$ is defined by

$$h_{ci}[t] = \begin{cases} 
\epsilon[t] & N_i \in H_c[t] \\
0 & \text{otherwise.} 
\end{cases}$$  \hspace{1cm} (5.16)

where $\epsilon[t]$ is a monotonically decreasing function of time. $h_{ci}[t]$ is used in the on-line learning rule that updates the weights $w_i$ when an input sample $x_j[t] \in T$ is presented at time $t$:

$$w_i[t + 1] = w_i[t] + h_{ci}[t](x_j[t] - w_i[t])$$  \hspace{1cm} (5.17)

Notice the similarity of this learning rule with (5.15). From this we see that $\epsilon[t]$ takes the
Chapter 5. Vector Quantization

There exists an equivalent batch learning algorithm that resembles the LBG algorithm. Kohonen [25] called it the batch map.

**Step 1.** Initialize the weight vectors of the $L$ neurons, for example with $L$ vectors randomly chosen from the training set $\mathcal{T}$.

**Step 2.** For each neuron $i$, collect a list of copies of all those training samples $x_j \in \mathcal{T}$ whose nearest weight vector belongs to neuron $i$.

**Step 3.** Make the union of all lists belonging to neurons in the neighborhood $H_i[t]$. Replace the weight vector $w_i$ of each neuron $N_i$ by a new weight vector formed by the centroid of all training samples in this union.

**Step 4.** Repeat from step 2 a few times.

Fig. 5.6 shows an example of a SOM batch training process. The input samples are randomly chosen in the unit square. The weight vectors are initialized in the middle of this unit square for better illustration of the training.

SOM have some very interesting properties

**Topology preserving** Given two inputs $x_0$ and $x_1$ that are close to each other in the input space $\mathbb{R}^k$. Then $x_0$ and $x_1$ will activate the same neuron or neighboring neurons. This property can be found in biological neural nets too, e.g., a stimulus on a human’s hand and one on his arm activate neurons in the same area of the

![Diagram of SOM topology and neighborhood](image-url)
Chapter 5. Vector Quantization

human brain. This property is due to the fact that all neurons in the neighborhood of the winner are adapted.

A topology preserving mapping from the high-dimensional input space to a two-dimensional SOM provides an opportunity to illustrate the relationships between different input regions that would otherwise be impossible to visualize. Fig. 5.7 shows an example application of this property from the field of data analysis [22]. In this example, the training set consists of 39-dimensional vectors containing statistical parameters reflecting the welfare of the world’s countries. We can see that the SOM maps countries with comparable welfare are close to each other. It has been suggested that the topology preserving property of SOM is advantageous for image compression because they lead to better entropy coding [1]. This will be investigated later in this thesis.

**Density approximation** Unlike the LBG or k-means algorithm the SOM weight vectors do not minimize the total distortion 5.9, but tends to approximate the probability density function (PDF) of the input vectors. This PDF is projected to the SOM such that the weight vectors are densest in the areas where the PDF is high. Fig.
5.8 illustrates this for a set of inputs randomly chosen in a two-dimensional normal distribution. We see that the \( k \)-means algorithm spreads around the neurons to minimize the total distortion, while the SOM tends to concentrate the neurons more in the high probability area.
5.5 Growing Neural Gas

In this method [15], unlike in the previously described methods, the number of neurons in the network is not fixed a priori. Like the SOM algorithm, it is a soft competitive learning algorithm and topology preserving.

Starting with 2 neurons, new neurons are inserted successively in the network. The place of insertion is determined by local error measurements. The complete learning algorithm goes as follows:

1. Initialize a set $A$ to contain two neurons $N_1$ and $N_2$

$$A = \{N_1, N_2\} \quad (5.18)$$

with weight vectors $w_1, w_2$ chosen randomly in the input training set $T = \{x_j\}$. Initialize the connection set $C$, $C \subset A \times A$ to the empty set:

$$C = \emptyset \quad (5.19)$$

This connection set is a set that contains neighborhood connections between two neurons. These neighborhood connections are unweighted and symmetric:

$$(i, j) \in C \iff (j, i) \in C.$$  

Neuron couples possessing such a connection are called direct topological neighbors.

2. Select at random an input vector $x_j \in T$.

3. Determine the winning neuron $N_{s1}$ and the second-nearest neuron $N_{s2}$ ($N_{s1}, N_{s2} \in A$) by

$$N_{s1} : \| x_j - w_{s1} \| < \| x_j - w_m \| \text{ with } N_{s1} \in A \text{ and } m \in A \setminus \{N_{s1}\} \quad (5.20)$$

and

$$N_{s2} : \| x_j - w_{s2} \| < \| x_j - w_m \| \text{ with } N_{s2} \in A \setminus \{s_1\} \text{ and } m \in A \subset \{N_{s1}, N_{s2}\} \quad (5.21)$$

4. If a connection between $N_{s1}$ and $N_{s2}$ does not already exist, create it:

$$C = C \cup \{(N_{s1}, N_{s2})\} \quad (5.22)$$
Set the age of the connection between $N_{s1}$ and $N_{s2}$ to zero ("refresh" the edge):

$$age[(N_{s1}, N_{s2})] = 0$$  \hspace{1cm} (5.23)

5. Add the squared distance between the input signal and the winner to a local error variable:

$$E_{s1} = E_{s1} + \|x_j - w_{s1}\|^2$$  \hspace{1cm} (5.24)

6. Adapt the reference vectors of the winner and its direct topological neighbors by fractions $\epsilon_b$ and $\epsilon_n$, respectively, of the total distance to the input signal:

$$w_{s1} = w_{s1} + \epsilon_b(x_j - w_{s1})$$  \hspace{1cm} (5.25)

$$w_i = w_i + \epsilon_n(x_j - w_i) \hspace{1cm} (\forall i \in H_{s1})$$  \hspace{1cm} (5.26)

Thereby $H_{s1}$ is the set of direct topological neighbors of $N_{s1}$

($H_{s1} = \{i \in \mathcal{A} \mid (c, i) \in \mathcal{C}\}$).

7. Increment the age of all edges emanating from $N_{s1}$:

$$age[(N_{s1}, i)] = age[(N_{s1}, i)] + 1 \hspace{1cm} (\forall i \in H_{s1})$$  \hspace{1cm} (5.27)

8. Remove edges with an age larger than $a_{max}$. If this results in units having no more emanating edges, remove those units as well.

9. If the number of input signals generated so far is an integer multiple of a parameter $\lambda$, insert a new neuron as follows:

(a) Determine the neuron $N_q$ with the maximum accumulated error:

$$N_q : E_q > E_m \text{ with } N_q \in \mathcal{A} \text{ and } m \in \mathcal{A}\{N_q\}$$  \hspace{1cm} (5.28)

(b) Determine among the neighbors of $N_q$ the neuron $N_f$ with the maximum accumulated error:

$$N_f : E_f > E_m \text{ with } N_f \in \mathcal{H}_q \text{ and } m \in \mathcal{H}_q\{N_q\}$$  \hspace{1cm} (5.29)

(c) Add a new neuron $N_r$ to the network and interpolate its reference vector from
Chapter 5. Vector Quantization

\[ \mathcal{A} = \mathcal{A} \cup \{N_r\}, \quad \mathbf{w}_r = (\mathbf{w}_q + \mathbf{w}_f)/2 \]  
(5.30)

(d) Insert edges connecting the new neuron \(N_r\) with neurons \(N_q\) and \(N_f\), and remove the original edge between \(N_q\) and \(N_f\):

\[ C = C \cup \{(N_r, N_q), (N_r, N_f)\}, \quad C = C \setminus \{(N_q, N_f)\} \]  
(5.31)

(e) Decrease the error variables of \(N_q\) and \(N_f\) by a fraction \(\alpha\):

\[ E_q = (1 - \alpha)E_q, \quad E_f = (1 - \alpha)E_f \]  
(5.32)

(f) Interpolate the error variable of \(N_r\) from \(N_q\) and \(N_f\):

\[ E_r = (E_q + E_f)/2 \]  
(5.33)

10. Decrease the error variables of all units:

\[ E_c = (1 - \beta)E_c, \quad \forall c \in \mathcal{A} \]  
(5.34)

11. If a stopping criterion (e.g., net size or some performance measure) is not yet fulfilled continue with step 2.

Typical parameter values are \(\lambda = 300, \epsilon_b = 0.05, \epsilon_n = 0.0006, \alpha = 0.5, \beta = 0.0005, a_{\text{max}} = 88\). Fig. 5.9 shows a ”video” from the construction of a growing neural gas network for an example input distribution.

Growing neural gas have never been used for image compression. However, because they introduce a topology, we could expect them to behave at least as well as SOM. Their growing construction opens up design options not possible with SOM like for example adapting the number of codebook vectors to the input distribution. Because neurons that never win are removed during the learning phase, a better approximation of the input vector space is obtained than with SOM. This is illustrated in fig. 5.10.

The same two-cluster discrete input space that was so well approximated by a growing neural gas of fig. 5.9 is not at all well approximated by a 4 × 4 SOM. Many SOM neurons converge to positions outside the input space and are thus useless. This argument leads
Figure 5.9: Training "video" for growing neural gas for an example input distribution. (a) 2 neurons (b) 3 neurons (c) 4 neurons (d) 5 neurons (e) 8 neurons (f) 12 neurons

to the assumption that growing neural gas will give better results in image compression than SOM.

Figure 5.10: 4 × 4 SOM obtained by training the input set with SOM batch algorithm
Chapter 6

VQ in image compression

In this chapter the use of VQ in image compression is developed. There exists a theoretical result, the \textit{vector quantization paradigm}, that states that VQ can attain the maximum possible theoretical compression. Because the computational cost of this method grows exponentially with dimensionality, it cannot be used in practice.

Vector quantization can however be used successfully for image compression. DCT/VQ, classified VQ and finite-state VQ are interesting implementations and are presented in this chapter.

6.1 VQ paradigm

In a 1948 paper Shannon [44] showed that fundamental limits to lossless and lossy data compression (with a fixed distortion) exist. A very interesting result is that these limits can be obtained, in theory, with vector quantization [16].

To use this \textit{vector quantization paradigm} in practice, the \(N\)-bits data sequence should be split up in shorter length vectors and these vectors should be vector quantized apart. In the context of image compression this means that an image should be decomposed into blocks and that vectors should be formed from this blocks. This method will of course not yield compression at the Shannon limit, but stays very powerful.

To understand why, consider a decomposition of an image in \(k\)-dimensional vectors \(x_j\). If \(k\) is small enough, we can expect that the important inter-pixel correlations that can be found in most images will result in important correlations between the \(x_j\) as well. This
translates into low distances in $\mathbb{R}^k$ between these $x_i$ that are correlated, that is, the $x_j$ will be clustered in $\mathbb{R}^k$.

Suppose now that we have as much codebook vectors $y_i$ as there are clusters, then the codebook vectors will be placed at the centroids of the clusters and replacing the $x_j$ by $y_i = Q(x_j)$ will not introduce much error. Fig. 6.1 illustrates this idea for a trivial two-dimensional example.

![Figure 6.1: Example of a codebook \{y_i\} for the case of clustered input vectors](image)

Vector quantization maps each $x_j$ onto a codebook vector $y_i$ and store the index $i$ of the codebook vectors. Whereas storing a $k$-dimensional vector containing 8-bit gray-scale values requires $8k$ bit, storing an index $i$ of length-$L$ codebook requires only $\log_2(L)$ bit. This gives a compression rate $cr$ of

$$cr = \frac{\log_2(L)}{8k} \quad (6.1)$$

However, some important problems make it impossible to use basic vector quantization successfully in image compression. Suppose we have found a globally optimal length-$L$ codebook for a training set of input vectors $x_j$, then in general this codebook will not be the best length-$L$ codebook for another set of input vectors $\tilde{x}_j$. Or, speaking in terms of images, the best codebook is image-dependent.

If, for example, we would encode a $256 \times 256$ pixels image and transmit a codebook of 256 vectors from $4 \times 4$ blocks, then $(256 \times 4 \times 4 \times 8)/(256 \times 256) = 0.5$ bpp would be needed only to encode the codebook. Another 0.5 bpp would be needed to encode the image itself. So
obviously, it is not possible to transmit a codebook for each image. The compression rate would diminish too much.

This means the codebook should be specified in advance. It is typically found by applying a training method like the LBG algorithm to a set of training vectors coming from some training images. For special categories of images, for example finger prints or medical images, the training images are chosen to be of the same category of the images to be encoded and it is possible to obtain good quality compression with relatively small codebooks.

When the nature of the image to encode is not known, a universal codebook constituted from a diverse set of training images is to be used. To result in a high quality images, such a universal codebook should be very large. To identify a codebook vector in an extremely large codebook many bits are needed but this cannot be allowed since it would cancel out the compression. When the universal codebook is too small blocking artifacts appear.

Another serious problem is edge degradation by a staircase effect. The origin of this effect problem lies in the codebook design. Most images consist of large smooth areas separated by edges. This means that the edge blocks in the training images are rare compared to the smooth blocks. Hence, the codebook will contain much more smooth blocks than edge blocks. Because there are not enough edge blocks in the codebook, edge blocks risk to be represented by smooth blocks. This causes the underlying block encoding to become
visible at edges.

Fig. 6.2 shows a 256 × 256 sized Lena image that was encoded with the codebook of 256 vectors found by the LBG algorithm and Lena itself as the training image. The quality of the resulting 0.5 bpp (bit per pixel) image is unacceptably low. Sure enough the strong block artifacts and edge degradation are present.

6.2 DCT/VQ

Many different solutions have been tried out to overcome the problems of basic VQ. Overviews can be found in [33] and [16]. An interesting type of vector quantizers for image compression are transform vector quantizers. These quantizers simply perform an image transform and follow this by a basic vector quantization. The DCT is an ideal transform to use in this configuration and it should be applied on image blocks anyway. Methods that combine DCT and VQ are referred to as DCT/VQ methods.

If the image blocks are small blocks of for example 4 × 4 pixels, this yields 16-dimensional vectors of DCT coefficients. We could try to vector quantize an input space of these 16-dimensional vectors, but this is not necessary.

High-frequency DCT coefficients are typically close to zero and can be omitted. We could for example keep only the 6 lowest frequency components and work with 6-dimensional vectors. This potentially has some advantages:

- A smaller dimensional input space can be better filled with a limited number of codebook vectors.
- Low frequency DCT coefficients are more significant and much higher than high-frequency DCT coefficients. Hence, an input vector is mapped onto this codebook vectors that best matches its low frequency components. This leads to no correlation between the high-frequency components of the input vector and its corresponding codebook vector. So it is a good choice to assume that the high-frequency components are zero.

If the transform is applied to large blocks of for example 8 × 8 pixels, the resulting 64-dimensional vectors are too large to be quantized at once. Even vectors that only keep the significant low-frequency components would be too large. In this case, the 8 × 8 matrix of DCT coefficients should be divided into several subvectors that are quantized apart.
6.3 Classified VQ

Classified vector quantization was introduced in 1986 by Ramamurthi and Gersho [37] and one of the first vector quantization methods yielding good results for image compression. The idea is to classify inputs into a certain number of classes. To each class corresponds a codebook. Then, the inputs are mapped onto a codebook vector in the codebook corresponding to its class. Fig. 6.3 illustrates this idea. Classified vector quantization can provide a solution to the problem of edge degradation in image compression by encoding edge-blocks by means of a codebook of edge blocks. Ramamurthi and Gersho [37] provide a sophisticated solution. For $4 \times 4$ blocks, they identify 31 different classes of horizontal, vertical and diagonal edge blocks and smooth blocks. For example, the 6 horizontal classes they identify are intuitive and illustrated in fig. 6.4. The biggest problem in classified VQ is the classifier. It needs to be able to identify blocks as belonging to a certain class, but this is not always straightforward. Look for example at fig. 6.4 (b). This block could classified in class (a) as well as in class (b). This ambiguity poses a problem for the
codebook design. If the classifier would put block 6.4 (b) in class (a) and a very similar block in class (b), the codebooks of class (a) and (b) will be similar and hence redundant. This stresses the importance of an accurate and consistent classifier.

In [37] the classifier is implemented by an edge enhancement step and a decision tree based on some threshold values. A shortcoming of this classifier is the somewhat arbitrary choice of these thresholds. The authors report a PSNR of 29.79 dB at 0.7 bpp for Lena (without entropy coding).

Ho and Gersho [18] proposed a method where classification takes place in the transform domain. They partitioned images in $8 \times 8$ blocks and perform a DCT over each block. This method could be categorized as a classified DCT/VQ method. They noticed that high DCT coefficients in certain areas correspond to edges in certain directions: fig.6.5 (a). By calculating the signal energy in these regions, blocks are categorized in 4 groups: horizontal (H), vertical (V) and diagonal (D) edge blocks and smooth blocks (L).

![H: energy in horiz. region, V: energy in vert. region, D: energy in diag. region](image)

Figure 6.5: (a) DCT areas corresponding to edges (b) subvector partition [18]

Because $8 \times 8$ blocks are too large to be vector quantized, they partition the blocks into a number of subvectors of small dimensions. This partition changes for each class and is illustrated in fig. 6.5 (b). The authors claim this method yields PSNR= 31.05dB at 0.302 bpp for Lena (with entropy coding).
Kim and Lee [23] propose a much simpler classifier. They divide images in $4 \times 4$ blocks and transform to the DCT domain. It is a classified DCT/VQ method. They identify the DCT coefficients as indicated in fig. 6.6.

A remarkable property is that the edge pattern of a $4 \times 4$ block is completely specified by the signs of the coefficients $d_{2,1}, d_{3,1}, d_{1,2}$ and $d_{3,1}$. Fig. 6.6 (b) shows that a horizontal edge is determined by the signs of $d_{2,1}$ and $d_{3,1}$. Similar relations hold for vertical and diagonal edges.

Instead of trying to find suitable thresholds to classify the blocks Kim and Lee construct a large set of 16-dimensional DCT-transformed vectors from training images. From these training vectors they construct 4-dimensional vectors $[d_{2,1} \ d_{3,1} \ d_{1,2} \ d_{3,1}]$ and use these to construct a codebook of 16 4-dimensional vectors by the LBG algorithm. These 16 vectors correspond to a classification in 16 classes.

The next step is to classify the complete 16-dimensional training vectors by comparison of the distance between their coefficients $[d_{2,1} \ d_{3,1} \ d_{1,2} \ d_{1,3}]$ and the codebook vectors. This yields 16 sets of training vectors corresponding to 16 classes. The training vectors in each class are then vector quantized by the LBG algorithm to for example 128 vectors. For this configuration of 16 classes of 128 vectors (0.75 bpp) a result of PSNR=32.88 is reported for Lena.

### 6.4 Finite-State VQ

Finite-state vector quantizers classify input blocks in classes and use a different codebook for each class, just like classified VQ. The difference is that in classified VQ the class
of the input block is measured by a classifier while in finite-state VQ it is predicted by considering the previously encoded blocks and their classes. This information defines the "state" of the VQ.

The advantage of finite-state VQ is that no index to the class of the block needs to be stored. The problem of finite-state VQ is of course the accuracy of the prediction. If a block predicted to be a smooth block is in reality an edge block, then this edge block will be encoded by a smooth block. It is clear that if such errors occur too frequently the image quality will suffer seriously. Aravind and Gersho [2] presented a pioneering work on finite-state VQ.

Kim [24] introduced a finite-state VQ design called side match VQ. In this design, the state is determined by the gray-scale values of those pixels of previously encoded blocks that are adjacent to the block to be encoded. Fig. 6.7 shows the pixels that determine the state for the case of 4 × 4 blocks and establish the notation. The state is determined by the vectors \( u = [u_1 \ u_2 \ u_3 \ u_4] \) and \( v = [v_1 \ v_2 \ v_3 \ v_4] \).

Kim uses a super-codebook of codebook vectors \( x \) and defines the codebook to use at the state \((u, v)\) by the subset of the super-codebook that minimizes the distortion \( D_{\text{tot}} = D(u, x_c) + D(v, x_r) \) with \( x_r = [x_{11} \ x_{12} \ x_{13} \ x_{14}] \) the first row vector and \( x_c = [x_{11} \ x_{21} \ x_{31} \ x_{41}] \) the first column vector of the super-codebook vectors. The author reported a PSNR of 30 dB at 0.25 bpp for Lena.

To my knowledge no result on finite-state VQ in the DCT domain has been published. It could be interesting to research this.
In this section I present an image compression scheme based on SOM developed by Amerijckx et al. [1]. At first, the image is split in blocks of for example $4 \times 4$ pixels. These blocks are transformed by a DCT.

Only the first 6 DCT coefficients are retained and used to form 6-dimensional vectors $\mathbf{x}_i = [d_{1,1}, d_{1,2}, d_{2,1}, d_{2,2}, d_{3,1}, d_{1,3}]$ (notation of fig. 6.6).

These vectors are encoded by vector quantization with a universal codebook. This codebook is found by training a SOM over an input space of training vectors from some training images. The neurons of this SOM form the universal codebook.

The topology preserving property of SOM causes input vectors that are very close to each other to be mapped on either the same neurons or two neighboring neurons. In images, two blocks that are successively encoded often do not differ much. In this case, topology preservation should cause these blocks to be encoded by neurons that are close to each other.

The authors claim that this property leads to better entropy coding by encoding the difference between the indexes of successively coded blocks instead the actual indexes and propose the use of a first-order differential coder. This differential coder is illustrated in fig. 6.8.

The index of block $i$ is encoded by taking the difference with the index of either block $b,d,f$ or $h$. To determine which of these blocks to consider as the previous block, the differences between the indexes of blocks $a$ and $b$, blocks $c$ and $d$, blocks $e$ and $f$ and blocks $h$ and $g$ are taken. The pair that has the smallest difference determines the block to which the block $i$ is compared.
If, for example the difference between the indexes of blocks $c$ and $d$ is the smallest, block $i$ is encoded by the difference between the indexes of $i$ and $d$. The output of this differential coder is entropy coded by a variable-rate coder that features run-length and Huffman coding.

The authors claim that this method outperforms the JPEG standard for compression rates higher than 25. This is an interesting result, the more that the codebook is found by a DCT/SOM equivalent to DCT/VQ.

It could be expected that extending this method to a classified $DCT/SOM$ could yield significant performance improvement, in the same way that DCT/VQ can be improved by classified DCT/VQ. Similarly, a finite-state DCT/SOM should be expected to improve a finite-state DCT/VQ. This will be investigated in the next chapter.
Chapter 7

SOM and image compression

In this chapter I evaluate the use of SOM for image compression. The DCT/SOM scheme of Amerijckx et al. [1], section 6.5, is the starting point of the evaluation and is compared to a DCT/VQ scheme.

7.1 Tools

The schemes were implemented in Matlab and codes can be found on the CD-ROM included.

I made use of the free SOM toolbox developed by the Laboratory of Information and Computer Science at Helsinki University of Technology\(^1\). This toolbox features the SOM-batch and SOM-online training algorithms and a method to find the index of the neuron or codebook vector closest to a given input vector.

Because the traditional \(k\)-means algorithm is slow, I used an accelerated version courtesy of Elkan [13] and available at his internet site\(^2\). Entropy coding is applied to all schemes. I chose to use an entropy coder that features differential and Huffman coding. The differential coder is the first-order differential coder of [1] that was explained in section 6.5. The Huffman coder I used is an implementation courtesy of Karl Skretting\(^3\).

All codebooks are found by training a set of training vectors with either the LBG or SOM-batch algorithms. The training vectors are taken from a set of 6 training images:

\(^1\)http://www.cis.hut.fi
\(^2\)http://www-cse.ucsd.edu/users/elkan/fastkmeans.html
\(^3\)Karl Skretting, Stavanger University, Signal Processing Group http://www.ux.his.no/~karlsk/
Tiffany, Lake, Peppers, Elaine, Boat and House. The test image is Lena. These images are shown by fig. 1.3 on page 8.

### 7.2 DCT/VQ

The first scheme implemented is a DCT/VQ on blocks of $4 \times 4$ pixels. I propose an improvement to this scheme. The compression rate can be improved by exploiting that the first DCT coefficient corresponds to the average gray-scale value of the block.

If we suppose that most parts of the image are smooth, neighboring blocks will often not differ much in average value. If we order the codebook vectors by the value of their first element and apply a differential coding to the indexes obtained after vector quantization, most of the codes will be small and this should result in more compression after Huffman coding.

I investigate the influence of the size of the codebook and codebook vectors on the image quality and the compression rate after entropy coding. Codebook sizes are 256, 128 and 64 and the vectors are 6-, 10- or 16-dimensional vectors.

<table>
<thead>
<tr>
<th>CB vector size</th>
<th>CB Size</th>
<th>PSNR</th>
<th>Bit Rate (in bpp)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>no E.C.</td>
<td>with E.C.</td>
</tr>
<tr>
<td>6</td>
<td>256</td>
<td>31.66</td>
<td>0.50</td>
</tr>
<tr>
<td>6</td>
<td>128</td>
<td>30.87</td>
<td>0.44</td>
</tr>
<tr>
<td>6</td>
<td>64</td>
<td>29.97</td>
<td>0.40</td>
</tr>
<tr>
<td>6</td>
<td>32</td>
<td>28.93</td>
<td>0.31</td>
</tr>
<tr>
<td>10</td>
<td>256</td>
<td>31.55</td>
<td>0.50</td>
</tr>
<tr>
<td>10</td>
<td>128</td>
<td>30.85</td>
<td>0.44</td>
</tr>
<tr>
<td>10</td>
<td>64</td>
<td>29.95</td>
<td>0.40</td>
</tr>
<tr>
<td>16</td>
<td>256</td>
<td>31.54</td>
<td>0.50</td>
</tr>
<tr>
<td>16</td>
<td>128</td>
<td>30.85</td>
<td>0.44</td>
</tr>
<tr>
<td>16</td>
<td>64</td>
<td>30.00</td>
<td>0.40</td>
</tr>
</tbody>
</table>

Table 7.1: DCT/VQ on Lena

The 6-dimensional vectors contain the 6 lowest frequency DCT components, the 10-dimensional vectors the 10 lowest DCT components and the 16-dimensional vectors all DCT components from the $4 \times 4$ block.
Table 7.1 shows the results for Lena. E.C. stands for entropy coding and CB for codebook. The compression is given by the bit-rate in units bits per pixel (bpp).

These results permit to conclude that the number of retained DCT components does not affect PSNR and bit rate much. On the other hand, the proposed ordering of the codebook vectors lowers the bit rate in all cases.

7.3 DCT/SOM

In this section results on the DCT/SOM scheme of Amerijckx et al. [1], section 6.5, are presented. The documentation in this article is not complete. The authors claim interesting results for a SOM of 128 neurons, but do not tell the dimension of the SOM they used. Most probably they used a two-dimensional SOM of $16 \times 8$ neurons.

They also claim that the topology preserving property of SOM make it well suited for differential coding and present an interesting first-order differential coder. However, they do not explain how they extend their differential coder to two dimensions. This is not a trivial problem because on a two-dimensional rectangular grid each neuron has four neighbors.

In my version of this compression system I chose to code the differences in horizontal and vertical directions apart, but this doubles the number of symbols to encode. Table 7.2 shows the results for Lena.

<table>
<thead>
<tr>
<th>number of neurons</th>
<th>SOM dim.</th>
<th>PSNR</th>
<th>Compression (in bpp)</th>
</tr>
</thead>
<tbody>
<tr>
<td>512</td>
<td>$32 \times 16$</td>
<td>31.35</td>
<td>0.46</td>
</tr>
<tr>
<td>256</td>
<td>$16 \times 16$</td>
<td>30.68</td>
<td>0.39</td>
</tr>
<tr>
<td>128</td>
<td>$16 \times 8$</td>
<td>29.67</td>
<td>0.33</td>
</tr>
<tr>
<td>64</td>
<td>$8 \times 8$</td>
<td>28.78</td>
<td>0.33</td>
</tr>
</tbody>
</table>

Table 7.2: Results of DCT/SOM compression on Lena.

This results are very different from the results reported in [1]. Table 7.3 shows the results reported by these authors. First, from comparing the compression rate of my results to these results for the same number of neurons, it shows that the differential coder in [1] performs better than my two-dimensional extension of a one-dimensional differential coder. However, their compression results for a fixed number of neurons are lower than the compression results obtained by the DCT/VQ scheme with ordered codebook (table 7.1).
This shows that the ordering introduced by the topology preserving property of SOM is not as good as a simple ordering by the DC-component of the codebook vectors. Of course, this argument only holds because we are working in the DCT domain. In the space domain it is more difficult to introduce an ordering in the codebook vectors and in this case SOM could facilitate this task considerably.

However, VQ in the space domain faces many obstacles, section 6.1, and even though solutions (e.g. classified VQ) have been proposed earlier in this work, they are not nearly as good as similar solutions in the DCT domain. Working in a transform domain is essential to the success of VQ.

An astonishing result is that the PSNR values I found are much higher than those reported by Amerijckx et al. [1] and that this is due to uncorrect results for PSNR values in [1]! Not only are the PSNR values reported for their method incorrect, the reference PSNR values for JPEG they used are also incorrect! This completely undermines the result claimed that this DCT/SOM method yields better results than JPEG in terms of PSNR for compression rates higher than 25. Fig. 7.1 (a) shows a graph taken from [1]. Fig. 7.1 (b) shows the results I obtained. The curve for JPEG was obtained with the routines available in Matlab. The compression is expressed in compression rate.

<table>
<thead>
<tr>
<th>number of neurons</th>
<th>PSNR</th>
<th>Compression (in bpp)</th>
</tr>
</thead>
<tbody>
<tr>
<td>512</td>
<td>25.41</td>
<td>0.42</td>
</tr>
<tr>
<td>256</td>
<td>25.34</td>
<td>0.37</td>
</tr>
<tr>
<td>128</td>
<td>24.8</td>
<td>0.25</td>
</tr>
</tbody>
</table>

Table 7.3: results of Amerijckx et al. [1] on Lena.

A comparison of DCT/SOM with the basic DCT/VQ with ordered codebook from section 7.2 also reveals that for the same number of codebook vectors DCT/SOM performs worse than DCT/VQ, in terms of PSNR as well as in terms of achieved compression.

This leads to the conclusion that a DCT/SOM compression system has absolutely no reason of existence. This result, together with the error in [1] puts a question mark on the benefits of self-organizing maps for image compression.

In recent years a real buzz was generated on self-organizing maps. They have been advocated for many different applications. Can we conclude that SOM were too soon considered as magical neural networks useful in all kinds of applications such as image compression? In the next section I will analyze some other image compression schemes based on SOM to try to find this out.
Chapter 7. SOM and image compression

Figure 7.1: Comparison of results of JPEG and DCT/SOM compression on Lena. (a) by Amerijckx et al. [1] (b) correct results

7.4 Other SOM methods

McAuliffe et al. [31] compared the LBG algorithm and SOM for image compression. The authors used 16-dimensional vectors from $4 \times 4$-blocks. They did not apply any transformation to these vectors and worked with codebooks of 256 entries. They found that SOM and LBG methods give comparable good quality results (PSNR=31 dB for their test image), but that the SOM method is less sensitive to initialization and converges more quickly than the LBG method. Hence, they conclude that SOM are most useful for image compression.

Some remarks can be made on this result. First, their arguments of lesser sensitivity to initialization and faster convergences are not important. The codebook needs to be found only once and this can be done by an image compression researcher. Then the codebook can be distributed to the users, who do not need to re-calculate a codebook, and it can be used to encode images an infinite number of times. Hence, there is no need for fast convergence or "good" initial conditions.

Second, the training and test images used were infrared images and the authors did not give further information on their content. I believe that the authors chose resembling training and test images. If not, they would have obtained the image quality of fig. 6.2. This quality is unacceptably low. Remember that this low quality is due to the fact that a 16-dimensional space of grayscale values is too large to be quantized by 256 vectors, for the SOM method as well as for the LBG method.
To better preserve edge quality, Kangas [21] proposed to "weigh" the training vectors. Training vectors at edges are repeated in the training set and this causes more edge vectors in the codebook. He also exploits the SOM topology to improve on entropy coding by predictive coding. Results of PSNR around 34.5 at 0.36 bpp are reported for the special class of facial images.

Kangas did not try out his method on a different types of images with a universal codebook. Similar bad quality should be expected in this case. From fig. 6.2 we can easily see that better edge quality would not change much: blocking artifacts also occur in smooth areas. This makes this method impossible for general image compression schemes.

Recently, Laha et al. [26] proposed a SOM method in the spatial domain that eliminates the blocking artifacts. They modify codebook vectors found by a SOM algorithm such that the $4 \times 4$ or $8 \times 8$ blocks corresponding to the codebook vectors fit on a cubic polynomial surface. They claim that this modification of the codebook vectors eliminates the blocking artifacts. They use mean-removed vectors to construct a universal codebook. Further, they use Huffman coding of the indexes generated by the encoder and differential coding of the mean values of the vectors. For Lena, they report a PSNR of 28.47 dB at 0.18 bpp. They say JPEG compresses to a PSNR of 28.24 at 0.20 bpp, but claim better psychovisual quality with their method.

Some remarks on this method should be given. There is no reason to use a SOM in this method instead of the LBG algorithm. The authors say that they used a SOM because of its density approximation property which should better preserve finer details. However, density approximation is not a desired property!

This can easily be understood from considering a typical training image with large smooth areas and few edge blocks. Density approximation on such an image leads to a situation where neurons are preserving detail in large smooth areas where it is absolutely not necessary, while more rare edge blocks, which do contain detail, are not represented in the codebook. Based on the results of McAuliffe et al. we can conclude that an equivalent method based on basic vector quantization should yield comparable results.

On the other hand, the method could be expected to improve compression slightly by exploiting the topology preserving property of SOM in a differential entropy coding scheme, but Laha et al. did not implement this.

Remember from table 7.1 that DCT/VQ yields PSNR 29.97 at 0.19 bpp. A DCT/VQ compressed image's quality is not affected by blocking artifacts. We can conclude that
DCT/VQ outperforms Laha et al.’s method by about 1.5 dB PSNR at a similar low-bit rate, but is computationally more complex because of the DCT.
Chapter 8

Evaluation

In this chapter I present the implementation of a classified DCT/VQ scheme. I investigate if this scheme improves the DCT/VQ scheme. Both schemes are compared to the standards JPEG and JPEG2000.

8.1 Classified VQ of Kim & Lee

In this section the classified VQ of Kim & Lee from section 6.3 is implemented. Kim and Lee claim that a division in 16 classes yields best results, but to investigate the performance of classified VQ at low-bit rates I also consider the cases of 8 and 4 classes. To store an image compressed by classified VQ two indexes should be stored: an index to identify the class of a block and an index to identify the codebook vector in the codebook corresponding to the block’s class.

The indexes to the classes are stored in a matrix $A$ and the indexes to the codebook vector in a matrix $B$. For a $512 \times 512$ pixels image and $4 \times 4$ pixels blocks, matrixes $A$ and $B$ are of dimension $128 \times 128$.

Remember that the classifier was implemented by vector quantization on these DCT coefficients that determine the edge character of a block. Hence, the probability of an image block belonging a class is not equal for all classes. Most image blocks are smooth non-edge blocks and are classified in the same class.

To get an idea of the probability that a random image block belongs to a class, we can classify all training blocks. Fig. 8.1 (a) shows the percentage of training blocks that are
classified in each class for the case of 8 classes. The classes are ordered in descending order of this percentage.

![Diagram](image)

Figure 8.1: (a) Classes ordered in descending order (b) Classes ordered for best entropy coding

I propose an improvement on this classified VQ scheme in view of better entropy coding of the matrix $A$ after differential coding. Because the majority of blocks are in the smooth class, differential coding of $A$ yields smaller codes when the other classes with high probability are ordered such that their indexes are as close as possible to the index of the smooth class.

Fig. 8.1 (b) shows the order for the case of 8 classes that minimizes the expectation value of the distance between subsequent class indexes.

To prove this, I compared compression obtained for the class orderings of fig. 8.1 (a) and fig. 8.1 (b) for Lena. I found that the elements of matrix $A$ were encoded with 1.46 bits on average for case (a) compared to 1.34 bits for case (b). This shows in effect that the proposed ordering yields higher compression.

For each of these classes a codebook is found by vector quantization of the training vectors in the class. The codebook vectors are ordered in ascending order of their first coefficient (the average value) as explained in section 7.2.

Notice that the number of classes does not seem to have much effect on the relation between PSNR and compression. Comparison of these results to the results for DCT/VQ of table 7.1 learns that for similar bit rates both methods yield similar PSNR. The result for 16 classes and 128 codebook vectors per class is about 1 dB PSNR lower than reported by Kim and Lee, but this could be due to a different choice of training images.
As expected, the improvement of classified VQ is not noticeable in the PSNR, but should be visible in the reconstruction of edges. Fig. 8.2 shows a zoom of Lena’s shoulder for (a) a DCT/VQ encoded image with a 256-entries codebook and (b) a classified DCT/VQ encoded image with 16 classes and a 32-entries codebook per class. Both of these images have a bit-rate of 0.30 bpp.

Surprisingly, the edges of the DCT/VQ image are better than the edges in the classified DCT/VQ image! The edges in the classified DCT/VQ image seem to be ”overtrained”. The diagonal edge blocks are too white in the lower left corner and too black in the upper right corner.

This effect is due to the low number (32) of codebook vectors in each class. Making this number of codebook vectors higher is not a solution. This would increase the bit-rate greatly and we know a priori that the result at higher bit-rates will not be competitive to JPEG or JPEG2000.

Lowering the number of classes could be a solution to increase the size of the each codebook without increasing the bit-rate, but some experiments have shown me that the results still do not attain the quality of DCT/VQ.

Another problem is the fact that classified DCT/VQ needs to store indexes to class and codebook vector. Finite-state VQ could be expected to yield some improvement if one could predict the class from the previously encoded class. Then we would gain the space for storing a class index and this would lower the bit-rate. Such a system would however

<table>
<thead>
<tr>
<th>Number of Classes</th>
<th>CB Size</th>
<th>PSNR</th>
<th>Bit Rate (in bpp)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>for Class</td>
</tr>
<tr>
<td>16</td>
<td>128</td>
<td>31.57</td>
<td>0.12</td>
</tr>
<tr>
<td>16</td>
<td>64</td>
<td>31.37</td>
<td>0.12</td>
</tr>
<tr>
<td>16</td>
<td>32</td>
<td>31.02</td>
<td>0.12</td>
</tr>
<tr>
<td>16</td>
<td>16</td>
<td>30.05</td>
<td>0.12</td>
</tr>
<tr>
<td>8</td>
<td>128</td>
<td>31.39</td>
<td>0.08</td>
</tr>
<tr>
<td>8</td>
<td>64</td>
<td>31.12</td>
<td>0.08</td>
</tr>
<tr>
<td>8</td>
<td>32</td>
<td>30.66</td>
<td>0.08</td>
</tr>
<tr>
<td>8</td>
<td>16</td>
<td>29.85</td>
<td>0.08</td>
</tr>
<tr>
<td>4</td>
<td>128</td>
<td>31.13</td>
<td>0.05</td>
</tr>
<tr>
<td>4</td>
<td>64</td>
<td>30.71</td>
<td>0.05</td>
</tr>
<tr>
<td>4</td>
<td>32</td>
<td>30.18</td>
<td>0.05</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>29.35</td>
<td>0.05</td>
</tr>
</tbody>
</table>

Table 8.1: Results for Lena compressed with different classified DCT/VQ configurations
be complicated and accurate prediction difficult to achieve.

### 8.2 Comparison to JPEG, JPEG2000

New lossy image compression systems should be compared to JPEG and JPEG2000 to evaluate their performance. Fig. 8.3 shows a comparison between JPEG, JPEG2000, DCT/VQ and classified DCT/VQ for 4 classes. The values for basic DCT/VQ and classified DCT/VQ are from table 7.1 and table 8.1. The JPEG compressed images were calculated by the Matlab implementation of JPEG. The JPEG2000 images were calculated by the free JASPER codec of M.D. Adams\(^1\).

From this figure we see that JPEG2000 is far superior to JPEG and DCT/VQ methods. For low bit-rates JPEG and DCT/VQ give similar PSNR values. Fig. 8.4 shows Lena at bit rates close to 0.20 bpp compressed by each of these four methods.

We can see that at this low bit rate DCT/VQ and classified DCT/VQ result in an image quality superior to JPEG, but much worse than JPEG2000. The DCT/VQ methods do not have the same flexibility as JPEG and JPEG2000. The JPEG bit rate is controlled by a quality factor that is a good, albeit not exact, indicator for bit rate and PSNR. The JPEG2000 bit rate is precisely set by the user and the embedded coder guarantees the best quality image at this bit rate.

\(^1\)http://www.ece.uvic.ca/~mdadams/jasper/
With DCT/VQ methods the bit rate is set precisely, but the resulting image quality cannot be predicted well. It depends heavily on the relation between the training images by which the codebook was found and the image to be encoded. To incorporate a compression method in a standard, knowledge of the quality of a compressed image is a crucial feature. The lack hereof is one of the biggest disadvantages of VQ methods. Other disadvantages are the weak performance at moderate to high bit rates and the need for storing codebooks. For each different bit rate a different codebook with a different size should be stored. If many different qualities are required, many codebooks should be stored.

These results and arguments show that DCT/VQ image compression is not as powerful as JPEG or JPEG2000 compression. In the next chapter I discuss the possibility of image compression by a combination of wavelet transform and VQ.
Figure 8.4: Lena at 0.20 bpp (a) JPEG (b) JPEG2000 (c) basic DCT/VQ (d) classified DCT/VQ
Chapter 9

DWT/VQ

In the previous chapters the deficiencies of VQ schemes in space and DCT domain have been pointed out. Another possibility for VQ based image compression lies in the combination with discrete wavelet transforms (DWT/VQ). Interestingly, many of the wavelet-based image compression schemes proposed by researchers of the first hour used vector quantization instead of the simpler scalar quantization.

But suddenly, around 1995, research in the field of DWT/VQ ended. Preference was given to scalar quantization after the development of the EZW, and later the SPIHT coder. The current state-of-the-art coder EBCOT was incorporated in JPEG2000 and features scalar quantization as well. This chapter discusses the reasons for this evolution and compares vector and scalar quantization for use with wavelet transforms.

9.1 DWT/VQ

An excellent and very complete survey on DWT/VQ image compression has been written in 1996 by Cosman et al. [7]. At the same time this is the last noteworthy publication on DWT/VQ. The goal of this section is not to repeat the review of this article, but to highlight some of the problems encountered in the design of DWT/VQ systems. In filter terminology DWT/VQ systems are refered to as "VQ of subbands".

The first problem is the choice of the vectors in the subbands. Two categories of solutions can be identified. An intra-band coder constitutes vectors in each subband separately. Fig. 9.1 shows the examples of 16-dimensional vectors in the $HL$ subband and 4-dimensional vectors in the $LH, HL$ subband.
The problem with intraband vectors is that the correlation between the subbands is not exploited. A crossband VQ constitutes vectors with elements from different subbands. Figures 9.2 (a) and (b) show examples of possible crossband vectors.

As explained in section 6.1 VQ is more successful when input vectors and codebook vectors resemble each other. After a DCT transform, $4 \times 4$ vectors all have more or less the same structure: a large DC coefficient and AC components that become smaller for higher frequencies. This makes DCT transformed blocks well suited for vector quantization. For most input vectors, we can expect to find a resembling codebook vector.

For wavelet transformed images this is not the case. The somewhat artificial choice of $k$-dimensional vectors from subbands causes input vectors to be spread out in the $k$-dimensional space. The expected distance between input vector and best matching
codebook vector is large. Otherwise stated, the DWT domain is not well suited for vector quantization.

Another problem, crucial to the success of DWT compression, is bit allocation: if we want to compress an image to a certain bit rate $x$ bpp, then how much bits should be allocated to each subband to result in the best quality image?

For scalar quantization minimization algorithms that solve this problem exist. They tell us exactly how many bits to allocate to each subband and on which elements in each subband to spend these bits.

Bit allocation is problematic in VQ. In crossband VQ bit allocation between subbands is clearly not possible since vectors consists of elements from different subbands. In intraband VQ vectors consist of elements that are of different importance to the image quality. So even if a bit allocation algorithm could find the bit rate (codebook size) to use with each subband, the bits are spent on coding entire vectors and spending bits on unimportant elements in the vectors is inevitable. This is a crucial disadvantage compared to scalar quantization.

Next to these disadvantages characteristic for DWT/VQ, the typical disadvantages of VQ stay present.

- The image quality depends on resemblance to the training vectors and is difficult to predict.
- VQ introduces an a priori loss of quality. Low-bit rate compression is relatively better than high-bit rate compression.
- To compress at different bit-rates, different codebooks need to be stored. In crossband DWT/VQ even more codebooks need to be stored since there is a codebook for each subband.

The results reported for the different methods reviewed in [7] vary between 28 and 32 dB PSNR at bit rates in the order of 0.2-0.3 bpp. This is comparable to the results of DCT/VQ schemes.
9.2 DWT/SQ

The alternative to DWT/VQ is the combination of DWT and scalar quantization. The *Embedded Zerotree Wavelet (EZW)* coding of Shapiro [45] is a landmark DWT/SQ method published in 1993. It provided a breakthrough in wavelet-based image compression, in the sense that it combined state-of-the-art compression with the remarkable flexibility of embedded coding (see section 4.2.1).

To obtain embedded code the property that the Euclidean distance is invariant to orthonormal wavelet transforms is exploited (section 2.12). Mathematically this property tells us that the mean squared error $MSE$ between the pixels' gray-scale values $p_{i,j}$ before, and the pixels gray-scale values $\hat{p}_{i,j}$ after quantization equals the MSE between transform coefficients $c_{i,j}$ before, and transform coefficients $\hat{c}_{i,j}$ after quantization[43].

$$MSE = \frac{1}{N} \sum_{i,j} (\hat{p}_{i,j} - p_{i,j})^2 = \frac{1}{N} \sum_{i,j} (\hat{c}_{i,j} - c_{i,j})^2$$ (9.1)

From this equation it is clear that transmitting the largest $c_{i,j}$ first corresponds to the biggest drop in MSE. Or, to obtain embedded coding the $c_{i,j}$ should be transmitted in descending order and the most significant bits of in the $c_{i,j}$ should be transmitted first. This is done by *bit plane coding*. Fig. 9.3 illustrates the principle of bit plane coding and serves as an example.

The coefficients $c_{i,j}$ are put in descending order (indicated by the sense of the arrows) and the most significant bits in the coefficients $c_{i,j}$ are coded first. If the "length" of the arrows is coded too, only the bits in the blue polygon need to be coded.

The whole procedure is put in an algorithm[43]:

1. Output $n = \lceil \log_2 \max_{(i,j)} \{|c_{i,j}|\} \rceil$. (n is the bit-depth of the coefficients. In the example n=6)

2. Output the number $\mu_n$ = the number of $c_{i,j}$ that satisfy $2^n \leq |c_{i,j}| < 2^{n+1}$. (In the example: $\mu_6 = 2$, $\mu_5 = 3$, $\mu_4 = 5$, $\mu_3 = \mu_2 = \mu_1 = 0$)

3. Output the coordinates $(i, j)$ of each of the $\mu_n$ coefficients that satisfy $2^n \leq |c_{i,j}| < 2^{n+1}$

4. Output the $n$-th most significant bit of all the coefficients with $c_{i,j} \geq 2^{n+1}$, in the same order used to send the coordinates.
5. Code the $\mu_n$ bits of row n-1


The problem of this simple algorithm is the coding of the coordinates $(i,j)$ in step 3. Coding them directly would occupy too much space to yield good compression. Luckily, intelligent but sophisticated algorithms have been developed to efficiently code these coordinates $c_{i,j}$.

They make use of the multiscale form of the DWT transformed image and the spatial correlation between coefficients in different subbands. Correlated coefficients are gathered in structures called *spatial orientation trees*. Fig. 9.4 shows an example (red arrows). To refer to the elements in this tree relative to each other, simple nomenclature from genealogy is often used: we speak about children, parents, descendants, ancestors, ...

To code the coordinates $(i,j)$ of those $c_{i,j}$ that satisfy $2^n \leq |c_{i,j}| < 2^{n+1}$ the concept of *(in)significance* is introduced. A wavelet coefficient $c_{i,j}$ is said to be *insignificant* with respect to a given threshold $2^n$ if $|c_{i,j}| < 2^n$ and *significant* if $|c_{i,j}| \geq 2^n$. The task of step 3 can now be reformulated as coding the coordinates of significant coefficients.

Shapiro’s EZW coder\[45\] achieves efficient coding by assuming that if a wavelet coefficient in a certain subband is insignificant with respect to a treshold, then all wavelet coefficients of the same spatial orientation in higher frequency subbands are insignificant with respect to the same treshold. Empirical evidence shows that this hypothesis is often true.

Fig. 9.4 shows an example. All (in)significance is with respect to the same treshold. If coefficient $A$ is insignificant, then probably all coefficients in group B are insignificant.
If coefficient $X$ is insignificant, then probably all coefficients in groups $Y$ and $Z$ are insignificant.

A coefficient is said to be part of a zerotree if itself and all of its descendants are insignificant. The zerotree root is the only element of each zerotree who's parent is insignificant. An isolated zero is an insignificant coefficient not part of a zerotree (at least one of its descendants is significant).

Fig. 9.4 illustrates these concepts. Coefficients $P, Q$ and $V$ are significant. Coefficients $X$ and $A$ are zerotree roots since all of their descendents and themself are insignificant. Coefficient $U$ is an isolated zero since its descendant $V$ is significant.

The EZW coder scans through the subbands from the low to the high frequency subbands and categorizes coefficients into 4 categories: positive and significant, negative and significant, isolated zero or zerotree root. Elements of a zerotree, except for the zerotree root, do not need to be coded since their insignificance is known from coding their zerotree root.

Under the assumption that there are many zerotrees this leads to an important reduction of the coefficients to be coded. In fig. 9.4, for example, the coefficients in $Y, Z$ and $B$ do not need to be coded because $X$ and $A$ are coded as a zerotree root.
Further compression is obtained by incorporating adaptive arithmetic coding. For a more rigorous discussion and more details I refer to the original article [45].

Set Partitioning in Hierarchical Trees (SPIHT) is technique related to EZW. It is more general and more powerful. Its performance is better. For a description I refer to the original article of Said and Pearlman [43].

Embedded Block Coding with Optimized Truncation of the embedded bit stream (EBCOT) [47] is yet another technique, even more sophisticated and powerful than SPIHT. It forms the core of the JPEG2000 standard (section 4.2.1).
Chapter 10

Conclusions

10.1 Summary

In this thesis the transform-scalar quantization-entropy coding structure of the image transform paradigm was explained.

The transform step decorrelates the image data, compacts its energy in a small number of coefficients and puts it in a well-suited form for quantization. The discrete cosine transform achieves these 3 goals, but requires the image to be split up in blocks which causes block artefacts at low bit rates. The discrete wavelet transform also achieves the transform goals but operates on the whole image and eliminates block artefacts. The DWT of an image is implemented by a two-dimensional extension of a cascade of two-channel subband filter banks. The 9/7 CDF wavelet is a good wavelet for image compression because of its length, symmetry and close to orthogonal property.

The JPEG standard features a DCT on $8 \times 8$ blocks, scalar quantization by a predefined quantization matrix, DPCM coding of DC-coefficients and zig-zag entropy coding of AC-coefficients. The JPEG2000 standards features a 9/7 CDF DWT and a sophisticated scalar quantization/entropy coding system called EBCOT. The performance of JPEG2000 is around 20% better than JPEG for moderate-good quality images. For lower bit rate compression the difference becomes larger. It also yield embedded code and additional flexibility.

Vector quantization is an alternative lossy image compression technique. It tries to exploit correlation between groups of pixels (vectors) by substituting each vector by a corresponding vector from a codebook. This codebook is found by applying MSE optimization
algorithms such as \textit{k}-means or the Linde-Buzo-Gray algorithm to a set of training vectors. Competitive neural networks provide an alternative means of finding a codebook. Self-organizing maps introduce an ordering between the codebook vectors, but do not minimize the MSE. Growing neural gas introduce an ordering too, but are more flexible and are better at approximating discrete distributions.

Vector quantization has some major disadvantages:

- Codebook storage. To compress at different bit-rates, different codebooks need to be stored.

- Unconsistent quality. The image quality depends on its resemblance to the training vectors and is difficult to predict.

- A priori loss of quality due to the use of a codebook. Low-bit rate compression is relatively better than high-bit rate compression.

This last disadvantage is problematic. The block division to form vectors causes strong blocking artefacts and a staircase effect at edges. Classified VQ and finite-state VQ are advanced architectures for VQ that try to solve this problem.

A better solution is to precede VQ by an image transform. This results in a structure transform - VQ - entropy coding similar to the image transform paradigm on which JPEG and JPEG2000 are founded.

I showed that a DCT/VQ scheme can be improved by ordering the codebook vectors by their first element - equal to the average value - and applying differential entropy coding. At moderate-high bit rates this method is clearly outperformed by JPEG, but at low bit rates it results in better quality and lower PSNR than JPEG. However, this quality gain is too low and the other disadvantages of VQ are far too important to recommend the method.

I showed that classified DCT/VQ yields no improvement compared to ordinary DCT/VQ.

I showed that self-organizing maps, contrary to what other authors claimed, do not add improvement compared to basic VQ. In the space domain, SOM are confronted with the same problems as VQ and quality is unacceptably low. The topological ordering of a SOM, exploited in a DCT/SOM scheme to add additional compression after differential entropy coding, is not as good as the ordering I proposed for DCT/VQ.
The combination of VQ and discrete wavelet transform is possible too. Difficulties are the choice of the vectors and bit allocation. The reported performances are similar to DCT/VQ.

Schemes featuring discrete wavelet transform and scalar quantization yield the best results. The EZW coder was a breakthrough coder. The SPIHT and EBCOT coders further improved performance and flexibility.

10.2 Future Work

The gap between the best VQ based methods and the JPEG2000 standard is very large. Even if VQ based methods could be improved, it would be unrealistic to expect them to attain performance competitive with JPEG2000. Other techniques once considered promising such as genetic algorithm or fractal image compression seem to be at a dead end too.

The safest bet for improvements still lies in the wavelet transform-scalar quantization-entropy coding structure. A nonseparable two-dimensional discrete wavelet transform is an interesting alternative for the cascade of two one-dimensional discrete wavelet transforms that is currently used. Which would be the best two-dimensional wavelet to use?

Other improvements could be incorporated in JPEG2000’s EBCOT coder. The coder can be optimized to minimize perceptual criteria instead of the MSE. Research is needed to find the best perceptual criterion in line with the current knowledge of the human visual system.

I believe that for such a perceptual criterion to find recognition it should be assessed by a worldwide accepted perceptual quality metric. Today all researchers present their results in the peak signal-to-noise ratio and this metric is not fully in accordance with the perceived quality (e.g. blocking artefacts are not detected). However, it is not an easy task to map such subjective concept as image quality into an objective value. Work needs to be done in this area.

Video compression is a field related to image compression. Broadband internet connections are widespread now and this opens up tremendous opportunities. Today’s popular MPEG standards are based on a DCT transform. Much research is being done nowadays on video compression methods based on discrete wavelet transforms, but standardization is not yet for tomorrow. Much research remains to do.
Bibliography


