Joonas Lehtinen

Coding of Wavelet-Transformed Images

Turku Centre *for* Computer Science

TUCS Dissertations
No 62, June 2005

CODING OF WAVELET-TRANSFORMED IMAGES

by

Joonas Lehtinen

ACADEMIC DISSERTATION

To be presented, with the permission of the Faculty of Mathematics and Natural Sciences of the University of Turku, for public criticism in the Auditorium of the Department of Information Technology on July 1, 2005, at 12 noon.

University of Turku
Department of Information Technology
Turku, Finland
2005

Supervised by

Professor Olli Nevalainen, Ph.D.
Department of Information Technology
University of Turku

Reviewed by

Professor Jussi Parkkinen, Ph.D.
Department of Computer Science
University of Joensuu

and

Professor Ioan Tabus, Ph.D.
Signal Processing Laboratory
Tampere University of Technology

ISBN 952-12-1568-2
ISSN 1239-1883
Painosalama Oy
Turku, Finland

Abstract

Compression methods are widely used for reducing storage and enhancing transfer of digital images. By selectively discarding visually subtle details from images, it is possible to represent images with only a fraction of the bits required for the uncompressed images. The best lossy image compression methods currently used are based on *quantization*, *modeling* and *entropy coding* of transformed images.

This thesis studies quantization and modeling of *wavelet-transformed natural images*. Usage of different quantization levels for the different regions of the image is discussed and a new *variable quality image compression* method is introduced. The benefits of the variable quality image coding are demonstrated for coding of mammography images. The quantization of the transform coefficients is controlled in most of the lossy image coding algorithms by setting a limit to the size of the compressed image or by directly defining the magnitude of the quantifier. It is shown here how the *distortion in the decompressed image can be used as the quantization criterion* and a new image coding algorithm that implements this criterion is introduced.

While a wavelet transformed image is encoded, both the coder and decoder know the values of the already encoded coefficients. The thesis studies how this coding context can be used for compression. It is shown that conventional prediction methods and scalar quantization can be used for modeling coefficients and introduce a new coding algorithm that *predicts the number of significant bits in the coefficients from their context*. A general method of adaptively modeling probability distributions of encoded coefficients from a property vector calculated from the coefficient context is given. This method is based on *vector quantization of the property vectors* and achieves excellent compression performance. Forming high quality code books for vector quantization is studied. *Self-adaptive genetic algorithms are used for the optimization of the code books* and a new model for *parallelization* of the algorithm is introduced. The model allows efficient distribution of the optimization problem to multiple networked processors and flexible reconfiguration of the network topology.

Acknowledgements

This work has been carried out at the Turku Centre for Computer Science. As well as funding for research, the Centre has provided an inspiring environment to work in. I am grateful for the flexibility of the graduate school, which has made it possible for me to combine research with working at IT Mill Ltd. I would especially like to thank Prof. Timo Järvi for his support and Prof. Ralf Back for spurring me on with this work.

Above all, I thank my instructor and collaborator Prof. Olli Nevalainen for his support and guidance through my studies, from the beginning to the completion of this dissertation. Even in the times when most of my time was spent to other projects he has persistently encouraged me to continue and has offered his help throughout the process. I am also very grateful for Prof. Jukka Teuhola for his comments and advice when writing the introduction for this dissertation as well as Prof. Ioan Tabus and Jussi Parkkinen for their comments and corrections.

I would like to thank my collegues, Antero Järvi and Juha Kivijärvi for collaboration and many interesting discussions; you have both shown me a good example on how algorithm research should be carried out and how much attention one should give to details.

Finally, I would to thank my parents and grandparents for believing me on this project, even though I have changed my estimate on the schedule many times over the years.

Contents

1 Introduction **1**
- 1.1 Motivation . . . 1
- 1.2 Introduction to transform coding . . . 2
  - 1.2.1 Image representation . . . 2
  - 1.2.2 Image compression . . . 4
  - 1.2.3 Entropy coding . . . 6
  - 1.2.4 Image quality metrics . . . 9
  - 1.2.5 Standards . . . 10
- 1.3 Outline of the thesis . . . 11

2 Wavelet transforms **13**
- 2.1 Basis for linear expansions . . . 13
- 2.2 Wavelet transform . . . 14
- 2.3 Boundary handling for finite signals . . . 19
- 2.4 Two-dimensional signal decomposition . . . 21

3 Transform coding **25**
- 3.1 Embedded Zerotree Wavelet coding . . . 25
  - 3.1.1 Significance maps . . . 26
  - 3.1.2 Coding with zerotrees . . . 27
- 3.2 Set partitioning in hierarchical trees . . . 28
  - 3.2.1 Coding bitplanes by sorting . . . 28
  - 3.2.2 List based sorting algorithm . . . 30
- 3.3 Context based coding . . . 31
  - 3.3.1 Context classification . . . 31
  - 3.3.2 Vector quantization of the context space . . . 33
- 3.4 Code book generation for vector quantization . . . 33
  - 3.4.1 Clustering by k-means . . . 34
  - 3.4.2 Genetic algorithms . . . 35

4 Summary of publications **37**
4.1 Variable quality image compression system based on SPIHT . . . 38
4.2 Distortion limited wavelet image codec 38
4.3 Predictive depth coding of wavelet transformed images 39
4.4 Clustering context properties of wavelet coefficients in automatic modelling and image coding 40
4.5 Clustering by a parallel self-adaptive genetic algorithm 41
4.6 Performance comparison 42

5 Conclusions **49**

Bibliography **51**

Publication reprints **57**

Publication errata **135**

Chapter 1

Introduction

1.1 Motivation

Digital images and video are rapidly replacing most analogue imaging technologies in all the phases of the imaging: production, transfer, consumption and storage. Even though the capacity of computers to store and transfer data has been exponentially growing as predicted by Moore's law [1], in most imaging applications needs for advanced image compression techniques are still increasing. Reasons for utilization of new compression technology vary from the possibility of using the computing capacity in enabling higher quality of images to the possibility of adding new application specific features into the compression techniques.

The most obvious reason for using compression when storing and transferring digital images and video is that the storage and bandwidth requirements for a compressed image data might be only a fraction of the requirements for the original contents. Image compression has been one of the key technologies that have enabled digital television, distribution of video over Internet, high resolution digital cameras and digital archiving of medical images in hospitals. The challenges of image and video compression research are not limited to absolute storage and bandwidth savings. Often, modest computational complexity of the coding and decoding might be even more important for practical applications. While the image resolution grows, memory requirements of the compression algorithms might grow in such a way that it would limit the usage of compression techniques in embedded applications, such as high resolution printers and scanners. Many application specific features for compression algorithms are also currently inspected to make the compression algorithms to

[1] In 1965 Gordon Moore, co-founder of Intel, predicted that the number of transistors per square inch on integrated circuits will double yearly. The current expected rate of doubling the transistor density is once in 18 months.

perform some tasks better than a general purpose algorithm would do or to guarantee that the quality of the images matches the required standards.

From the above it is obvious that research of efficient compression methods is an important issue in the all fields that require storage and transfer of still images and video. In the present study we focus on compression of natural still images. The aim is to search for more efficient compression methods as well as provide more flexibility for selection of compression parameters. We restrict ourselves to lossy wavelet transform coding techniques [62] due to their excellent compression performance. For other image compression techniques, see [5, 34, 55].

1.2 Introduction to transform coding

In this section a brief overview to transform coding of natural images is given. All the steps of the process are discussed, but only minimal details are given. An introduction to wavelet transforms is later given in Chapter 2 and techniques for coding the transformed data are discussed in Chapter 3.

1.2.1 Image representation

Spatial representation of a digital image can be seen as a matrix of pixels (picture elements) where each pixel represents the average colour of the image on the area the pixel covers. In most applications, pixels are rectangular and evenly sized in all regions of the image. The *spatial resolution* of the image is defined as the number of pixels per length unit, for example 150 dots per inch (DPI). Terms *image resolution* and *image size* are often used somewhat erroneously as synonyms to represent dimensions of the image in pixels.

In black and white or grayscale images, pixel values represent the *luminance* (brightness) of the pixels. Each luminance value is represented using a fixed number of bits, which is often referred as the *luminance resolution*. This amounts typically 1, 2, 4, 8, 12 or 16 bits per pixel (BPP). On colour images, each pixel is represented by a set of values representing different components of the used color system. The commonly used color systems [22] include RGB, where the values represent the intensity of red, green and blue light; YUV where values represent luminance and 2D coordinates on the *chrominance* (color) plane; and CMYK, a subtractive color model often used in printing. Typical *color resolutions* include 15, 16, 24 and 36 BPP, where each color component is expressed by 5, 6, 8 or 12 BPP. In image compression systems, different color components are often compressed separately as different grayscale images and they can be represented with different spatial resolutions. Because any grayscale image compression algorithm can be easily extended to include compression of color images, this dissertation does not discuss about compression

of color images.

Many signal processing applications require signal frequency information, which has lead to the development of different *frequency representations* for the signal. When in the spatial domain, the image luminance is given as a function of location, whereas in the frequency domain representation, the amplitudes of the different *frequency components* in the image are represented as a function of these frequencies. Thus the discrete image signal in spatial domain can be represented as a linear combination of all its frequency components. For example in the discrete Fourier transform [4], a one-dimensional discrete signal $x[n]$ ($n \in \{0, 1, \ldots, N-1\}$) can be represented as a linear combination of the frequency components defined by Fourier series:

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] W_N^{-nk}, \tag{1.1}$$

where $X[k]$ ($k \in \{0, 1, \ldots, N-1\}$) is the frequency representation of the signal and $W_N = e^{-2\pi\sqrt{-1}/N}$.

To produce the frequency representation of non-continuous signals, such as images, the signal is normally divided spatially to *windows* of equal size. Each window is then individually transformed to the frequency domain, most often using the fast Fourier transform (FFT) [5] or the discrete cosine transform (DCT) [1, 25]. The typical size of the window in most DCT-based image and video compression techniques is 8×8 pixels. Because each window is processed separately, some image compression algorithms might fail to preserve signal continuity on window borders leading to visible blocking artifacts.

The windowed frequency representation uses fixed sized spatial regions when analyzing the signal for different frequency components. In natural images, high frequency details tend to be spatially smaller than the low frequency details, which makes the analysis of different frequencies with equally sized filters inefficient. *Multi-resolution representations* [44, 48] combine some features of frequency representation and spatial representation by analyzing spatially smaller regions of the image for high frequency components and larger regions for low frequency components.

The multi-resolution representation is usually formed by analyzing the signal with *wavelet functions* [23]. Wavelet functions are localized functions that have value of zero or very near to zero outside a bounded domain and average of zero. There exists many kinds of wavelet functions but unlike sine waves, most wavelets are irregular and asymmetric.

Representation of a *video* builds on the image representation: the most trivial representation of a video sequence is just concatenation of the individual frame images in the video. Practical video formats might also include multiple audio tracks and subtitles that are synchronized with the video. Video com-

pression algorithms commonly include features that benefit from the similarity of consecutive frames in the video by predicting some of the frames from the neighboring frames and only storing the prediction errors [58]. Even in those techniques, some of the frames are stored as separate images to make random access for the video stream possible. Moreover, fragments of the predicted frames or the prediction errors are often stored using conventional still image compression algorithms. The field of video compression can be seen as an extension of the still image compression. This dissertation does not include any video compression techniques, but we recognize that the algorithms presented for still image compression can be applied to video compression as well.

1.2.2 Image compression

The basic goal of the image compression is to find such a representation for an image that only a minimal number of bits is used. This both allows one to store more image data on a limited storage space as well as makes it possible to transfer images faster over a channel with limited bandwidth. The compression efficiency can be measured with the *compression ratio* $R = S_o/S_c$, where S_c is the size of the compressed representation of the image (in bits) and $S_o = WHB$, where W, H and B are the width, height and luminance resolution of the image (in bits). An even more used metric is bits per pixel (BPP), which is simply defined as $B_r = B/R$.

The image compression algorithms can be divided into two classes: *lossless* and *lossy*. If the image compression is completely reversible, it is said to be lossless. If the decompressed image is only an approximation of the original image, the compression is said to be lossy. Lossless image compression techniques achieve generally compression ratios in range 1 - 5 on natural images [34, 13], while lossy methods typically achieve several times better compression ratios. For example the compression ratio for a comic image on the page 68 is 1.8 when using lossless GIF [55] compression. Lossy compression techniques JPEG [51] and SPIHT[54] achieve compression ratios of 8 and 16 respectively with good image quality.

Most lossy image compression methods are based on some kind of coding of the transformed and quantized (approximated) image representation. Generic compression and decompression processes are demonstrated in Figure 1.1. First, the original spatial image is transformed to frequency or multi-resolution representation with a transform T. A part of the information is lost when quantizing the coefficients in the transformed image representation with a quantizer Q in. The probability model of the quantized coefficients is created with a modeling algorithm M. The probability model approximates the frequencies of the coefficients in the coded image in a such way that the model can be coded with as few bits as possible and at the same time is accurate enough to be used in

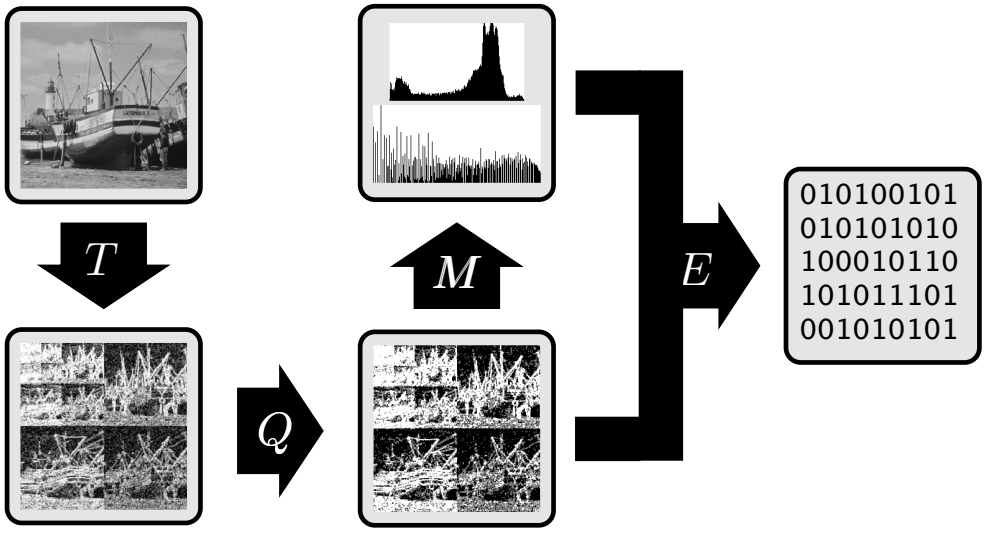


Figure 1.1: Structure of a general image compression algorithm. T represents an image transform, Q a quantizer, M a modeling method and E a entropy coder.

encoding of the coefficients. Finally, the coefficients are coded with an entropy coder E using the created model. The purpose of the entropy encoder is to code the quantized coefficients using as few bits as possible. Any of the four phases T, Q, M, E can be combined together to achieve some application specific features or better compression performance. It is also possible to stream data through all the compression phases to save memory needed for buffering intermediate results [10].

Decoding of the image is done in the reverse order to the coding process. First, the entropy coded bits are decoded back to quantized coefficients using the same model that was used in the encoding phase. The model is either saved as (entropy coded) side information or dynamically generated from the decoded quantized coefficients while decoding. The quantized coefficients are then scaled to represent the original transformed image and finally an inverse transformation is used to get a decoded spatial domain image as a result. As with encoding, all the phases of the decoding might be combined and the data can be streamed through the phases. In symmetric algorithms decoding is computationally equivalent to coding. In asymmetric algorithms modeling cost required in coding might be considerably higher than in decoding.

The purpose of quantization is to remove unimportant details of the image in such a way that the compression ratio is optimal for the selected image quality. Scalar quantizers belong to the most simple type of quantizers: all transformed

image coefficients are scaled with some constant q to get the scaled coefficients $\lfloor qc_i \rfloor$, where c_i are the original coefficients.

Vector quantization is a more general quantization method. There the signal x of length NK is divided to N K-sized vectors $x_i = [x(iK), x(iK + 1), \ldots, x(iK + K - 1)]^\top$ and the quantizer tries to find similar code vectors $c_j \in C$ to represent each x_i. The quantized image can be then represented with just N indexes that identify the code vectors from C. The set C is called code book. Construction of a code book is a hard problem, which limits the usage of the vector quantization in some applications. Moreover, both the coder and the decoder must use the same code book. Thus, either the code book must be submitted alongside the code vector indexes or a static code book must be used for all images.

Most transform coding techniques could be turned into lossless methods by skipping the quantization step, provided that the transform itself is lossless. Still, many lossless image compression techniques [63, 29] do not rely on transforms, but code the image directly from its spatial representation. These kinds of lossless coding algorithms rely on *prediction coding*, where the luminance of each image pixel is predicted from the values already known by the decoder. This allows both the coder and the decoder to make the same prediction for the pixel. If the prediction is good, coding only the prediction error can provide an efficient coding system.

1.2.3 Entropy coding

Entropy coding is a mapping C from an m-sized alphabet $X = \{x_i | 0 \leq i < m\}$ to a set of unique codewords $\{c_i = C(x_i) | 0 \leq i < m\}$, with the purpose of minimizing the size of the coded message. If the probability of symbol x_i in the alphabet is $p(x_i)$, the *expected length* (in bits) of the message composed of n symbols coded with the mapping C is:

$$R = n \sum_i p(x_i) l(C(x_i)), \tag{1.2}$$

where $l(c_i)$ is the length (in bits) of the codeword c_i. It is required that the sequence of the codewords is *uniquely decodable*: mapping C must be reversible and no codeword is allowed to be a prefix to another codeword. *Entropy* [15] of an information source defines a lower bound for the message length

$$H = -n \sum_i p(x_i) log_2(p(x_i)) \tag{1.3}$$

when the $p(x_i)n$ is the frequency of symbol x_i in the message.

Huffman coding [30, 37] provides a simple way of building binary codes with coding efficiency near the optimum as defined by the entropy. The code

"ACDCCBCA" → 00101111010100

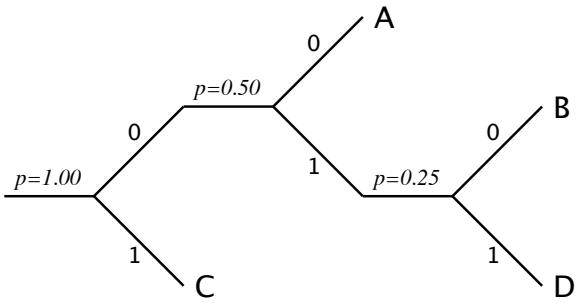


Figure 1.2: Huffman coding of message "ACDCCBCA" and corresponding decision tree for decoding.

is built iteratively by selecting two symbols with the lowest probabilities and assigning them codes 0 and 1 for selecting between them. Then the two symbols are removed from the alphabet and replaced by one symbol the probability of which equals to the sum of the probabilities of the removed symbols. The process is then iterated until all the symbols have been assigned codes. This happens when the alphabet has reduced to a single symbol. The result of the process can be visualized with a decision tree that is used in decoding by reading the coded message one bit at a time and following the decisions on the tree as illustrated in the Figure 1.2.

One limitation of the Huffman code and all other similar binary codes is that at least one bit is used to represent a symbol, which makes the coding of alphabets with very skewed probability distributions inefficient. More generally, only symbols x_i for which

$$log_2(1/p(x_i)) = \lfloor log_2(1/p(x_i)) \rfloor \tag{1.4}$$

can have codes with optimal length in those systems. One way to overcome these limitations is to transform the original alphabet to another one where equation (1.4) holds better for the alphabet and use the new alphabet for generating Huffman-codes. This can be done by combining the letters of the original alphabet to longer words and adding the new words as letters to the new alphabet.

Arithmetic coding [65] provides optimal compression performance as defined by the entropy equation (1.3). The idea of the arithmetic coding is to represent the whole message with only one codeword so that its probability equals to the combined probability of all the symbols in the message. The coding process is illustrated in Figure 1.3. The length of the black region equals to the

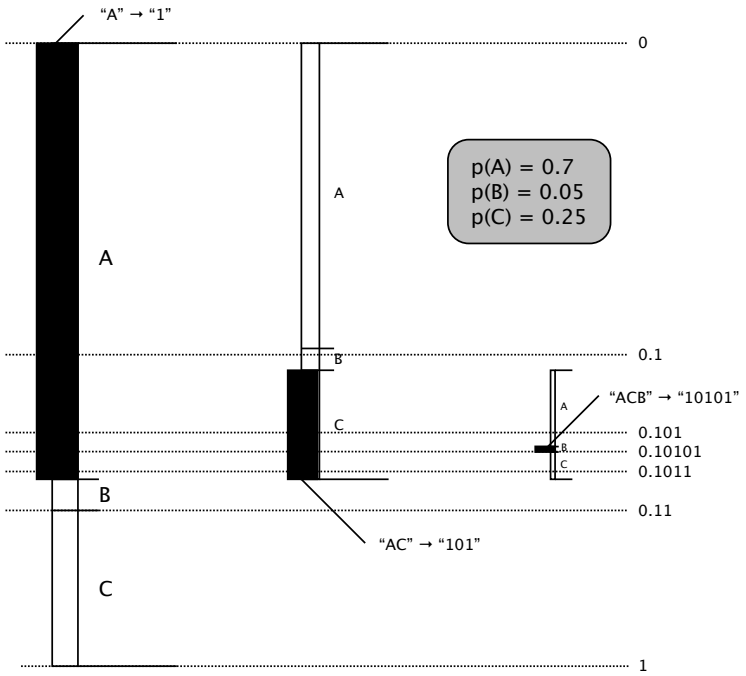


Figure 1.3: Arithmetic coding process of message "ACB" to binary string "10101".

probability of the corresponding message. The message is finally coded as the shortest binary number that exists on the probability range corresponding to the message. For example, in the case of Figure 1.3, the probability of message "ACB" is $0.7 * 0.25 * 0.05 = 0.00875$ and thus the corresponding entropy for the message is 6.837 bits. In the example the message is coded as "10101" using only five bits, but on average the code length equals to the entropy. In practical implementations of arithmetic coding the coder receives the message as a stream of symbols and updates the range for each symbol. In order to store the limits of the range with practical accuracy, the coder re-scales the range every time it has reduced to short enough and sends the bits corresponding to scalings to the output code stream.

In many applications, the probability distribution of the alphabet varies highly in different parts of the coded message. For example in the message "AAAAAAABBBBBBB", the probabilities of A and B are both 0.5, but still it might be possible to code the message with less than one bit per symbol if the use of different probability distributions would be allowed in the different parts of the message. Because both the coder and the decoder must use the

same probability distribution for the alphabet, sending a distribution for a large alphabet to the decoder might be impractical in some applications. For these reasons, the use of a *dynamically varying distribution* for the alphabet is often preferred to static probability model. This distribution is adaptively learned in the course of the coding process. Supporting such a dynamic probability model with Huffman coding is rather impractical, because the coding tables and decoding trees must be rebuilt at each update of the probability model. The arithmetic coding supports the use of a dynamic probability model very well, because the coder and decoder only have to know the cumulative probability for each symbol in the alphabet in order to be able to update the range limits. Speed-optimized versions of the arithmetic coding with dynamic probability distributions have been developed based on state automata, where each state represents one distribution. One of the most popular is the binary arithmetic coder named QM-coder that is currently used in the JPEG image compression standard [51].

1.2.4 Image quality metrics

In lossy image compression the challenge is to represent the image using the minimal number of bits and at the same time to preserve the image quality as well as possible. The measurement of the compressed image size is trivial, but there exists no universal quality metric because the definition of quality depends on the application. In special fields, such as medical imaging, the image quality is measured as the impact it has to the diagnosis based on the image. In most medical applications the compression is required to be *diagnostically lossless*: no such information is lost that the diagnosis made from the image could be disturbed. The goal of lossy image compression of natural images is often to preserve the overall *visual quality* of the image as observed by humans. Unfortunately this definition of quality does not state the exact observation conditions for the image and on the other hand there exists no widely adopted procedure to easily do the required quantitative testing [45]. Objective quality measurement is an active field of research and new methods [53, 38] are developed for automated quality assessment. Such methods can provide a new basis for development of more efficient image and video compression methods in the future.

In the image compression research literature the most popular distortion metric is the *mean square error* (MSE) calculated as:

$$d_{\mathrm{MSE}}(u, \tilde{u}) = \frac{1}{HW} \sum_{x=0}^{W-1} \sum_{y=0}^{H-1} [u(x, y) - \tilde{u}(x, y)]^2, \tag{1.5}$$

where the functions u and $\tilde{u}$ represent the original and decompressed $H \times W$ sized images, correspondingly. The MSE is commonly normalized to the

dynamic range of the image and expressed on the logarithmic scale as the *peak signal to noise ratio* (PSNR), which is defined as follows:

$$\mathrm{PSNR}(u, \tilde{u}) = 10 \log_{10} \frac{(2^{R_{\mathrm{lum}}} - 1)^2}{d_{\mathrm{MSE}}(u, \tilde{u})}, \tag{1.6}$$

where R_{lum} is the luminance resolution of the image.

More advanced distortion metrics have been developed that model the human visual system better than simple MSE [67, 8, 9]. When advanced distortion metrics are used in the design of image compression algorithms as their optimization criteria, they can have tremendous impact on the visual quality of the decompressed images. Unfortunately, many advanced distortion metrics are computationally unsuitable for image compression. Furthermore, PSNR is widely used in image compression literature and thus using it as optimization criteria for coding makes performance comparison of coding algorithms easier.

1.2.5 Standards

Most practical applications of image and video compression techniques are based on widely accepted standards. New standards are usually created in a process where the best available ideas are collected from literature and adapted to form the base for the new standard. In the context of image and video compression, the most recognized standard organizations are *Joint Photographic Experts Group* (JPEG), *Joint Bi-level Image experts Group* JBIG and *Moving Picture Experts Group* (MPEG).

The *ISO/IEC 10918-1* standard: "Digital compression and coding of continuous-tone still images" [51], commonly called "*JPEG*", is the most widely adopted image compression standard. It is based on the quantization of DCT-transformed images with 8×8 transform block size. The quantized image coefficients are coded either using Huffman coding or QM-coder. The standard supports color images by compressing different color components of YUV color space separately and by scaling down the chrominance components spatially by the factor of two in both dimensions.

The new standard *ISO/IEC 15444-1:2000* developed by JPEG called "*JPEG 2000* image coding system" [60, 46] is a complete revision of the previous JPEG standard, and provides improved compression performance and an extensive list of advanced compression features. The new standard is based on wavelet image coding algorithm similar to the coding algorithms discussed in the chapter 3. JPEG 2000 is targeted to a wide range of image compression applications, including general still image coding, video coding, variable quality coding, volumetric imaging, document imaging and wireless applications. The core part of the standard was published in March 2002, but the other parts of the standard are still under development [24].

1.3 Outline of the thesis

This work is based on the following hypotheses: 1) Variable quality coding of wavelet transformed images can provide considerably better compression performance than conventional methods; 2) MSE can be used as a control criterion for wavelet image compression; 3) Both scalar and vector quantization of context information derived from wavelet coefficient neighbors can be used for efficient compression; and 4) Parallel processing can be effectively used in generation of code books for vector quantization.

In this thesis we introduce four new algorithms for coding of wavelet transformed natural images. We show how the image quality can be controlled in embedded coding of images and introduce new effective implementations for zerotree based compression algorithms. We also show that it is possible to achieve good compression performance by a simple context based coefficient prediction method and how clustering methods can be applied efficiently for context based classification and prediction. In addition, we present a new state of the art clustering algorithm that supports parallel processing.

This thesis is divided into two parts: introductive summary of the research and reprints of the publications. The idea is to first provide sufficient background information in the introductory part and then present the actual compression algorithms and the results on the reprints.

This chapter has given a brief introduction to the basic concepts of transform coding of grayscale images as well as motivation and outline for the thesis. The next chapter explains the wavelet transforms used in the algorithms presented. The third chapter provides a detailed overview on the transform coding process and discusses the algorithmic techniques used as basis for the work presented in the reprints. The fourth chapter provides an overview of the publications and finally the results are summarized in the chapter five.

Chapter 2

Wavelet transforms

The purpose of this chapter is to give the reader a quick summary on the wavelet transforms used by the algorithms presented in this thesis. The chapter is not a general introduction to the theory behind linear expansions and wavelet functions, instead the goal is only to summarize the concepts that are necessary for understanding the structure of the transformed image data and algorithms for coding it. While the material of this section has been collected from various sources using diverse notation conventions, the representation has been unified in order to be able to provide a consistent summary without going into unnecessary details. For more detail and theory behind the concepts presented, references to literature are given.

2.1 Basis for linear expansions

This section introduces the basic linear expansion of discrete signals that is needed for computing the wavelet transforms. Furthermore some properties of the signals and expansions are defined here because they are needed for explaining the wavelet transforms later in this chapter.

Hilbert space of square-summable sequences $l_2(\mathcal{Z}) \subset \mathcal{C}^\infty$ is a set of all infinite-dimensional vectors x, that satisfy the norm constraint $\langle x, x \rangle < \infty$, where notation $\langle u, v \rangle$ stands the dot product defined as $\sum_{n \in \mathcal{Z}} u[n]v[n]$. A *linear expansion* [62] of a discrete signal $x \in l_2(\mathcal{Z})$ can be formulated as

$$x[n] = \sum_{k \in \mathcal{Z}} X[k]\varphi_k[n], \tag{2.1}$$

where the set of vectors $\{\varphi_k | \varphi_k \in l_2(\mathcal{Z}) \wedge k \in \mathcal{Z}\}$ is called the basis of the expansion and vector X contains the expansion coefficients corresponding to the original signal. It is said that the basis is *complete* for the space S if all signals $x \in S$ can be expanded as defined by equation (2.1). It can be shown

that for each complete basis there exists a dual set $\{\tilde{\varphi}_k | \tilde{\varphi}_k \in l_2(\mathcal{Z}) \wedge k \in \mathcal{Z}\}$ that can be used to compute the transformed signal $X \in l_2(\mathcal{Z})$ as

$$X[k] = \sum_{n \in \mathcal{Z}} \tilde{\varphi}_k[n] x[n]. \tag{2.2}$$

It is said the basis $\{\varphi_k\}$ is *orthogonal* if for all $i, j (i \neq j)$ it holds: $\langle \varphi_i, \varphi_j \rangle = 0$. Furthermore, if the *norm* $\sqrt{\langle \varphi_k, \varphi_k \rangle} = 1$ for all $k \in \mathcal{Z}$ and $\{\varphi_k\}$ is an orthogonal basis, it is called *orthonormal*. An important property of the orthonormal basis is that for all $k \in \mathcal{Z} : \varphi_k = \tilde{\varphi}_k$ and the energy of the signal x is conserved in the transform defined in equation (2.2) :

$$\langle x, x \rangle = \langle X, X \rangle. \tag{2.3}$$

In the field of signal compression another commonly used type of basis is *biorthogonal basis*, where the set $\{\varphi_k\}$ is complete and linearly independent, but is not orthonormal. A biorthogonal basis and its dual $\{\tilde{\varphi}_k\}$ satisfy

$$\forall i, j \in \mathcal{Z} : \langle \varphi_i, \tilde{\varphi}_j \rangle = \delta[i - j], \tag{2.4}$$

where Dirac's delta function $\delta[i]$ is defined as $\delta[0] = 1$ and for all $i \neq 0 : \delta[i] = 0$. In biorthogonal transforms, the conservation of energy is defined as

$$\langle x, x \rangle = \langle X, \tilde{X} \rangle, \tag{2.5}$$

where $\tilde{X}[k] = \langle \varphi_k, x \rangle$ and $X[k] = \langle \tilde{\varphi}_k, x \rangle$.

In most signal compression applications, the dimensionality d of signal vector $x \in S \subset l_2(\mathcal{Z})$ can be large, because the dimension corresponds directly to the number of samples in the signal. If each sample of the signal x is represented with b bits, the size $|S|$ of the space is 2^{bd}. When a transformation is used to map all signals $x \rightarrow X \in \tilde{S}$ and if no information is lost in the transform, $|S| \leq |\tilde{S}|$. If the number of bits needed to represent each element of the transformed signal X is B, the dimension of the transformed signal is $D \geq \log_{2^B} |S| = db/B$. From equation (2.1) it can be seen that the number of base vectors φ_k equals to D each having d elements. Because of the large number of base vectors, the transformation is practical only if the base vectors φ_k can be easily computed during the transform process.

2.2 Wavelet transform

This section briefly introduces wavelet-functions and show how they are used for constructing the transforms for infinite discrete 1D-signals. First an example of a simple transform is given and later it is shown how filter banks can be used for transforming signals iteratively. Finally the wavelet filters, used by the

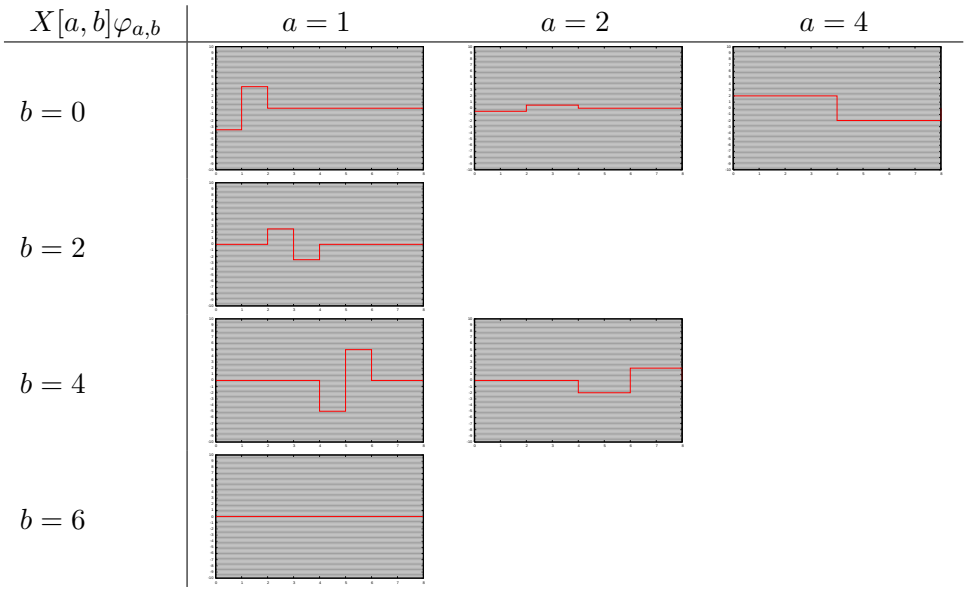


Figure 2.1: Haar expansion of signal $x = (-2, 5, 5, 0, -9, 1)^\top$ to different components.

new coding algorithms described in this work, are given. The complete wavelet transforms can implemented when the filtering equations and the filter coefficients are combined with the details on boundary handling and two-dimensional extensions described later in this chapter.

Wavelets [62, 7, 17] are a class of functions that can be used as a basis for a linear expansion with good localization both in space and scale. A discrete wavelet basis $\{\varphi_{a,b} \in l_2(\mathcal{Z})\}$ can be constructed from a finite *mother wavelet* function $\varphi \in l_2(\mathcal{Z})$ by modulation (a) and translation (b) as

$$\varphi_{a,b}[n] = \frac{1}{\sqrt{a}}\varphi\left[\left\lfloor\frac{n-b}{a}\right\rfloor\right]. \tag{2.6}$$

The *Haar* wavelet basis can be defined with equation (2.6) where the mother wavelet is

$$\varphi[n] = \begin{cases} \frac{1}{\sqrt{2}} & \text{if } n = 0 \\ -\frac{1}{\sqrt{2}} & \text{if } n = 1 \\ 0 & \text{otherwise.} \end{cases} \tag{2.7}$$

The orthonormal basis defined by the mother function φ is $\{\varphi_{2^i,j2^i} | i, j \in \mathcal{Z}\}$. It can be shown that this basis spans the space $S \subset l_2(\mathcal{Z})$ of signals $x \in S$ for which $\sum_{n\in\mathcal{Z}} x[n] = 0$. An example of linear expansion for signal $x = (-2, 5, 5, 0, -9, 1)^\top$ is shown in Figures 2.1 and 2.2.

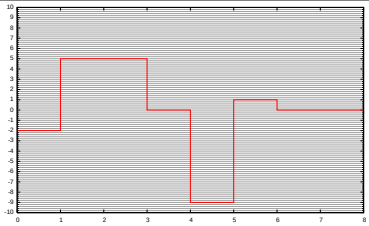
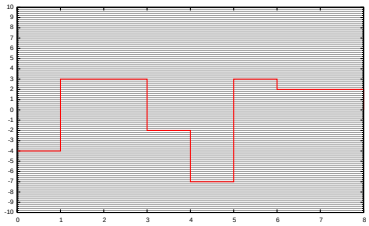
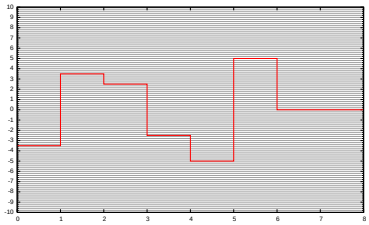
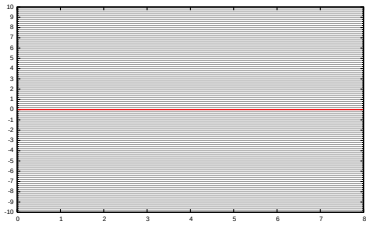
Function	Signal
x	
$x - \sum_{a \geq 4, b \in \mathcal{Z}} X[a,b]\varphi_{a,b}$	
$x - \sum_{a \geq 2, b \in \mathcal{Z}} X[a,b]\varphi_{a,b}$	
$x - \sum_{a \geq 1, b \in \mathcal{Z}} X[a,b]\varphi_{a,b}$	

Figure 2.2: Different iterations for Haar expansion of signal $x = (-2, 5, 5, 0, -9, 1)^\top$. It can be seen how the different frequency components of the signal x are separated by iteratively substracting them from the signal starting from the lowest frequencies. Finally, when all the frequency components are summed together, the original signal is perfectly reconstructed as shown in the last image.

In order to expand the Haar basis to cover any discrete signal $x[n] \in l_2(\mathcal{Z})$ for which $\sum_{n\in\mathcal{Z}} x[n]$ is not constrained to be 0, another definition of Haar is used:

$$\begin{aligned} \varphi_{2k}[n] &= \begin{cases} \frac{1}{\sqrt{2}} & \text{if } n \in \{2k, 2k+1\} \\ 0 & \text{otherwise} \end{cases} \\ \varphi_{2k+1}[n] &= \begin{cases} \frac{1}{\sqrt{2}} & \text{if } n = 2k \\ -\frac{1}{\sqrt{2}} & \text{if } n = 2k+1 \\ 0 & \text{otherwise.} \end{cases} \end{aligned} \tag{2.8}$$

For this definition of basis functions, one can define the transform as

$$\begin{aligned} X[2k] &= \langle \varphi_{2k}, x \rangle &= \frac{1}{\sqrt{2}}(x[2k] + x[2k+1]) \\ X[2k+1] &= \langle \varphi_{2k+1}, x \rangle &= \frac{1}{\sqrt{2}}(x[2k] - x[2k+1]) \end{aligned} \tag{2.9}$$

and the corresponding inverse transform as

$$x[n] = \sum_{k\in\mathcal{Z}} X[k]\varphi_k[n]. \tag{2.10}$$

The transform (2.9) using the basis (2.8) defined above analyzes the signal $x[n]$ in one scale, whereas the basis defined with equation (2.6) from the mother function (2.7) analyzes the signal in multiple scales. However, the transform (2.9) can be easily extended to multi-resolution analysis by recursively applying the same transform to the *low frequency component* $y_0[k] = X[2k]$ of the transform result. The frequency components are computed iteratively as follows:

$$\begin{aligned} y_1^{(i)}[k] &= \langle \varphi_{2k+1}, y_0^{(i-1)} \rangle \\ y_0^{(i)}[k] &= \langle \varphi_{2k}, y_0^{(i-1)} \rangle, \end{aligned} \tag{2.11}$$

where $y_0^{(0)}[k] = y_0[k] = X[2k]$ and $y_1^{(0)}[k] = y_1[k] = X[2k+1]$.

The iteration results $y_1^{(0)}, y_1^{(1)}, \ldots, y_1^{(M)}$ and $y_0^{(M)}$ are called the *bands* or the *components* of the signal decomposition analyzing the signal in $M+1$ scales. Furthermore $y_0^{(M)}$ is the *low-frequency* or *scaling component* and $y_1^{(0)}, y_1^{(1)}, \ldots, y_1^{(M)}$ are the *high-frequency components* of the signal listed from the highest to lower frequencies. It should be noted that the components $y_0^{(0)}, y_0^{(1)}, \ldots, y_0^{(M-1)}$ need not be stored in signal compression applications, because they can be reconstructed iteratively from the other frequency components. This is done iteratively by reconstructing each of the low frequency components $y_0^{(i)}$from the already known components $y_0^{(i+1)}$ and $y_1^{(i+1)}$ by reversing the iteration step defined by the equation (2.11).

An implementation of the transform can be created with *filter banks*. The idea is to use a *low pass filter* $h_0[n]$ and a *high pass filter* $h_1[n]$ for computing the

transform result $X[k]$ as a convolution of the *analysis filters* with the original signal $x[n]$:

$$\begin{array}{rclcrcl} y_0[k] & = & X[2k] & = & h_0[2k] * x[2k] & = & \sum_{l\in\mathcal{Z}} h_0[2k-l]x[l] \\ y_1[k] & = & X[2k+1] & = & h_1[2k] * x[2k] & = & \sum_{l\in\mathcal{Z}} h_1[2k-l]x[l]. \end{array} \tag{2.12}$$

The analysis filters can be formed from the basis functions by inverting the time:

$$\begin{array}{c} h_0[n] = \varphi_0[-n] \\ h_1[n] = \varphi_1[-n]. \end{array} \tag{2.13}$$

For the transformation defined by the equation (2.9), the analysis filters are

$$\begin{array}{rcl} h_0[n] & = & \begin{cases} \frac{1}{\sqrt{2}} & \text{if } n \in \{-1, 0\} \\ 0 & \text{otherwise} \end{cases} \\ h_1[n] & = & \begin{cases} \frac{1}{\sqrt{2}} & \text{if } n = 0 \\ -\frac{1}{\sqrt{2}} & \text{if } n = -1 \\ 0 & \text{otherwise.} \end{cases} \end{array} \tag{2.14}$$

Applying these filters to equation (2.12) leads back to the earlier definition (2.9) of the transform.

One of the most frequently used orthonormal bases for multi-resolution analysis is defined by *Daubechie's family of orthonormal wavelet mother functions* [62]. The filters used to implement discrete wavelet transform for this family are constructed by iteratively solving roots for polynomial function that defines the filters [16, 18]. The length of the filtering functions depend on the number of iterations used. For an example, the coefficients for Daubechies low-pass filters $h_0^{(D_4)}$ and $h_0^{(D_6)}$ of lengths 4 and 6 respectively are

$$h_0^{(D_4)} = \begin{pmatrix} 0.48296291 \\ 0.8365163 \\ 0.22414386 \\ -0.129409522 \end{pmatrix} \tag{2.15}$$

and

$$h_0^{(D_6)} = \begin{pmatrix} 0.33267 \\ 0.806891 \\ 0.459877 \\ -0.135011 \\ -0.08544 \\ 0.03522 \end{pmatrix}. \tag{2.16}$$

The corresponding high-pass filters for an orthonormal Daubechies filter bank can be computed as

$$h_1[n] = (-1)^n h_0[-n + L - 1], \tag{2.17}$$

where L is the length of the low-pass filter. The wavelet transform can be then computed by recursive convolutions as shown in the equations (2.11) and (2.12).

A popular basis for wavelet transforms used in lossy compression applications is often shortly called as B9/7. The name refers to a *biorthogonal Daubechies wavelet basis* where the lengths of the low- and high-pass analysis filters $h_0^{(B_{97})}$ and $h_1^{(B_{97})}$ are 9 and 7 respectively. These filters provide a very efficient representation for natural images while being relatively short. Both filters are symmetric, so that these filters are defined by the means of a filters:

$$h_0'^{(B_{97})} = \begin{pmatrix} 0.852699 \\ 0.377403 \\ -0.110624 \\ -0.023849 \\ 0.037829 \end{pmatrix} \tag{2.18}$$

and

$$h_1'^{(B_{97})} = \begin{pmatrix} 0.788485 \\ -0.418092 \\ -0.040690 \\ 0.064539 \\ 0 \end{pmatrix}, \tag{2.19}$$

where $\forall n \in \{-4, -3, \ldots, 3, 4\} : h_i^{(B_{97})}[n] = h_i'^{(B_{97})}[|n|]$.

Because of the biorthogonality, the *synthesis filters* $g_0^{(B_{97})}$ and $g_1^{(B_{97})}$, that are used to compute the inverse transform, can be calculated from the analysis filters as

$$\begin{array}{rcl} g_0^{(B_{97})}[n] & = & (-1)^n h_1^{(B_{97})}[n] \\ g_1^{(B_{97})}[n] & = & (-1)^n h_0^{(B_{97})}[n]. \end{array} \tag{2.20}$$

2.3 Boundary handling for finite signals

The transforms above discuss only the case where the discrete signal x belongs to an infinite-dimensional space $l_2(\mathcal{Z})$. If the impulse response is short for the filters used to implement the recursive transform, the finite-dimensional signal $x' \in \mathcal{R}^N$ can be transformed by constructing an infinite-dimensional signal x from x' with *zero padding*:

$$x[n] = \begin{cases} x'[n] & \text{if } 0 \leq n < N \\ 0 & \text{otherwise.} \end{cases} \tag{2.21}$$

The problem with zero-border handling is that when the filter length is longer than 2, the transformed signal X might contain non-zero elements outside the original range $(0, N-1)$ of x'. This signal growth happens on each iteration

in recursive transforms. As the purpose of signal compression is to represent the signal with as few bits as possible, it is not feasible to store the additional elements of the transformed signal X, that are required for reconstruction of the original signal.

A strategy for constructing a transform where the length of the signal does not grow is called *circular convolution*. There the transform is done with an infinite-dimensional signal x formed from the finite-dimensional signal x' as

$$x[n] = x'[n \bmod N]. \tag{2.22}$$

The transform results in a periodic transformed signal X with the period N. A weakness of the circular signal border handling is that it might introduce a discontinuation point on the signal border. This should be avoided in signal compression applications, because the signal discontinuities appear as large coefficients on high-frequency bands of the transformed signal.

If the filter is symmetric, it is possible to extend the original signal x' with *symmetric extension* [6]. Symmetric extension of x' is calculated by mirroring the signal on the borders to construct the corresponding infinite signal x. The details of the mirroring depend on the length of the filters used, the signal length of x' and whether we are considering the analysis or synthesis filter banks. Mirroring does not introduce harmful discontinuities to the signal and is thus a preferable extension used for signal compression applications.

Mirroring of the signal borders can be done either as *whole point* extension or as *half point* extension [10]. The whole point extension of a signal $x' \in \mathcal{R}^N$ is defined as

$$x[n] = \begin{cases} x[-n] & \text{if } n < 0 \\ x'[n] & \text{if } 0 \leq n < N \\ x[2N - n - 2] & \text{if } n \geq N \end{cases} \tag{2.23}$$

and the half point extension is defined as

$$x[n] = \begin{cases} x[-n - 1] & \text{if } n < 0 \\ x'[n] & \text{if } 0 \leq n < N \\ x[2N - n - 1] & \text{if } n \geq N. \end{cases} \tag{2.24}$$

The difference between whole and half point extensions is whether the coefficient on the border is duplicated when creating the signal extension by mirroring or not.

In the common case of the B9/7 wavelet transform with an even length signal [3], the whole point extension is used for the analysis as well as for the synthesis of the left border in low pass filtering and for synthesis of the right border in high pass filtering. The half point extension is used for the synthesis of the right border in low pass filtering and for synthesis of the left border in high pass filtering.

2.4 Two-dimensional signal decomposition

One-dimensional discrete wavelet transformations are directly extensible to two-dimensional discrete signals by using matrices for signal representation instead of vectors [39]. For implementation efficiency and complexity reasons the most popular approach for constructing a multi-resolution representation of an image is to apply one-dimensional wavelet filtering for the columns and the rows of the image.

One of the most frequently used two-dimensional signal decompositions is the *dyadic split decomposition* [62] that is also called the *Mallat decomposition* or the *octave-band decomposition*. The dyadic split decomposition is very effective for natural images and is selected to be used in the coding algorithms presented in this work. Another popular two-dimensional signal decomposition used with natural images is wavelet packet decomposition that can be seen as a generalization of the dyadic split decomposition [14, 57].

The dyadic split decomposition with L levels can be computed for an image $z \in \mathcal{R}^{W \times H}$ with the following algorithm:

- For all $l \in \{0, \ldots, L-1\}$
  - For all $r \in \{0, \ldots, H/2^l - 1\}$
    - Transform $x = [z_{0,r}, \ldots, z_{W/2^l-1,r}]^\top \rightarrow \{y_0, y_1\}$ as defined in equation (2.12).
    - For all $i \in \{0, \ldots, W/2^{l+1} - 1\}$:
      - $z_{i,r} \leftarrow y_0[i]$
      - $z_{i+W/2^{l+1}-1,r} \leftarrow y_1[i]$
  - For all $c \in \{0, \ldots, W/2^l - 1\}$
    - Transform $x = [z_{c,0}, \ldots, z_{c,H/2^l-1}]^\top \rightarrow \{y_0, y_1\}$ as defined in equation (2.12).
    - For all $i \in \{0, \ldots, H/2^{l+1} - 1\}$:
      - $z_{c,i} \leftarrow y_0[i]$
      - $z_{c,i+H/2^{l+1}-1} \leftarrow y_1[i]$,

where L is the number of iteration levels used.

The goal of the dyadic split decomposition is to separate the image bands that analyze the image in different resolutions and have minimal correlation. The organization of the bands is illustrated in Figure 2.3. The bands can be divided into three orientation pyramids A, B and C storing the vertical, horizontal and diagonal high frequency details of the image correspondingly. The bottom levels of the pyramids store the highest frequencies and each of the upper levels store the details with half the frequency compared to the level immediately below it. This way, the wavelet coefficients stored in the same spatial location of

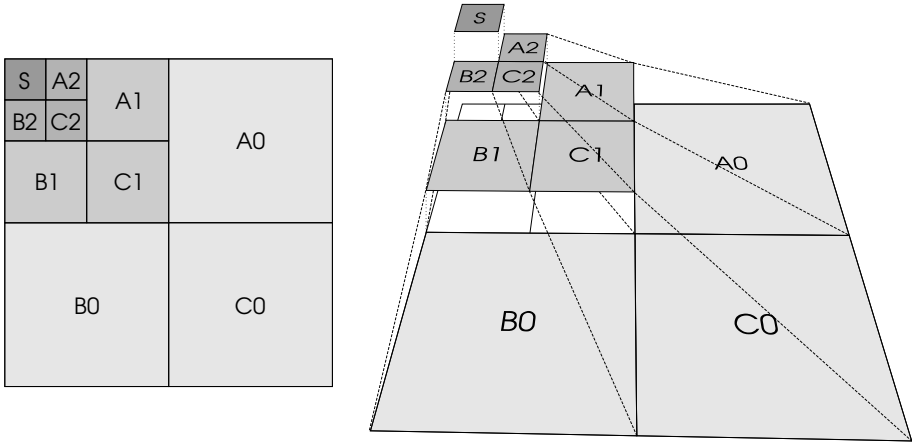


Figure 2.3: Illustration of the dyadic split decomposition. The different orientation components A, B and C are enumerated from the highest to the lowest frequency components. The low frequency component is denoted with S.

different orientation pyramids describe the same spatial location in the original image. In natural images, the discrete signal represents samples from a continuous and fairly smooth light intensity function and thus the energy of high frequency details is generally much lower than that of the low frequency details. Because of this and the recursive nature of the decomposition, the magnitude of the wavelet coefficients on the lower pyramid levels is generally considerably smaller than on the higher levels.

Figure 2.4: Example of dyadic split decomposition. The image on the left is transformed with 4-level dyadic split decomposition using an orthonormal Daubechies wavelet filter [62]. For illustrative purposes, the transformation result on the right is presented in logarithmic scale.

Chapter 3

Transform coding

This chapter introduces the transform coding concepts on which our algorithms presented in this thesis are built on. The concepts are summarized on a level of details that only gives a sufficient background knowledge for reading the thesis; for more details and results of the algorithms the reader is referred to the original publications.

First, the principles of the embedded zerotree wavelet (EZW) coding introduced by J. M. Shapiro are discussed. Then, the SPIHT sorting algorithm by A. Said and W. Pearlman, which builds on the EZW, is introduced. These two algorithms are the basis for the first two of our coding algorithms introduced later in this work. The latter two of our coding algorithms introduced in this work utilize coefficient coding contexts and prediction. A related context based coding algorithm by C. Chrysafis is described in order to help understanding these techniques. Furthermore, the basics of scalar- and vector quantization used for the prediction of the probability distribution are introduced.

Finally, optimization techniques necessary for generating code books used in vector quantization are briefly discussed. These techniques are used by our fourth coding algorithm. They also lay the basis for our clustering algorithm described in the final part of this work.

3.1 Embedded Zerotree Wavelet coding

The *embedded zerotree wavelet* (EZW) algorithm is an efficient image coder introduced by J. M. Shapiro [56]. It exploits the self-similarity of the bands in the octave-band decomposition on different scales and codes the wavelet coefficients as series of successive approximations using arithmetic coding.

EZW produces an *embedded bit stream*, which is a bit stream where the bits are ordered by their impact to the MSE of the decoded image. This guarantees that any prefix of the resulting bit stream can be decoded into the best possible

approximation of the original image with a given number of bits. This property makes it possible to meet the bit-rate or quality constraints exactly and to progressively view the image with increasing quality when receiving the coded image over a slow connection.

3.1.1 Significance maps

When coding the scalar quantized wavelet coefficients, estimating whether each coefficient is zero or non-zero is one of the problems with large significance to the resulting bit-rate. This is emphasized when the target coding rate is less than 1 BPP, because the value of the most coefficients must be 0. EZW focuses on this problem: the wavelet coefficient matrix is coded as a sequence of significance maps for different scalar quantizations. In other words, the non-quantized absolute values of the coefficients c_j in octave-band decomposition are compared against a set of thresholds $\{T_i | i \in \mathcal{N}\}$, where

$$T_i = \frac{T_0}{2^i} \tag{3.1}$$

and T_0 is chosen so that for all $c_j : |c_j| < 2T_0$.

When progressively sending better approximations of the coefficients in the octave-band decomposition, each coefficient can be in two different states: *insignificant state* and *significant state*. All coefficients c_j with $|c_j| < T_i$ are defined to be insignificant for the current threshold T_i and all the others significant. Because $T_i > T_{i+1}$ the state of the coefficients can change only from insignificant to significant state. When sending significance maps for $T_0, T_1, \ldots$ sequentially, only the state changes for coefficients which have been insignificant in previously sent map must be coded.

If the coefficient values are evenly distributed, the best approximation $a(c_j)$ of the coefficient c_j is

$$a(c_j) = s(c_j) \sum_{i=\mathcal{N}} b_i(|c_j|) T_i \tag{3.2}$$

where the sign $s(c_j)$ of c_j is defined as

$$s(c_j) = \begin{cases} 1 & \text{if } c_j > 0 \\ -1 & \text{if } c_j < 0 \\ 0 & \text{if the sign is unknown or } c_j = 0 \end{cases} \tag{3.3}$$

and $b_i(c_j)$ is the significance for the unknown part of c_j after i comparisons:

$$b_i(c_j) = \begin{cases} 1 & \text{if } T_i \leq |c_j| - \sum_{k=0}^{i-1} b_k(|c_j|) T_k \\ 0 & \text{if } T_i > |c_j| - \sum_{k=0}^{i-1} b_k(|c_j|) T_k \\ 0.5 & \text{if the threshold comparison result is unknown.} \end{cases} \tag{3.4}$$

By combining the above definitions of $b_i(c_j)$ and $s(c_j)$ an embedded bit stream can be produced by sending $b_i(c_j)$ for each i for each j and also sending $s(c_j)$ after the first 1 bit sent for the corresponding coefficient c_j. The decoder can calculate an approximation of the octave-band decomposition coefficient for any prefix of such a bit stream by using the definition of $a(c_j)$ above.

3.1.2 Coding with zerotrees

In natural images, the absolute values of the coefficients on octave-band decomposition at high frequency bands tend to be smaller than the coefficients on the lower frequency bands. More specifically, a coefficient on a band in octave-band decomposition is likely to have a larger absolute value than the four coefficients directly "below" it on the same orientation, see the pyramid decomposition in Figure 2.3.

EZW exploits this phenomenon by defining that for a given threshold T_i, a coefficient c_j is a root of *zerotree* if and only if

$$\forall c_l \in Z_j : |c_l| < T_i, \tag{3.5}$$

where Z_j is the set of all coefficients below c_j in the pyramid decomposition, including the c_j itself. All coefficients in the Z_j thus have the same orientation and are in the same spatial location as c_j on the bands corresponding to the same or higher frequencies than the band where c_j resides on. With this definition it is possible to code the insignificance state of all the coefficients in the set Z_j with a single bit by marking c_j to be a root of the zerotree.

The EZW coding algorithm combines the idea of marking the zerotrees with the embedded bit stream defined in the section 3.1.1. The algorithm can be summarized as follows:

- Set the initial threshold T_0 by finding the smallest k for which all c_j : $|c_j| < 2^{k+1}$ and set $T_0 = 2^k$
- For all $i \in \{0, \ldots, k\}$
  - For each coefficient c_j for which the significance status is not known, encode the status with arithmetic coding as one of the following:
    - Top of the zerotree (coding for all the other coefficients in Z_j can be omitted for this i);
    - Insignificant, but not the top of the zerotree;
    - Significant with a positive sign;
    - Significant with a negative sign.
  - For each significant coefficient c_j with $|c_j| \geq T_{i-1}$, code $b_i(c_j)$ with arithmetic coding. For coefficients c_j with $|c_j| < T_{i-1}$ the $b_i(c_j) = 1$ and thus coding of $b_i(c_j)$ can be omitted.

3.2 Set partitioning in hierarchical trees

The *set partitioning in hierarchical trees* (SPIHT) algorithm presented by A. Said and W. Pearlman [54] is an extension and different implementation of the ideas introduced in EZW. The SPIHT algorithm produces an embedded bit stream by storing significance maps and successive approximation information so compactly that it achieves without entropy coding the same MSE as EZW with entropy coding for the same BPP. By entropy coding the resulting bit stream, the compression efficiency is even better.

3.2.1 Coding bitplanes by sorting

If the thresholds T_i are selected to be powers of two in the method of sequentially submitting better approximations of all the coefficients described in the chapter 3.1.1, the method equals to sending the bit-planes for the matrix of quantized absolute coefficient values. Because the thresholds are computed from the initial threshold, this is achieved by selecting T_0 so that there exists $m \in \mathcal{N} : T_0 = 2^m$.

When coding natural images, most bits sent by a bit-plane coding algorithm are insignificant zero-bits in the front of the first one-bit that occurs in the coefficients. If the location and the number of significant bits for each significant coefficient are known, only a small number of significant bits $(s_2, s_3, \ldots)$, after the first one-bit $(s_1 = 1)$ in significant coefficients along with their signs must be coded. A good approximation of the coefficients would then be zero for the insignificant coefficients and the other coefficients could be calculated as

$$s(2^{M-1} + s_2 * 2^{M-2} + \ldots + s_{M-m}2^m + 0 + 2^{m-2} + 2^{m-3} + \ldots + 2^0), \quad (3.6)$$

where $s \in \{-1, 1\}$ is the sign, M is the number of significant bits, and $T = 2^m$ is the threshold. The sub-threshold part $T(|c|/T - \lfloor |c|/T \rfloor)$ of the coefficient absolute value $|c|$ is approximated to be $\lfloor T * 0.01111\ldots_2 \rfloor$ instead of the more intuitive alternative $\lfloor T/2 \rfloor$, because the distribution of the coefficient values is centered around zero.

Both the EZW and the SPIHT work basically in the same manner. They send the coefficient magnitudes, as measured by the number of their significant bits, by sorting the coefficients to the order of their magnitudes and sending the results of the comparisons done in sorting. If the decoder knows the number of coefficients with each magnitude, it can reconstruct the coefficient magnitudes by reversing the sorting process by using the received comparison results. An example of the *coding of bit-planes by sorting* is illustrated in the Figure 3.1.

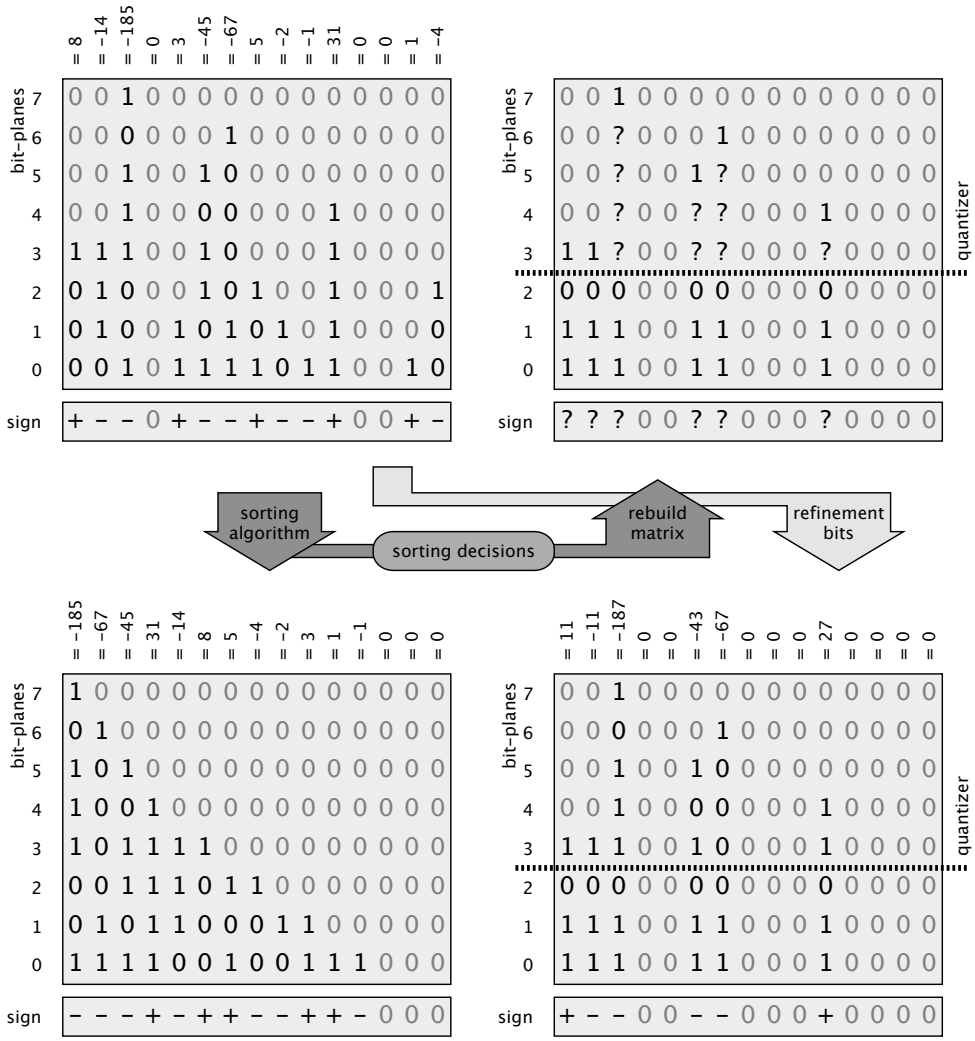


Figure 3.1: An example of coding bit-planes by sending sorting decisions. The top-left matrix represents the bits of the original wavelet coefficients stored as absolute values with separate signs. Insignificant digits are grayed. The sorting information is created by sorting the coefficients by their magnitudes. One possible sorting result is illustrated on the bottom-left matrix. The magnitude comparisons made by the sorting algorithm together with the number of coefficients of each magnitude is used to reconstruct an approximation of the original coefficients as illustrated on top-right matrix. The sorting information for a given quantization level contains the information on where the coefficients significant on that level are located. Finally, the significant signs and unknown significant bits marked with question marks are added to the approximation illustrated by the bottom-right matrix.

3.2.2 List based sorting algorithm

The SPIHT sorting algorithm defines four sets of coefficients:

- $\mathcal{H}$ contains all the coefficients on the low frequency band of the octave-band decomposition: this is the same as all the coefficients of the band S in the Figure 2.3.
- $\mathcal{O}_{i,j}$ is the set of four coefficients below the coefficient in the coordinates (i, j) on the coefficient matrix:

$$\mathcal{O}_{i,j} = \{c_{2i,2j}, c_{2i,2j+1}, c_{2i+1,2j}, c_{2i+1,2j+1}\}.$$

- $\mathcal{D}_{i,j}$ is the set of all coefficients below the coefficient in the coordinates (i, j) on the coefficient matrix:

$$\mathcal{D}_{i,j} = \mathcal{O}_{i,j} \bigcup \{c_{i',j'} \mid \exists i'', j'' : c_{i'',j''} \in \mathcal{D}_{i,j} \wedge c_{i',j'} \in \mathcal{O}_{i'',j''}\}.$$

- $\mathcal{L}_{i,j} = \mathcal{D}_{i,j} \setminus \mathcal{O}_{i,j}$.

The magnitude sorting algorithm in the SPIHT is based on maintaining three lists of coefficients: list of insignificant sets (LIS), list of insignificant points (LIP) and list of significant points (LSP). The algorithm then traverses through the coefficient matrix comparing significance of the coefficients to the thresholds $\{T_0, T_1, ...\}$. After going through the matrix for each threshold, refinement details (signs and successive approximation bits) are sent for each coefficient in the LSP list.

The SPIHT algorithm can be summarized as follows:

- Select m for which all $c_j : |c_j| < T_0 = 2^m$.
- Initialize lists: LSP $\leftarrow \emptyset$, LIP $\leftarrow \mathcal{H}$, LIS $\leftarrow \{\mathcal{D}_{i,j} : \forall c_{i,j} \in \mathcal{H}\}$.
- For all $i \in \{0, \ldots, k-1\}$
  - For all $c_{i,j} \in$ LIP : Send the result of $|c_{i,j}| \geq T_i$. If $|c_{i,j}| \geq T_i$ move $c_{i,j}$ from LIP to LSP and send the sign of $c_{i,j}$.
  - For all $\mathcal{D}_{i,j} \in$ LIS:
    - Send the result of test $(\forall c_{i',j'} \in D_{i,j} : c_{i',j'} < T_i)$. If the result is false:
      - For all $c_{i',j'} \in O_{i,j}$: Send the result of $|c_{i',j'}| \geq T_i$. If the result is true, send the sign of $c_{i',j'}$ and add the coefficient to LSP, otherwise, add it to LIP.
      - Remove $\mathcal{D}_{i,j}$ from LIS.

· If $\mathcal{L}_{i,j} \neq \emptyset$ add $\mathcal{L}_{i,j}$ to LIS.

- For all $\mathcal{L}_{i,j} \in$ LIS:
  - Send the result of the test $(\forall c_{i',j'} \in L_{i,j} : c_{i',j'} < T_i)$. If the result is false:
    - $\forall c_{i',j'} \in O_{i,j}$: Add $\mathcal{D}_{i',j'}$ to LIS.
    - Remove $\mathcal{L}_{i,j}$ from LIS.
- For all $c_{i,j} \in$ LSP : if $|c_{i,j}| \geq T_{i-1}$ send the result of the comparison:

$$|c_{i,j}| - T_{i-1}\lfloor |c_{i,j}|/T_{i-1} \rfloor \geq T_i.$$

3.3 Context based coding

The idea of the context based coding is to use already coded information available both in the decoder and encoder for predicting symbols that must be coded next. The use of context information can be applied to a wide range of coding applications and it has been used both in lossless image coding as well as in lossy coding algorithms.

Context based prediction can be used for predicting the values to be coded, the distribution of values or both of these. If the values are predicted directly, the prediction results can be subtracted from the originals and the difference can then be coded. This gives a coding scheme which is commonly referred as *prediction coding*. The context can also be used for classifying the values to different classes with similar properties. If the values in a class are statistically similar, the classes can be coded separately with an arithmetic coder using one statistical model for each class.

An efficient implementation of *context based wavelet coding* was introduced by C. Chrysafis and A. Ortega[11, 10] referred here as C/B. The basic idea of C/B is to quantize the octave-band decomposition using a scalar quantizer and then classify each coefficient to be coded using information from the surrounding already coded coefficients. The coefficients are coded with an arithmetic coder using the adaptive probability model selected by the classification for each coefficient. The compression performance of C/B is very good and in many cases better than SPIHT and EZW.

3.3.1 Context classification

Context based wavelet coefficient classification is a function $C_i \rightarrow k$, where k denotes the index of the coefficient class as defined by the context C_i. The context C_i for coding the i:th coefficient c_i consists of all the previously coded coefficients $\{c_0, c_1, \ldots, c_{i-1}\}$ that are already known by the decoder. This classification $C_i \rightarrow k$ can be used for selecting the probability distribution that is used for entropy coding the coefficient c_i.

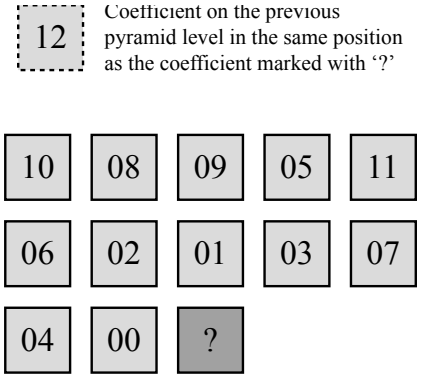


Figure 3.2: The context used by the C/B algorithm to classify the coefficient denoted with '?'. The neighboring coefficient positions that have been already coded and thus known by the decoder are enumerated from 0 to 12.

In practical implementations, C_i might be too large to be processed efficiently for each coefficient. This is why the most context based systems only consider a close neighborhood of the predicted coefficient. This smaller J-sized neighborhood N_i for coefficient c_i can be defined by mapping the index i of the predicted context to a set of indexes $\{n(i,j)\}$ of the coefficients belonging to the neighborhood as

$$N_i = \{c_{n(i,j)} | 0 \leq j < J\} \subseteq C_i. \tag{3.7}$$

For example, C/B defines the context used for the classification as illustrated in the Figure 3.2.

C/B uses scalar quantization when selecting the class for the context. First, an estimate $\hat{c}_i$ of the absolute value for coefficient c_i is calculated as the weighted average of the context coefficients:

$$\hat{c}_i = \sum_{j=0}^{J} w_j |c_{n(i,j)}|, \tag{3.8}$$

where the weights $\{w_j\}$ are specific to the different context positions. Then, the estimate $\hat{c}_i$ is quantized to calculate the class index k. Quantization is done by partitioning the estimate space to K ranges that each correspond to the class with index $k \in [0, K-1)$ in a such way that $q_k \leq \hat{c}_i < q_{k+1}$ when $\hat{c}_i$ belongs to class k. One class is assigned for each range in a such way that the ranges cover the estimate space fully:

$$\forall i : \hat{c}_i \in [0, \infty) = [q_0, q_1) \bigcup [q_1, q_2) \bigcup \cdots \bigcup [q_{K-1}, \infty), \tag{3.9}$$

where $\forall k : q_k < q_{k+1} \wedge q_k \in \mathcal{Z}$. In C/B, the quantization borders $\{q_k\}$ are selected to model $\hat{c}_i$ as an exponential random variable, but always allocating one class for zero predictions by selecting $q_0 = 0, q_1 = 1$.

A context classifier based on the scalar quantization ignores the signs for the coefficients because *sign prediction* of wavelet coefficients is hard to do reliably. It is possible to use context based methods also for predicting signs [41], but the results have not been very encouraging.

3.3.2 Vector quantization of the context space

A general way of classifying context information is to use *vector quantization* instead of scalar quantization. While in scalar quantization the one-dimensional estimate space is partitioned into ranges, in vector quantization the J-dimensional context space of all the contexts $\{N_i\}$ in the image is partitioned into K clusters $\{T_k | 0 \leq k < K\}$, where $N_i \in T_k$ if and only if c_i belongs to class k. Vector quantization can be used in many applications that require classification of complex data. In many cases the performance of vector quantization is very good when compared to application specific methods.

The most commonly used distance metric in vector quantization is the *Euclidian distance*:

$$d(x, y) = \sqrt{\sum_{j=0}^{J-1} (x[j] - y[j])^2}. \tag{3.10}$$

For each cluster T_k a *centroid* (mean vector) $m_k \in \mathcal{R}^J$ is defined as the cluster's representative item:

$$m_k = \frac{1}{|T_k|} \sum_{N_i \in T_k} N_i. \tag{3.11}$$

The partition is defined so that each vector N_i is assigned to the cluster with centroid m_k nearest to it:

$$N_i \in T_k \Rightarrow \forall l \neq k : d(m_l, N_i) \geq d(m_k, N_i). \tag{3.12}$$

3.4 Code book generation for vector quantization

The partition of the space together with the representative items for each cluster is called a *code book*. When a code book exists, mapping any vector to clusters is simple. If the vectors are coded by storing only the indexes of the clusters they belong to, the decoder can use the code book for approximating the coded vectors with the representative items of their clusters. Finding an optimal partition for the context space is a hard optimization problem. In this section we define the problem and discuss two optimization techniques for finding solutions.

We define indices of the cluster for each context as $P = (P_0, P_1, \ldots, P_{WH-1})$, where the number of contexts is determined by the dimensions W and H of the coefficient matrix. One possible way of defining the partition of the context space is by centroids $M = (m_0, m_1, \ldots, m_{K-1})$ of clusters. The problem of finding optimal partitioning can then be defined as finding such a solution $\omega^\star = (P, M)$ for the set of different contexts $\{N_i | i \in [0, WH)\}$ that minimizes the given *objective function* $e(\omega)$. The most commonly used objective function is *mean square error* (MSE) for the quantization:

$$e_{\mathrm{MSE}}(\omega) = \frac{1}{KWH} \sum_{i=0}^{WH-1} d(N_i, m_{p_i})^2 \tag{3.13}$$

The number of clusters K in the partition is selected so that the classification is as good as possible, but at the same time the number of clusters is as small as possible. Usually each cluster has its own dynamic probability distribution that is used for encoding the coefficients whose context belongs to that cluster. If K is too large, the number of coefficients per probability distribution is too small to adjust the probability distributions properly, which leads to inefficient entropy encoding. If the number of clusters is too small, there is not enough clusters available for the different types of contexts, which also leads to inefficient entropy encoding. The good selection for K depends on the application and the data and is often selected by hand.

3.4.1 Clustering by k-means

The well-known *k-means* algorithm is a simple and efficient technique for approximate optimizing the clustering solution. This algorithm is also often referred as *Generalized Lloyd algorithm* (GLA), *Linde-Buzo-Gray algorithm*(LBG) and *iterative self-organizing data analysis technique* (ISODATA) [47, 42]. The algorithm improves the existing solution iteratively and thus needs an initial solution to begin with. The quality of the result thus depends on the quality of the initial solution. The k-means algorithm is often used as the final or an intermediate step for other more complicated clustering algorithms to guarantee that the result is locally optimal.

The k-means algorithm consists of repeatedly recalculating cluster centroids, until a local optimum is found.

K-means algorithm:

- $\omega_0 \leftarrow$ given initial solution or a random solution of the clustering (P, M);
- $i \leftarrow 0$
- While $i \leq 1 \vee e_{\mathrm{MSE}}(\omega_i) < e_{\mathrm{MSE}}(\omega_{i-1})$

- Calculate the new centroids $M = (m_0, m_1, \ldots, m_{K-1})$ from mapping defined in ω_i as shown by equation 3.11.
- Calculate a new context to cluster mapping $P = (P_0, P_1, \ldots, P_{WH-1})$ by assigning contexts $\{N_j | j \in [0, WH)\}$ to new centroids M as shown by equation (3.12).
- $i \leftarrow i + 1$;
- $\omega_i \leftarrow (P, M)$.

3.4.2 Genetic algorithms

Genetic algorithms [28, 21, 49] are optimization techniques inspired by the evolutionary process in nature and they have been successfully applied to clustering [31]. In genetic algorithms solutions, often called the *individuals*, can be seen as *chromosomes* that consist of *genes* corresponding to the features of the individual solutions. The simplest coding for genes is to use one gene to represent each bit in the binary representation of the solution, but in many applications it is more efficient to select a higher-level application-specific gene assignment. Genetic algorithms iteratively produce *generations* of solutions, where the overall goal is to find new generations that have at least one solution that is better than the best solution of the previous generations.

New solutions are produced by *genetic operations*, which include *selection*, *crossover* and *mutation*. Selection is used to preserve selected individuals between generations. It can preserve the best solutions, try to preserve needed dissimilarity in the next generation or be completely random. Crossover is an operation where two chromosomes are sliced to gene sequences that are combined to compose one or more new individuals. Mutation simulates random changes to genes in the chromosome.

An example of a simple genetic algorithm [32] is defined as follows:

- $i \leftarrow 0$;
- Create the first generation G_0 of S individual solutions.
- Repeat until a stopping condition is met
  - Preserve S_B surviving individuals from the previous generation G_i to the new generation G_{i+1}.
  - Select $(S - S_B)/S_\gamma$ pairs of individuals from G_i to produce offspring, where S_γ is the number of offspring produced by a single crossover. Add the offspring to new generation G_{i+1};
  - Mutate some individuals in G_{i+1};
  - $i \leftarrow i + 1$;

- Output the best individual $\omega_{\text{best}} \in G_i$.

The parameters of the above algorithm include the number of individuals preserved between generations (S_B), the size of generations (S), the number of individuals created by crossover operation (S_γ) and the selection of the genetic operations.

Genetic algorithms can be applied to a clustering problem directly, but in many approaches they have been combined to a local optimization algorithm, such as k-means, for better performance [20]. Also the use of genetic algorithms is not limited to the optimization of the individual solutions: in algorithms with many parameters it is possible to use a genetic algorithm for the optimization of the parameter set to produce algorithms that are more efficient and easier to use. The *Self-adaptive genetic algorithm* (SAGA) [33] shows that an excellent performance can be achieved by including optimization strategy parameters, such as crossover method, mutation probability and mutation noise range.

Chapter 4

Summary of publications

The thesis constitutes of five publications: The first two publications introduce new techniques for *limiting distortion* in embedded lossy wavelet compression methods. The third publication introduces a simple new method for *coefficient prediction and coding using contexts*. The fourth publication introduces an advanced new method for *coding the wavelet coefficients by clustering of context space*. The final paper introduces a new *method for clustering using a distributed self-adaptive genetic algorithm*.

This chapter outlines the contents of the included articles. A brief performance comparison of the described algorithms with standard compression algorithms is given in the end of the chapter. The included publications are:

1. Antero Järvi, Joonas Lehtinen and Olli Nevalainen, **Variable quality image compression system based on SPIHT**, Signal Processing: Image Communications, vol. 14, pages 683–696, 1999.

2. Joonas Lehtinen, **Distortion limited wavelet image codec**, Acta Cybernetica, vol. 14, no. 2, pages 341–356, 1999.

3. Joonas Lehtinen, **Predictive depth coding of wavelet transformed images**, Proceedings of SPIE: Wavelet Applications in Signal and Image Processing, vol. 3813, no. 102, Denver, USA, 1999.

4. Joonas Lehtinen and Juha Kivijärvi, **Clustering context properties of wavelet coefficients in automatic modelling and image coding**, Proceedings of IEEE 11th International Conference on Image Analysis and Processing, pages 151–156, Palermo, Italy, 2001.

5. Juha Kivijärvi, Joonas Lehtinen and Olli Nevalainen, **A parallel genetic algorithm for clustering**, in Kay Chen Tan, Meng Hiot Lim, Xin Yao and Lipo Wang (editors), *Recent Advances in Simulated Evolution and Learning*, World Scientific, Singapore, 2004 (to appear).

4.1 Variable quality image compression system based on SPIHT

The SPIHT algorithm [54] outlined in chapter 3 provides an efficient compression system that produces an embedded bit stream. A new *variable quality compression system based on SPIHT* (vqSPIHT) is presented. The system supports marking a region of interest (ROI) in the image as a list of rectangles and to preserve more details on those regions. This is achieved by selecting a point in the embedded bit stream after which only bits contributing to the details on the ROI are included in the final image.

An application of the vqSPIHT compression system to *mammography image coding* is given. In most medical imaging applications, it is necessary to preserve the diagnostic quality of the compressed images, that is, to guarantee that no details contributing to the diagnosis are lost. When the mammogram images are coded with SPIHT, the most easily lost diagnostically important details are the microcalcifications. The microcalcifications often occur in groups and the size of each microcalcification can be only a few pixels. The compression system must preserve the exact shapes and locations of the microcalcifications, as they are important in the diagnosis of breast cancer. We integrate an *automatic microcalcification detection algorithm* [19] to vqSPIHT in order to construct the ROI automatically for compression of mammogram images. This compression system achieves considerably better compression efficiency than SPIHT, when both algorithms are constrained to be diagnostically lossless. After publishing this, other studies [50, 27] have agreed that variable quality compression systems can be more suitable to mammography compression than traditional lossy compression.

The original SPIHT algorithm uses three lists for maintaining sorting state information during the compression. When compressing high-resolution images, such as mammograms, the memory consumption of a straightforward implementation for SPIHT is very high. We present a new *memory efficient implementation* of the algorithm by modifying it to use matrices for state maintenance during sorting. The memory requirements of matrix based vqSPIHT are less than half of the memory requirements for original SPIHT. Later similar reduced memory implementation of SPIHT have been published [64].

4.2 Distortion limited wavelet image codec

In this paper a *new EZW based wavelet coding scheme is introduced.* We refer to the new coding scheme with name *Distortion Limited Wavelet Image Codec* (DLWIC). The codec is designed to be *simple to implement*, *fast* and have *modest memory requirements.* It is also shown, how the *distortion of the result can*

be calculated while progressively coding a transformed image, if the transform is unitary.

EZW exploits spatial inter-band correlations of the wavelet coefficient magnitudes by coding the bit-planes of the coefficient matrix in a hierarchical order. The order is defined by a quad-tree structure where the completely zero subtrees (zerotrees) can be represented by only one symbol. In DLWIC the correlations between different orientations are also taken into account by binding together the coefficients on three different orientation bands in the same spatial locations. The maximum absolute values of the coefficients in all subtrees are stored in a *two-dimensional heap structure*. This allows the coder to test the zerotree property of a subtree with only one comparison. A binary arithmetic coder with multiple separate probability distributions (states) is used to reach compression performance that is similar to the previously known EZW variants.

A wavelet transform is used to construct an octave band composition. The value of the square error (SE) is updated in the compression process. We start by calculating the initial SE as the total energy of the image and decrease it while bits are sent according to the information content of the bits. For every bit sent, the change in SE of the image is defined by the difference between the predictions for the coefficient before and after the bit is sent. Calculations can be implemented efficiently with table lookups. This has the advantage that we know the MSE of the final decompressed image already in the coding phase and *can stop the transmission of bits when an acceptable distortion level is reached*.

The efficiency of the DLWIC is compared to an advanced EZW variant and the industry standard JPEG using a set of test images. An estimation on speed and memory requirements of the DLWIC algorithm are made.

A simplified implementation of the DLWIC algorithm was published as *GNU Wavelet Image Codec* (GWIC) [40] that does not include distortion control features, but adds support for color images. Because of the simplicity of DLWIC and the availability of the implementation, it has been used as a basis for various projects [59, 12, 35], including a wavelet-based video codec [36], a mobile thin-client [2], a low-power embedded system [52].

4.3 Predictive depth coding of wavelet transformed images

In this paper, a new *prediction based method for lossy wavelet image compression* is presented. It uses dependencies between the subbands of different scales as do the more complex tree-based coding methods. Furthermore, the compression method uses dependencies between spatially neighbouring coefficients on a subband and between subbands representing different orientations on the same scale. Despite of its simpicity, the prediction based method achieves good

compression performance.

Coding of octave band composition created by biorthogonal 9-7 wavelet transform is straightforward. The coefficients of the transformed image are quantized with a simple uniform scalar quantizer. For each quantized coefficient that is not zero, the sign and the significant bits are coded. In addition to these, the decompressor must know the number of coded bits for each coefficient, which we call the depth. The *depth of each coefficient is predicted from nine related coefficient depths using a linear predictor* and the prediction error is then coded using arithmetic coding.

The linear predictor uses six known spatial neighbors of the coefficient, two other coefficients on the same level (scale) and spatial location, but on different subbands and the coefficient on the previous level in the same location. The weights for the linear predictor are approximated adaptively during the compression.

The compression method is tested with a standard set of images and the results are compared with SFQ, SPIHT, EZW and context based algorithms. The compression performance of PDC is found to be comparable with compared methods, which shows that simple linear predictors can be used for wavelet coefficient modeling.

4.4 Clustering context properties of wavelet coefficients in automatic modelling and image coding

A *new method for modeling and coding wavelet coefficients by clustering* is proposed. The method shows how any properties calculated from the coding context can be used for modeling the probability distribution of the coefficient being coded.

The coding system first transforms the image using a wavelet based transform. Then, *optimal coding parameters* (Δ, W, C) *meeting the given target bitrate are iteratively searched.* Symbol Δ stands for the scalar quantization step size, W is the set of weights for the context and C is the set of the cluster centroids that define the partitioning. The iteration process alternates between finding a suitable Δ for the clustering C found in the previous iteration and constructing new clustering for a given Δ. A formula for calculating the weights W is given. The coefficient is modeled by selecting the cluster centroid nearest to the weighted context parameters of the coefficient and using the probability model connected to that cluster for entropy coding. The coefficients are quantized using *rate distortion quantization* [10] and finally coded using an arithmetic coder.

The coding system is tested against the state of the art methods [54, 11, 66] with a simple set of context properties. The coding efficiency of the system

matches the state of the art methods, but the implementation of the proposed iterative optimizer is fairly complex and slow. Because the framework is fully self-adaptive, it can be used as a research tool for testing the efficiency of different selections for context properties.

4.5 Clustering by a parallel self-adaptive genetic algorithm

The quality of a clustering solution is critical for compression performance in context classification based wavelet image compression. We describe a new *algorithm for finding high-quality clustering solutions*. The algorithm utilizes *self-adaptive genetic algorithms for solution optimization* and a new *island model for parallelization*.

The *self-adaptive genetic algorithm for clustering* (SAGA) [33] is based on *individual level self-adaptation* [26, 43], where each individual consists of a solution candidate for the problem and a set of *strategy parameters*. The candidate solution includes both the cluster centroids and the individual to cluster mapping. The strategy parameters include the *cross-over method*, the *mutation probability* and the *noise range* used in the mutations. The actual genetic optimization algorithm is fairly similar to the algorithm described in Chapter 3.4.2.

The parallel SAGA (parSAGA) algorithm uses the *island parallelization model* [61], where a number of genetic algorithms run independently but communicate with each other. This simulates a biological model where separate populations live on geographically remotely located islands and occasionally send individuals to other islands. To easily control the process centrally, we implement the island model by using a common so-called *genebank*, where all the islands can send their individuals and receive new individuals. The probability of each individual for traveling from an island to another is controlled by a island topology model where the locations of the islands on a two-dimensional virtual sea are taken into account and the moving to one direction can be favored over another.

Two new *statistical measures are proposed* for the island model and the algorithm is tested with different parameters against other algorithms. The tests show that both SAGA and parSAGA algorithms outperform the compared algorithms when solving hard problems. While the quality of the solutions produced both by SAGA and parSAGA algorithms are equal, the distribution model of the parSAGA algorithm allows the solutions to be achieved faster.

4.6 Performance comparison

The goals of the compression algorithms summarized in the previous sections vary: The first two algorithms (vqSPIHT and DLWIC) demonstrate how to add quality control features into an embedded wavelet coder and the latter two (PDC and ACPC) show how the scalar and vector quantization can be applied to wavelet image coding. While the compression performance has not been a high priority goal, it is important to achieve an acceptable compression performance when adding new features.

The Figures 4.2, 4.3, 4.5 and 4.4 profile the performance of several compression algorithms for a set of test images shown in Figure 4.1. For each image, the quality measured as PSNR of the decompressed image is shown for all compared algorithms with bitrates between 0 and 0.5 BPP. The Figure 4.6 includes magnifications of the compression results for the test image Barbara compressed with different bit-rates and algorithms.

The following algorithms are included in the comparison: JPEG, JPEG 2000, SPIHT, vqSPIHT, ACPC, PDC and DLWIC. The JPEG implementation used is provided by Independent JPEG Group's libjpeg package version 6b using the default options for compression. The JPEG 2000 implementation used is the Java-based JPEG verification model version 4.1 provided by the JPEG Group with default options. The SPIHT algorithm implementation used is provided by A. Said and W. Pearlman. Note that this version of SPIHT gives slightly better results than the original algorithm [54] used for comparison in the original papers. Our variable quality SPIHT algorithm is used with empty ROI. Our clustering based context modeling algorithm here referred as ACPC is used for measuring compression performance. The compression performance calculations use a code book with 16 clusters and the context described in [11] with zeros for unknown coefficients. PDC and DLWIC algorithms of this work use parameters described in the included publications.

The compression performance of JPEG 2000, SPIHT, vqSPIHT, PDC and ACPC algorithms is fairly similar for all tested images. JPEG 2000 and ACPC consistently give the best results, but are not comparable because the ACPC implementation is an order of magnitude slower than the other algorithms tested. DLWIC gives good compression results for bit rates below 0.1 BPP, but is left behind the more complex wavelet compression algorithms with higher bit rates. The quality produced by JPEG is about 2-4dB lower PSNR than the wavelet based algorithms with the same bit rates. When the bit rate is lowered below 0.2 BPP, the quality of JPEG drops at a considerably faster pace than for wavelet-based algorithms.

Figure 4.1: Test images for compression performance comparison. From left to right the images are standard test images Barbara, Goldhill and Lenna of size 512×512. The landscape image below is of size 2048×1024. All images use 8 bits luminance resolution.

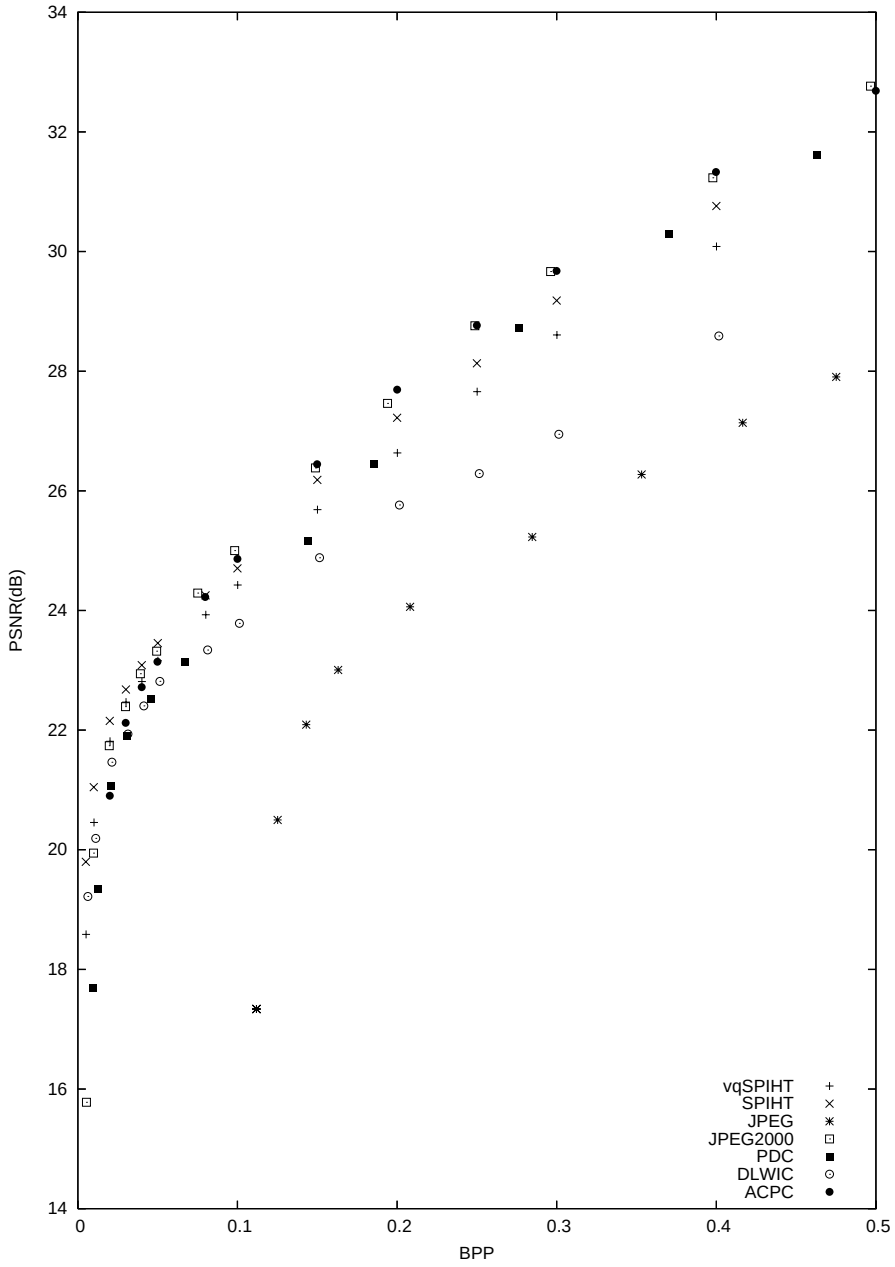


Figure 4.2: Compression performance comparison for the test image Barbara.

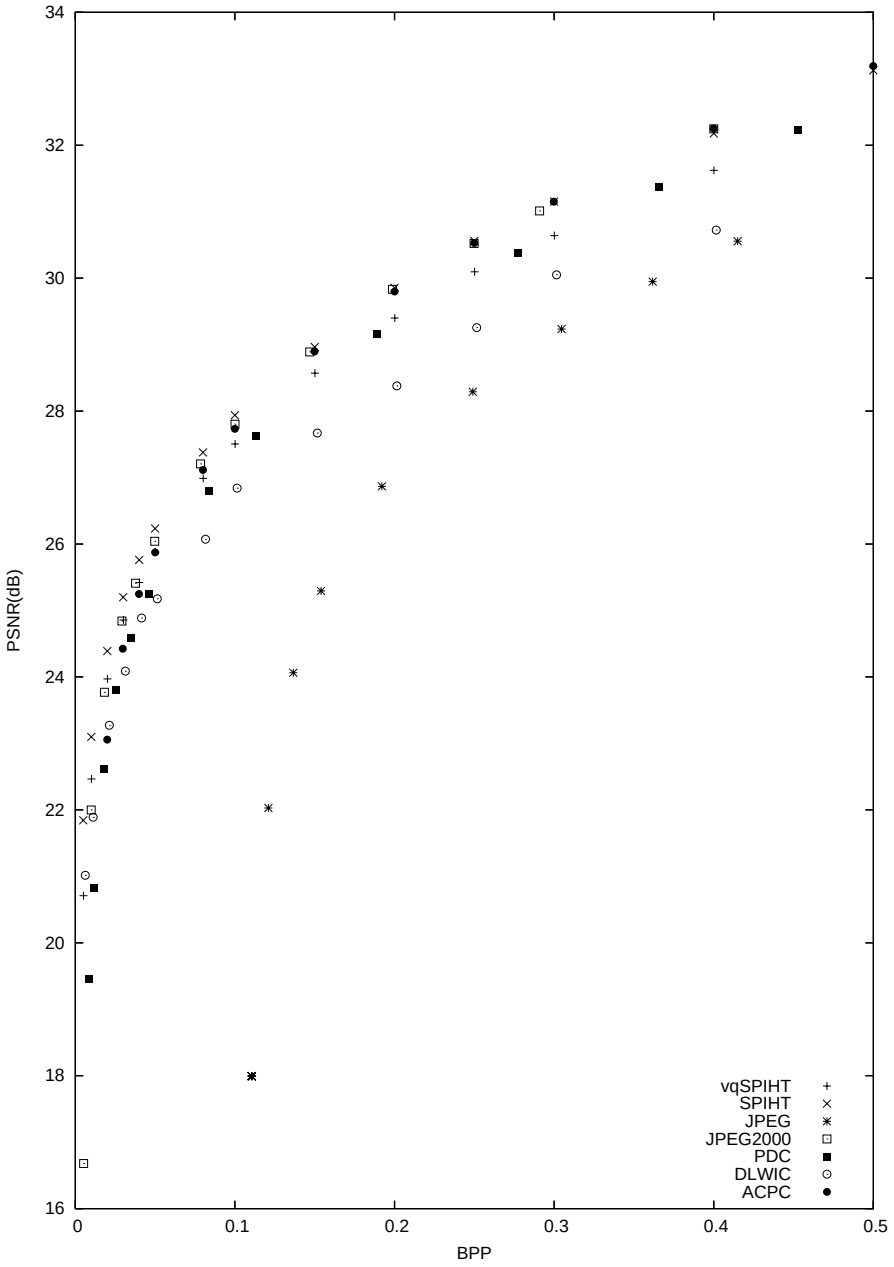


Figure 4.3: Compression performance comparison for the test image Goldhill.

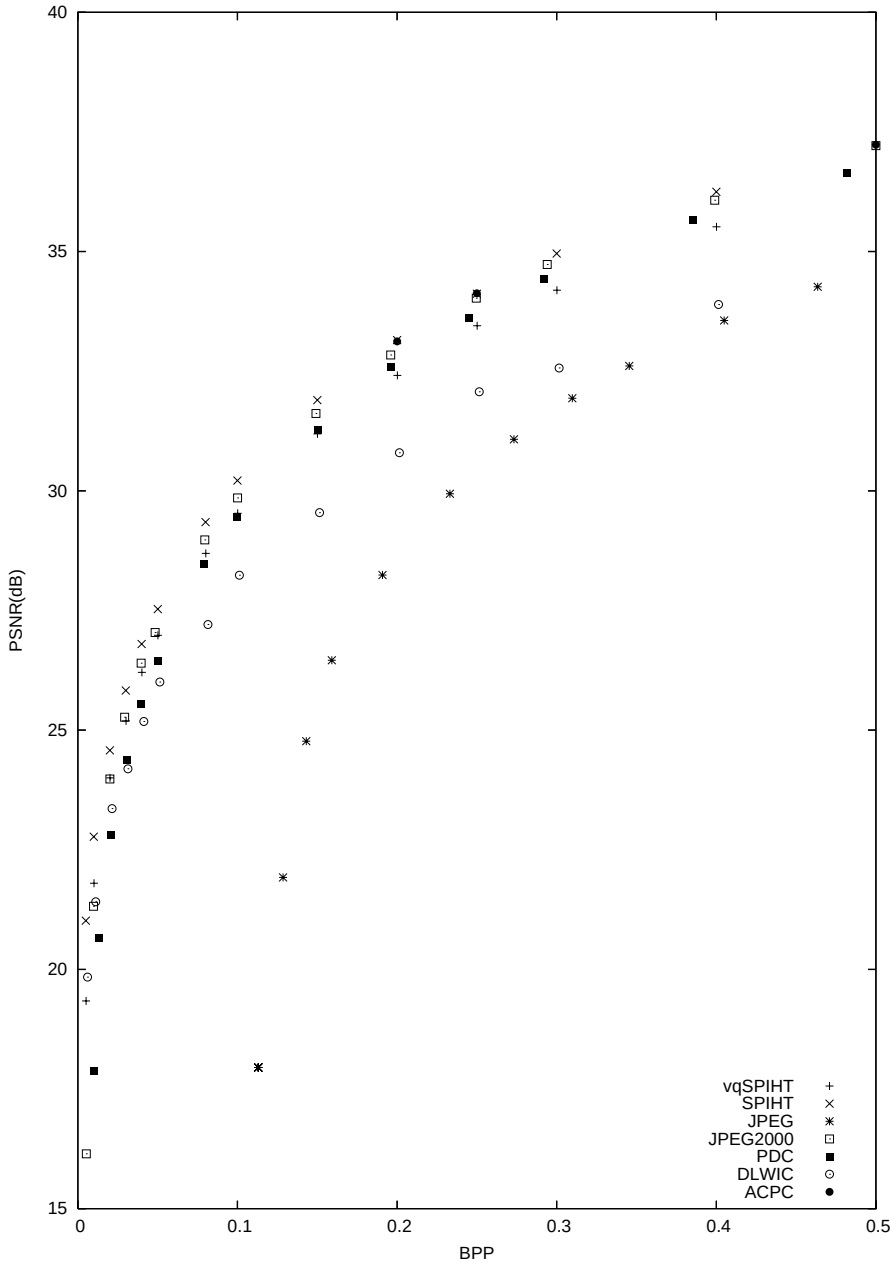


Figure 4.4: Compression performance comparison for the test image Lenna.

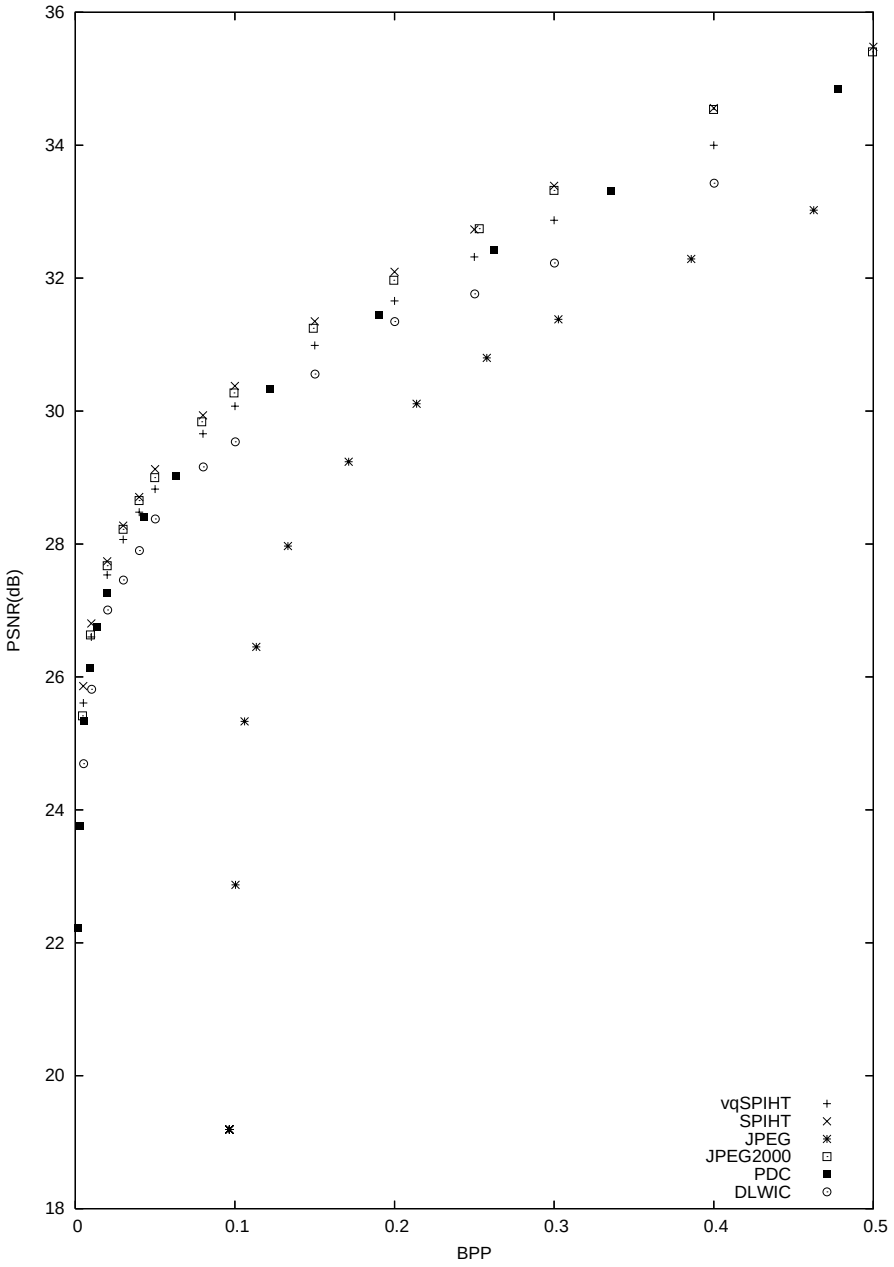


Figure 4.5: Compression performance comparison for the test image Landscape.

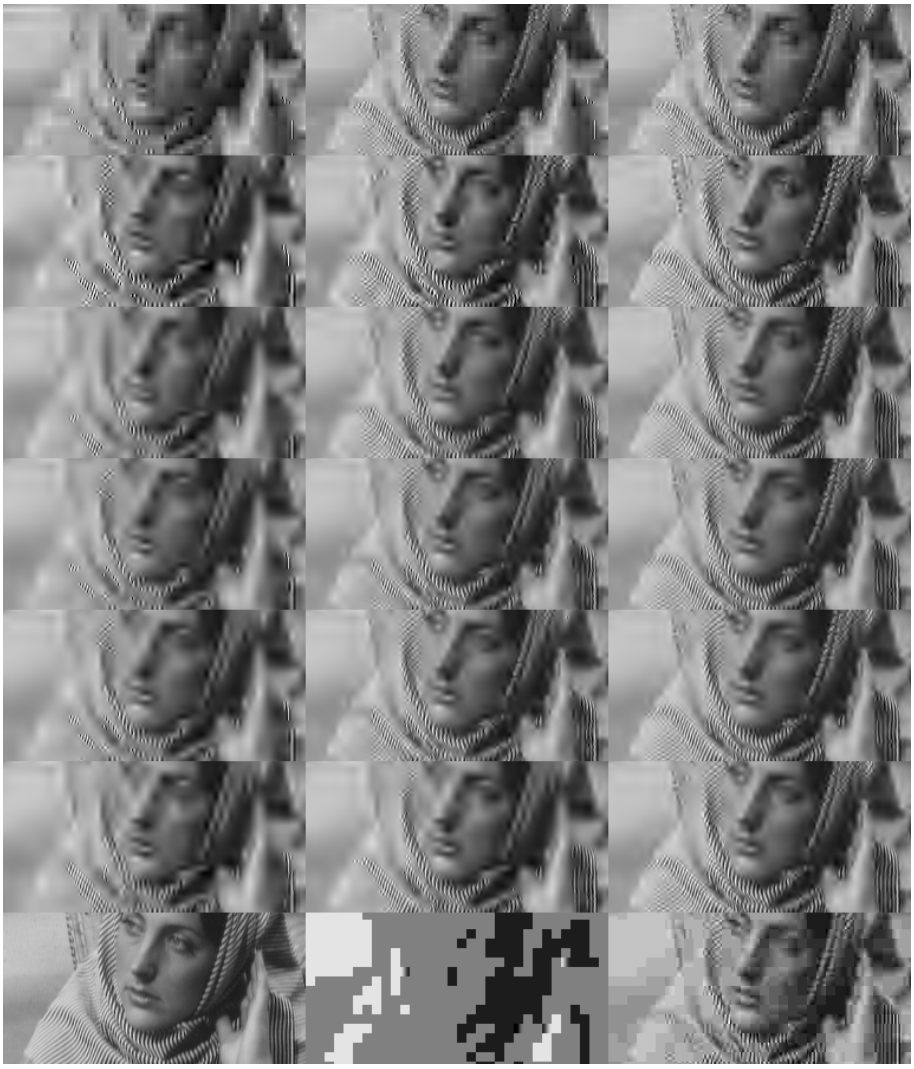


Figure 4.6: Magnifications of the compression results for the test image Barbara compressed with different bit-rates and algorithms. The images on left column are compressed with 0.05 BPP, middle with 0.1 BPP and 0.2 BPP on the right column. The algorithms tested are from top to bottom: DLWIC, PDC, ACPC, vqSPIHT, SPIHT, JPEG2000 and JPEG. Because the JPEG algorithm could not reach bit-rates under 0.1 BPP, the original image is presented instead of JPEG 0.05 BPP.

Chapter 5

Conclusions

In this work, four new methods for coding wavelet transformed images and one new method for clustering have been proposed. It was shown how the different regions of an image can be coded in variable quality in an embedded wavelet image coding method and how this can be applied to mammography image compression. A method was proposed for limiting distortion by maintaining an estimate of the MSE for the image throughout the coding process in the embedded method. It was shown how the context based prediction can be applied to wavelet image compression for constructing a simple but efficient coding system. A novel framework for wavelet image compression by clustering a given set of context properties was proposed and it was shown to provide excellent compression performance. Finally, a new clustering method based on self-adaptive distributed genetic optimization was proposed.

The proposed *variable quality image coding algorithm based on set partitioning in hierarchical trees* (vqSPIHT) extends the state of the art SPIHT algorithm both by adding a *novel method for enhancing image quality on selected regions* and by proposing a *memory efficient alternative implementation of the SPIHT* sorting algorithm. The efficiency and the benefits of variable quality image compression were demonstrated with an integrated system for mammography compression with automated ROI detection. The compression efficiency of vqSPIHT was shown to be superior to conventional methods in the mammography image compression and the memory efficiency of the implementation was observed to be significantly better than that of the original SPIHT algorithm.

The proposed *distortion limited wavelet image codec* (DLWIC) provides a simplified implementation of EZW and shows *how an embedded wavelet image coder can be controlled by target distortion instead of target bit-rate.* The implementation of the algorithm is based on an efficient *two-dimensional heap structure for zerotree property testing.* The compression efficiency of the DL-

WIC algorithm was shown to be comparable to SPIHT, while the compression algorithm is considerably simpler.

The proposed *predictive depth coding* (PDC) algorithm demonstrates *how the conventional prediction coding principles, originally used in lossless image-coding, can be applied to lossy wavelet transform coding.* It was shown how the number of significant bits in wavelet coefficients can be successfully predicted from the context by using a simple linear predictor. The compression efficiency of the proposed method is comparable to other wavelet based methods. It was also demonstrated that context based methods can be used for wavelet coefficient sign prediction, but the achieved coding efficiency benefits are questionable.

The proposed *automatic context property based coding* (ACPC) algorithm shows *how clustering methods can be used for the classification of wavelet coefficient context properties.* Our algorithm provides an *automated framework for using any context properties for modeling and coding the wavelet coefficients.* The framework makes it easy to test the suitability and efficiency of any context based statistical measure for compression systems. It was observed that selection of context properties can produce excellent compression results with the proposed ACPC coder. The speed and complexity of the optimization algorithm limits the use of the system in practical applications.

The parallelization of genetic algorithms for clustering was studied and a new *parallel self-adaptive genetic algorithm* (parSAGA) was proposed. A *general model that allows implementation of different island model topologies by parametrization* was given. The parSAGA was observed to achieve the same quality results as the sequential SAGA algorithm, but in considerably shorter time. Both algorithms were observed to outperform the other tested methods in large clustering problems. The speedup of the parSAGA over the sequential SAGA was in some cases observed to be superlinear because the proposed genebank model retains more diversity in populations and thus keeps finding better solutions more efficiently. Finally, *two new statistical diversity measures for parallel genetic algorithms* were proposed and used for studying the behavior of the distribution model.

References

[1] N. Ahmed, T. Natarajan, and K. R. Rao. Discrete Cosine Transform. *IEEE Transactions on Computers*, 23(1):90–93, January 1973.

[2] M. Al-Turkistany and A. Helal. Intelligent adaptation framwork for wireless thin-client environments. In *IEEE Symposium on Computers and Communications - ISCC'2003*, Kemer - Antalya, Turkey, June–July 2003.

[3] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies. Image coding using wavelet transform. *IEEE Trans. on Image Proc.*, 1(2):205–220, April 1992.

[4] R. N. Bracewell. *The Fourier Transform and Its Applications*. McGraw-Hill, 2000.

[5] E. O. Brigham. The fast fourier transform. In *Prentice Hall*, 1974.

[6] C. M. Brislawn. Preservation of subband symmetry in multirate signal coding. *IEEE Trans. Signal Processing*, 43(12):3046–3050, December 1995.

[7] C. Burrus. *Introduction to Wavelets and Wavelets Transforms*. Prentice Hall, 1997.

[8] P. Le Callet and D. Barba. A robust quality metric for color image quality assessment. In *ICIP03*, pages I: 437–440, 2003.

[9] M. Carnec, P. Le Callet, and D. Barba. An image quality assessment method based on perception of structural information. In *ICIP03*, pages III: 185–188, 2003.

[10] C. Chrysafis. *Wavelet Image Compression Rate Distortion Optimizations and Complexity Reductions*. PhD thesis, December 1999.

[11] C. Chrysafis and A. Ortega. Efficient context-based entropy coding for lossy wavelet image compression. In *DCC, Data Compression Conference*, Snowbird, UT, March 1997.

[12] S. Chukov. Jwic - java based wavelet image codec. *http://wavlet.chat.ru/*, 1999.

[13] D. A. Clunie. Lossless compression of grayscale medical images - effectiveness of traditional and state of the art approaches.

[14] R. Coifman and M. V. Wickerhauser. Entropy-based algorithms for best basis selection. *IEEE Transactions on Information Theory*, 38(2), March 1992.

[15] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley-Interscience, August 1991.

[16] I. Daubechies. Orthonormal bases of compactly supported wavelets. *Commun. os Pure and Appl. Math.*, 41:909–996, November 1988.

[17] I. Daubechies. Ten lectures on wavelets. In *Cbms-Nsf Regional Conference Series in Applied Mathematics*, volume 61. Society for Industrial & Applied Mathematics, 1992.

[18] I. Daubechies. *Ten Lectures on Wavelets*. Society for Industrial & Applied Mathematics, May 1992.

[19] J. Dengler, S. Behrens, and J. F. Desaga. Segmentation of microcalcifications in mammograms. *IEEE Transactions on Medical Imaging*, 12(4), December 1993.

[20] P. Fränti, J. Kivijärvi, T. Kaukoranta, and O. Nevalainen. Genetic algorithms for large-scale clustering problems. *The Computer Journal*, 40(9):547–554, 1997.

[21] D. E. Goldberg. *Genetic Algorithms in Search, Optimization & Machine Learning*. Addison-Wesley, Reading, USA, 1989.

[22] R. Gonzalez and R. Woods. *Digital Image Processing*. Addison-Wesley Publishing Company, 1992.

[23] A. Graps. An introduction to wavelets. *IEEE Computational Sciences and Engineering*, 2(2):50–61, 1995.

[24] Joint Photographic Experts Group. Official site of the joint photographic experts group: Jpeg 2000 status. *http://www.jpeg.org/jpeg2000/*, April 2005.

[25] M.A. Haque. A two-dimensional fast cosine transform. *ASSP*, 33:1532–1539, 1985.

[26] R. Hinterding, Z. Michalewicz, and A. E. Eiben. Adaptation in evolutionary computation: A survey. In *Proceedings of the 4th IEEE International Conference on Evolutionary Computation*, pages 65–69, April 1997.

[27] C. Ho, D. Hailey, R. Warburton, J. MacGregor, E. Pisano, and J. Joyce. Digital mammography versus film-screen mammography: technical, clinical and economic assessments. Technology report no 30, 2002.

[28] J. H. Holland. *Adaptation in Natural and Artificial Systems.* University of Michigan Press, Ann Arbor, USA, 1975.

[29] P. G. Howard and J. S. Vitter. Fast and efficient lossless image compression. In *Data Compression Conference*, pages 351–36, 1993.

[30] D. A. Huffman. A method for the construction of minimum-redundancy codes. In *Proc. IRE*, pages 1098–1101, September 1952.

[31] J. Kivijärvi. *Optimization Methods for Clustering.* PhD thesis, University of Turku, Department of information technology, 2004.

[32] J. Kivijärvi. *Optimization Methods for Clustering.* PhD thesis, Turku Centre for Computer Science, January 2004.

[33] J. Kivijärvi, P. Fränti, and O. Nevalainen. Self-adaptive genetic algorithm for clustering. *Journal of Heuristics*, 9(2):113–129, 2003.

[34] J. Kivijärvi, T. Ojala, T. Kaukoranta, A. Kuba, L. Nyúl, and O. Nevalainen. The comparison of lossless compression methods in the case of a medical image database. Technical Report 171, Turku Centre for Computer Science, April 1998.

[35] S. Knoblich. Gnu wavelet image codec with els coder. *http://home.t-online.de/home/stefan.knoblich/gwic_prj.html*, May 2000.

[36] S. Knoblich. Wavelet video codec based on gwic. *http://home.t-online.de/home/stefan.knoblich/gwic_codec.html*, May 2000.

[37] D. Knuth. Dynamic huffman coding. *J. Algorithms*, 2:163–180, 1985.

[38] C. Lee and O. Kwon. Objective measurements of video quality using the wavelet transform. *Optical Engineering*, 42(1):265–272, January 2003.

[39] T. S. Lee. Image representation using 2d gabor wavelets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(10):959–971, 1996.

[40] J. Lehtinen. Gwic - gnu wavelet image codec. *http://www.jole.fi/research/gwic/*, 1998.

[41] J. Lehtinen. Predictive depth coding of wavelet transformed images. In *Proceedings of SPIE: Wavelet Applications in Signal and Image Processing*, volume 3813, Denver, USA, 1999.

[42] S. P. Lloyd. Least squares quantization in pcm. *IEEE Trans. on Information Theory*, 28(2):129–137, 1982.

[43] G. Magyar, M. Johnsson, and O. Nevalainen. An adaptive hybrid genetic algorithm for the three-matching problem. *IEEE Transactions on Evolutionary Computation*, 4:135–146, 2000.

[44] S. G. Mallat. A theory for multiresolution signal decomposition: The wavelet representation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 11(7):674–693, 1989.

[45] J. L. Mannos and D. J. Sakrison. The effects of a visual fidelity criterion on the encoding of images. *IEEE Trans. Information Theory*, IT-20(4), July 1974.

[46] M. W. Marcellin, M. J. Gormish, A. B., and M. P. Boliek. An overview of jpeg-2000. In *Proc. of IEEE Data Compression Conference*, pages 523–541, 2000.

[47] J. B. McQueen. Some methods of classification and analysis of multivariate observations. In *Proc. 5th Berkeley Symposium Mathemat. Statist. Probability*, volume 1, pages 281–296, University of California, Berkeley, CA, 1967.

[48] Y. Meyer. *Wavelets and operators*, volume 37 of *Cambridge Studies in Advanced Mathematics*. Cambridge University Press, Cambridge, 1992. Translated from the 1990 French original by D. H. Salinger.

[49] M. Mitchell. *An Introduction to Genetic Algorithms*. The MIT Press, Cambridge, USA, 1996.

[50] M. Penedo, W. A. Pearlman, P. G. Tahoces, M. Souto, and J. J. Vidal. Region-based wavelet coding methods for digital mammography. *IEEE Trans. on Medical Imaging*, 22:1288–1296, October 2003.

[51] W. Pennebaker and J. Mitchell. *Jpeg : Still Image Data Compression Standard*. Van Nostrand Reinhold, 1992.

[52] C. Pereira, R. Gupta, and M. Srivastava. Pasa: A software architecture for building power aware embedded systems. In *In the proceedings of the IEEE CAS Workshop on Wireless Communications and Networking - Power efficient wireless ad hoc networks*, California, USA, September 2002.

[53] M. Pinson and S. Wolf. A new standardized method for objectively measuring video quality. *IEEE Transactions on Broadcasting*, 50(3):312–322, September 2004.

[54] A. Said and W. A. Pearlman. A new fast and efficient image codec based on set partitioning in hierarchical trees. *IEEE Transactions on Circuits and Systems for Video Technology*, 6:243–250, June 1996.

[55] K. Sayood. *Introduction to Data Compression*. Morgan Kaufmann, 1996.

[56] J. M. Shapiro. Embedded image coding using zerotrees of wavelet coefficients. *IEEE Transactions on Signal Processing*, 31(12), December 1993.

[57] J. R. Smith and S. Chang. Frequency and spatially adaptive wavelet packets. In *Proc of IEEE International Conference on Acoustics, Speech and Signal Processing*, May 1995.

[58] P. Symes. *Digital Video Compression*. McGraw-Hill, 2003.

[59] T. Szirányi. *Self-Organizing Image Fields*. PhD thesis, Magyar Tudományos Akadémia, 2001.

[60] D. S. Taubman and M. W. Marcellin. *JPEG 2000: Image Compression Fundamentals, Standards and Practice*. Kluwer Academic Publishers, Massachusetts, USA, 2002.

[61] M. Tomassini. Parallel and distributed evolutionary algorithms, 1999.

[62] M. Vettereli and J. Kovačević. *Wavelets and Subband Coding*. Prentice Hall, Englewood Cliffs, NJ, 1995.

[63] M. Weinberger, G. Seroussi, and G. Sapiro. Loco-i: A low complexity, context-based, lossless image compression algorithm. In *Proc. IEEE Data Compression Conference*, Snowbird, Utah, March–April 1996.

[64] F. W. Wheeler and W. A. Pearlman. Spiht image compression without lists. In *IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP 2000)*, Istanbul, Turkey, June 2000.

[65] I. H. Witten, R. M. Neal, and J. G. Cleary. Arithmetic coding for data compression. *Comm. of the ACM*, 6(30):520–540, 1987.

[66] Z. Xiong, K. Ramchandran, and M. T. Orchard. Space-frequency quantization for wavelet image coding. *IEEE Trans. Image Processing*, 1997.

[67] R. D. Zampolo and R. Seara. A measure for perceptual image quality assessment. In *ICIP03*, pages I: 433–436, 2003.

Publication reprints

Variable quality image compression system based on SPIHT

Antero Järvi, Joonas Lehtinen and Olli Nevalainen

Published in Signal Processing: Image Communications, vol. 14, pages 683–696, 1999 (sent 1997).

ELSEVIER

Signal Processing: *Image Communication* 14 (1999) 683–696

SIGNAL PROCESSING: IMAGE COMMUNICATION

www.elsevier.nl/locate/image

Variable quality image compression system based on SPIHT

A. Järvi[a,b,*], J. Lehtinen[a,b,1], O. Nevalainen[b]

[a] *Turku Centre for Computer Science, Lemminkäisenkatu 14 A, FIN-20520 Turku, Finland*
[b] *Department of Computer Science, University of Turku, Lemminkäisenkatu 14 A, FIN-20520 Turku, Finland*

Received 10 July 1997

Abstract

An algorithm for variable quality image compression is given. The idea is to encode different parts of an image with different bit-rates depending on their importance. Variable quality image compression (VQIC) can be applied when a priori knowledge on some regions or details being more important than others is available. Our target application is digital mammography, where high compression rates achieved with lossy compression are necessary due to the vast image sizes, while relatively small regions containing signs of cancer must remain practically unchanged. We show how VQIC can be implemented on top of SPIHT (Said and Pearlman, 1996), an embedded wavelet encoding scheme. We have revised the algorithm to use matrixes, which gives more efficient implementation both in terms of memory usage and execution time. The effect of the VQIC on the quality of compressed images is demonstrated with two test pictures: a drawing and a more relevant mammogram image.

Keywords: Image compression; SPIHT; Wavelet transform; Region of interest; Variable quality image compression (VQIC)

1. Introduction

The number of large digital image archives is increasing rapidly in many fields, including health care. The cost-efficient archiving causes a need for high-quality image compression techniques aiming at major savings in storage space and network bandwidth when transmitting the images. Despite the potentionally critical nature of medical images, some degeneration of the image quality must be allowed, since the best *lossless compression* methods can only be about half the size of a typical medical image. For example, a digital mammogram with pixel size of 50 μm is approximately of size 5000×5000 pixels with 12 bits per pixel, and thus needs about 50 Mb for storage without compression. This clearly demonstrates the need for *lossy compression*.

Lossy image compression methods are usually designed to preserve perceived image quality by removing subtle details, that are difficult to see with human eye. The frequent quality measure used for

* Corresponding author: Tel.: +358 2 3338795; e-mail: ajarvi@cs.utu.fi; fax: +3582 2410154.

[1] Joonas Lehtinen acknowledges support by the Academy of Finland.

0923-5965/99/$ - see front matter
PII: S0923-5965(98)00038-1

evaluation of the distortion in a compressed image is the *mean-square error* (MSE). However, in medical imaging, the distortion of an image is defined as the impact that compression causes to diagnostic accuracy, and finally to clinical actions taken on the basis of the image [1]. In this application area, there is an evident conflict between the two opposite goals; achieving high compression ratio and maintaining diagnostically lossless reconstruction accuracy. One possible way to alleviate this conflict is to *design an image compression method that uses lossy compression, but saves more details in important regions of the image than in other regions*. General-purpose image compression methods can also be considered to fit into this scheme; important regions and details are those, that the human visual system is sensitive to. In medical imaging, the definition of important regions involves application specific medical knowledge. Obviously, the accuracy of this knowledge is crucial for good performance of the system. If the criteria for important regions are too loose, the gain in compression ratio is lost. On the other hand, with too strict criteria the quality requirements are not met, since some important regions are treated erroneously as unimportant.

Today, the standard in lossy image compression is the JPEG [2] algorithm, which is based on the scalar quantization of the coefficients of the windowed discrete cosine transforms (DCT), followed by entropy coding of the quantization results. JPEG is generally accepted and works well in most cases, but because it uses DCT and divides the image into blocks of fixed size, it may distort or even eliminate small and subtle details. This can be a serious drawback in digital mammography, where images contain a large number of diagnostically important small low contrast details, that must preserve their shape and intensity.

Wavelet-based image compression methods are popular and some of them can be considered to be "state of the art" in general-purpose lossy image compression (see [6] for an introduction to the topic). Wavelet compression methods can be divided into three stages: *wavelet transform*, *lossy quantization* and *encoding* of the wavelet coefficients and *lossless entropy coding* [14]. The wavelet transform is used to de-correlate the coefficients representing the image. The transform collects the image energy to relatively small number of coefficients, compared to the original highly correlated pixel representation. In the quantization phase this sparse representation and dependencies between coefficients are exploited with specially tailored quantization and coding schemes. The widely know *embedded zerotree encoding* (EZW) by Shapiro [12] is an excellent example of such coding scheme, and also a good reference to wavelet based image compression is general.

One of the most advanced wavelet-based image compression techniques is SPIHT (Set Partitioning In Hierarchical Trees) by Said and Pearlman [7,11]. SPIHT is clearly a descendant of EZW using similar zerotree structure and *bitplane coding*. In bitplane coding bits of wavelet coefficients are transmitted in the order of their importance, i.e. the coefficients are encoded gradually with increasing accuracy. Because of this, the encoding can be stopped at any stage. The decoder then approximates the values of the original coefficients with precision depending on the number of bits coded for each coefficient. This property called *embedded coding* is the main reason for choosing SPIHT as the basis of the *Variable Quality Image Compression system* (VQIC). The implementation of variable quality property is straightforward in embedded coding: the encoding of the coefficients of the whole image is ceased somewhere in the middle, and subsequently only bits of coefficients that influence the important regions are encoded.

Another reason for choosing SPIHT is that it appears to perform very well in terms of compression ratio; in the case of digital chest X-rays, a compression ratio of 40:1 has been reported to cause no significant difference when compared to the original in a radiologist evaluation [4]. In an other study, SPIHT compression of mammograms to ratio 80:1 has been found to yield image quality with no statistically significant differences from the original mammogram [9]. A further indication of good performance is the fact that the output of SPIHT encoding is so dense that additional compression with lossless entropy coding gives an extra gain of only few percentages [11].

VQIC technique can be applied to any image that is spatially segmentable to a set of regions that

must be saved in better quality. We do not cover the segmentation problem in this paper since it is completely application-specific. Since we are targeting at applications where important regions are small, the segmentation is done with some kind of feature detector. The feature detection task for VQIC purpose is considerably easier compared to applications, where the interest is in the presence or absence of the feature. In VQIC, a moderate number of false positive detections can be tolerated, as long as all of the true features are detected. Thus, most existing feature detection algorithms are suitable, because they can be tuned to be oversensitive. Several suitable fully automatic segmentation methods exists for this purpose in the field of medical imaging [5], especially for mammography [3].

Before describing further details, we briefly discuss relevant research. VQIC has been used in the compression of image sequences in video conferencing [10]. In this work, the pixel values are predicted and the prediction errors are transformed by 2D-DCT. Coefficients of the blocks with minor importance are quantized with coarser level than more important details, heads and shoulders. In another research, the importance of a region is determined on the basis of the visibility of distortions to human eye [8]. This information is used in the construction of a constant quality MPEG stream by adjusting quantization parameters defined by the MPEG standard. The focus in both papers is the segmentation of important regions, and VQIC is achieved by variable quantification of DCT coefficients. A recent paper by Shin et al. describes a selective compression technique, which integrates detection and compression algorithms into one system [13]. The compression technique used is intraband coding of wavelet packet transform coefficients, where variable quality is achieved by scaling the wavelet coefficients in important regions to increase their priority in coding. The method is tested with a digital mammogram, where detected microcalcifications are considered as ROIs. Even though the aims of this work are close to ours, the actual methods differ considerably.

Our work is organized as follows. In Section 2 we introduce an algorithm called *variable quality SPIHT* (*vqSPIHT*), which is basically a reimplementation of SPIHT, with the added VQIC functionality. We revise the memory organization of SPIHT to use matrix-based data structures. This new implementation reduces the working storage requirements of the algorithm considerably. In Section 3 we discuss the compression performance of vqSPIHT algorithm, and show with two examples that it can be superior to SPIHT or JPEG. We first demonstrate this with a set of details in a high contrast drawing compressed to several bit-rates with all three compression algorithms. We also discuss a more relevant application for VQIC – compression of digital mammograms, and make a comparison between SPIHT and vqSPIHT compressed images.

2. vqSPIHT algorithm

In this section, we explain informally the basic ideas behind SPIHT and vqSPIHT algorithms to facilitate the reading of the vqSPIHT algorithm in a pseudo-code format. We also discuss the implementation based on matrix data structures and its implications to practical memory requirements.

2.1. Structure of the wavelet transformed coefficient table

Wavelet transform converts an image into a coefficient table with approximately the same dimensions as the original image. Fig. 1 shows the structure of the *wavelet coefficient table*, which contains three wavelet coefficient pyramids (pyramids *A*, *B* and *C*) and one table of *scaling coefficients S*. The scaling coefficients represent roughly the mean values of larger parts of the image and wavelet coefficient details of various sizes. Since in practice the transform is stopped before scaling table *S* would shrink to a single coefficient, table *S* looks like a miniature version of the original image. The top levels of the three wavelet pyramids are located adjacent to the scaling table (level three in Fig. 1), and contain coefficients representing large details, whereas coefficients at level zero contribute mainly to the smallest details in the image. Pyramid *A*

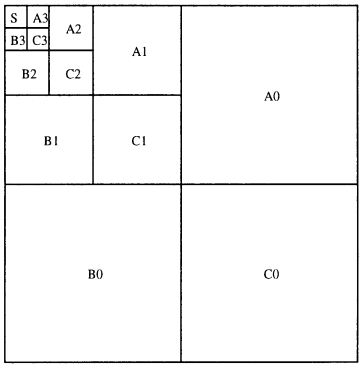


Fig. 1. An example of a wavelet coefficient table that contains three four-level pyramids: *A*, *B* and *C*. Scaling coefficients are located on the square noted by *S*.

contains coefficients for vertical details and pyramid *B* respectively for horizontal details. The third pyramid *C* contains correction coefficients needed in the reconstruction of the image from pyramids *A*, *B* and table *S*.

In the SPIHT algorithm, the pyramids are divided into *sub-pyramids* which corresponds to zerotrees in EZW. A sub-pyramid has a top element somewhere in the coefficient table, and contain four coefficients one level lower in the corresponding spatial location in the same pyramid, 16 elements two levels lower, and so on. The sub-pyramids are extended to the scaling coefficient *S* in the following way. The scaling coefficients are grouped into groups of four. The coefficient in the upper left corner has no descendants, whereas the three remaining coefficients in the group (upper right corner, lower left corner and lower right corner) serve as top elements of three sub-pyramids in pyramids *A*, *B* and *C* in corresponding order.

In our approach to VQIC, we must determine which coefficients contribute to the value of a given pixel in the original image. In an *octave-band decomposition*, which we use, a coefficient of any pyramid in higher level corresponds to four coefficients in the next level at same spatial location. We use this rule of multiples of four in choosing important coefficients.

2.2. The basic functioning of SPIHT

To understand how SPIHT works, one must keep in mind that it is not a complete image compression scheme, but a method tailored for optimal embedded encoding of wavelet transformed image coefficients. The encoding is optimal in the sense of MSE. SPIHT does not presuppose any particular wavelet transform. The only requirement is that the transform has the octave band decomposition structure, as described above. Also, the optimal encoding with respect to MSE is achieved only if the transform has the energy conservation property. In the implementation of vqSPIHT, we use the same biorthogonal B97 wavelet transform [14], that is used in the SPIHT.

2.2.1. Bitplane coding in SPIHT

Optimal progressive coding of SPIHT is implemented with a *bitplane coding scheme*. The order of coding is based on the energy saving property stating that the larger the wavelet coefficient is, the more its transmission reduces the MSE. Furthermore, since SPIHT uses uniform scalar quantization, transmission of a more significant bit in any coefficient reduces the MSE more than transmission of a less significant bit in a possibly larger coefficient [11].

According to this principle, all coefficients are sorted to a decreasing order by the *number of significant bits*. The number of significant bits in the coefficient having the largest absolute values is noted by *n*. The output is generated by transmitting first all the *n*th bits in coefficients that have at least *n* significant bits, then $(n-1)$th bits of coefficients that have at least $(n-1)$ significant bits, and so on. Because the most significant bit of a coefficient is always one, the sign of the coefficient is transmitted in place of the most significant bit.

In addition to transmitted bitplanes, the sorting order and the length of each bitplane are needed in the decoder to resolve the location of each transmitted bit in the reconstructed wavelet coefficient table. This information is not transmitted explicitly, instead the same algorithm is used in both the encoder and the decoder, and all branching decisions made in the encoder are transmitted to the

decoder. The branching decisions are transmitted interleaved with the bitplane coded bits and the signs of coefficients. Because of progressive nature of SPIHT coding, the transmitted bit-stream can be truncated at any point and the original coefficient matrix approximated with optimal accuracy with respect to the number of transmitted bits. For each coefficient, the most significant not transmitted bit is set to one, and the rest to zero, thus achieving a good approximation in uniform quantization.

2.2.2. Exploitation of the pyramid structure of wavelet transform

An important property in most natural images is that the low- and high-frequency components are spatially clustered together. This means in the wavelet coefficient pyramid, that there is high correlation between the magnitudes of the coefficients of different levels in corresponding locations. Also, since the variance of the frequency components tends to decrease with increasing frequency, it is very probable that the coefficients representing fine details in a particular spatial location will be small, if there is a region of small coefficients in the corresponding location on coarser level of the pyramid. Thus, it is probable that there exists sub-pyramids containing only zeroes on the current bitplane. These *zerotrees* can be encoded with one bit, thus cutting down the number of sorting decisions considerably and also the branching decisions that must be transmitted to the decoder. The way these dependencies between coefficients are exploited in SPIHT coding is described in the presentation of vqSPIHT algorithm.

2.3. The vqSPIHT algorithm

2.3.1. Extension of SPIHT to vqSPIHT

To expand the SPIHT algorithm to vqSPIHT, we define a *Region Of Interest* (ROI) as a region in the image that should be preserved in better quality than the rest of the image. ROIs can be presented as a binary map that is highly compressible with simple run-length encoding and thus does not affect bit-rate significantly.

In selective coding mode of vqSPIHT, only coefficients affecting ROIs are coded. This mode is triggered when a certain *percentage* α *of the wanted final output file size* has been reached. The choice of α is important, and its best value is highly application-dependent. Some applications might demand more sophisticated definition of α depending on the file size, area of the ROIs and some indicator on how ROIs are scattered, for example.

To implement selective coding, we construct a *look-up-table* (LUT), that is used in the function *influence*(i,j) defining whether a coefficient (i,j) contributes to any ROI or not. The LUT is constructed by scaling each level of all three pyramids to the same size as the original image, and comparing the map of ROIs to the scaled levels. All the coefficients that overlap with any ROI are marked in the LUT.

2.3.2. Implementation with matrices

Instead of lists that are used in the original implementation of SPIHT, our implementation of vqSPIHT uses two matrices for keeping track of significant and insignificant coefficients and sub-pyramids. With the matrix data structures, we can considerably reduce the working storage requirements of encoding and decoding.

We introduce a *point significance matrix* (PSM) to indicate whether a coefficient is known to be *significant*, *insignificant* or still has an *unknown* state. The labels of PSM are coded with two bits and the dimensions of the PSM are the same as in the coefficient table. We also need a *sub-pyramid list matrix* (SPLM), which is used for maintaining an implicit list of the sub-pyramids containing only insignificant coefficients. The list structure is needed, because the order of sub-pyramids must be preserved in the sorting algorithm. The dimensions of the SPLM are half of the dimensions of the coefficients matrix, because the coefficients on the lowest level of the pyramids cannot be top elements of sub-pyramids. There are two types of sub-pyramids. A sub-pyramid of type A contains all descendants of a particular coefficient, excluding the top coefficient itself. A sub-pyramid of type B is otherwise similar, but the immediate offspring of the top coefficient is excluded in addition to the top coefficient. The list structure with SPLM is simple: the lower bits of an element tells the index of the next element in the list. The type of the sub-pyramid is coded with the highest bit.

2.3.3. Pseudo-code of vqSPIHT

The algorithms for the encoder and the decoder are similar. We use notation 'input/output x', which consists of two steps: in the encoder x is first calculated and then transmitted to entropy coder; in the decoder x received from decoder and then used in the construction of a new estimate for the coefficient. Variable *nbc* indicates the number of bits transmitted or received thus far. Constant *filesize* indicates the requested final file size in bits.

Function influence (i,j) is defined to be *true*, if the element (i,j) in the LUT is marked to influence an ROI, and *false* otherwise. Let $c_{i,j}$ be the value of the coefficient (i,j) in the wavelet coefficient table and the coordinate pair (i,j) denote either a single coefficient or a whole sub-pyramid of type A or B having coefficient at (i,j) as the top element. The meaning of (i,j) will be evident from the context. Finally, we define the significance of a coefficient with function $S_n(c_{i,j})$ as follows: $S_n(c_{i,j}) = 1$ if the number of bits after the first 1-bit in the absolute value of $c_{i,j}$ is at least $n-1$, otherwise $S_n(c_{i,j}) = 0$. The significance of a sub-pyramid is defined with function $S_n(i,j)$. $S_n(i,j) = 0$ if $S_n(c) = 0$ for all coefficients c belonging to sub-pyramid (i,j), otherwise $S_n(i,j) = 1$.

The input for both the encoder and decoder is the wavelet coefficient table, the ROI map, α and filesize. The output of the decoder is an approximation of the original wavelet coefficient table. See Fig. 2 for an outline of the algorithm.

1. *Initialization*

- Input/output n, which is the number of significant bits in the coefficient having the largest absolute value.
- Construct LUT according to the given ROIs.
- Set $nbc = 0$.
- Set the PSM label of all scaling coefficients (coefficients in the area S in Fig. 1) to *insignificant* and the PSM label of all other coefficients to *unknown*.
- Create a list of all scaling coefficients that have descendants in SPLM and make them of type A.

2. *Sorting step for PSM*

- For every element (i,j) in PSM do
  - If (*nbc*/*filesize* $< \alpha$ OR *influence* (i,j)) then
    - If (i,j) is labeled to be *insignificant* do:
      - Input/output $S_n(c_{i,j})$.
      - If $S_n(c_{i,j}) = 1$ then set the PSM label (i,j) to *significant* and input/output the sign of $c_{i,j}$.
    - Else If (i,j) is labeled to be *significant* do:
      - Input/output the n-th most significant bit of $|c_{i,j}|$.

3. *Sorting step for SPLM*

- For each element (i,j) in the list in SPLM do:
  - If sub-pyramid (i,j) is of type A AND (*nbc*/*filesize* $< \alpha$ OR *influence* (i,j)) then
    - Input/output $S_n(i,j)$.
    - If $S_n(i,j) = 1$ then
      - For each (k,l) belonging to immediate offspring of (i,j) do:
        - If (*nbc*/*filesize* $< \alpha$ OR *influence* (k,l)) then
          - Input/output $S_n(c_{k,l})$.
          - If $S_n(c_{k,l}) = 1$ then set the PSM label (k,l) to *significant* and input/output the sign of $c_{k,l}$, else set the PSM label (k,l) to *insignificant*.
      - If (i,j) is not on one of the two lowest levels of the pyramid then move (i,j) to the end of the list in SPLM and change its type to B, else remove (i,j) from the list in SPLM.
  - If sub-pyramid (i,j) is of type B AND (*nbc*/*filesize* $< \alpha$ OR *influence* (i,j)) then
    - Input/output $S_n(i,j)$.
    - If $S_n(i,j) = 1$ then
      - For each (k,l) belonging to immediate offspring of (i,j) do:
        - If (*nbc*/*filesize* $< \alpha$ OR *influence*(k,l)) then add (k,l)) to the end of list in SPLM as sub-pyramid of type A.
      - Remove (i,j) from the list in SPLM.

4. *Quantization-step update*

- If $n > 0$ then
  - Decrement n by 1.
  - Jump to the beginning of the PSM sorting step 2.

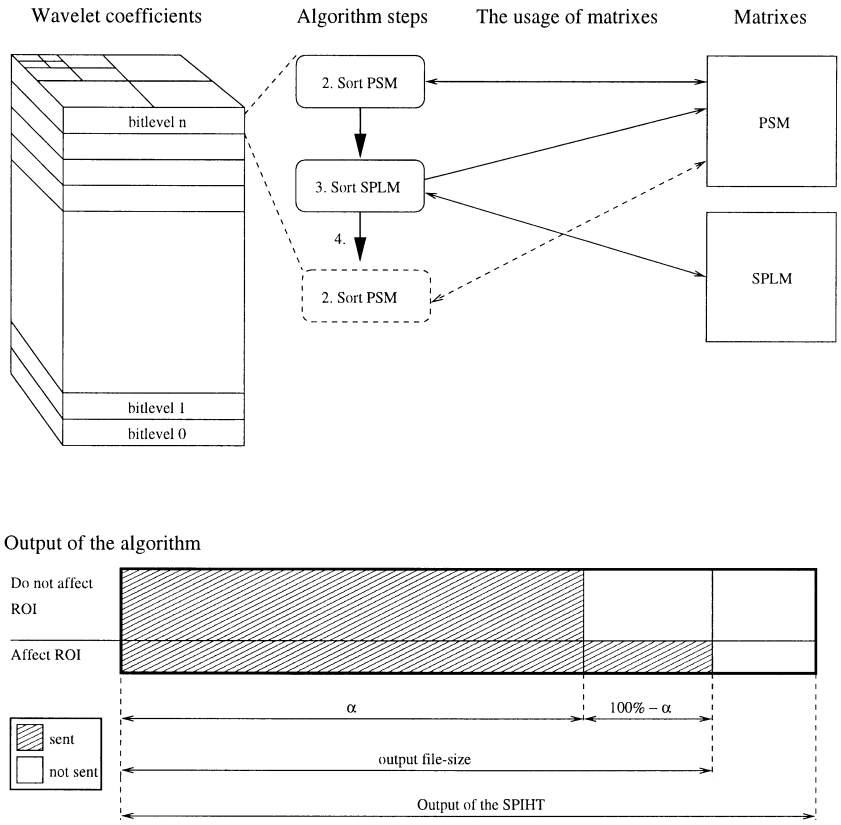


Fig. 2. The vqSPIHT matrix implementation is illustrated on the upper half of the picture: The wavelet coefficients on the left are processed one bitlevel at a time. Each bitplane is processed in two steps (2 and 3). The first step tests the significance of each coefficient that is labeled to be insignificant in PSM matrix and transmits the necessary information. The second step processes all the trees in SPLM and sets new points in PSM as *significant* or *insignificant*. The bottom part of the picture illustrates the thresholding of transmitted data. After the alpha limit has been reached, only bits of coefficients and decision information that affects the ROI is sent.

The *original SPIHT* algorithm always transmits bits in two alternating phases: In the first phase the branching decisions of the sorting step and the signs of new significant coefficients are transmitted. In the second phase the bits of all significant coefficients on the current bitplane are transmitted. Interrupting the first phase can cause transmission of branching decisions that cannot be used in reconstruction. *In vqSPIHT*, we have therefore combined the sending of the significant coefficients to the sorting phase to avoid the problem.

3. Test results

3.1. Numerical quality indicators

As a measure of image quality, we use the point signal to noise ratio (PSNR) for the whole image and for the ROIs:

$$D_{\mathrm{PSNR}} = 10\log_{10}\frac{2^{\mathrm{bpp}}-1}{D_{\mathrm{MSE}}}\mathrm{dB}.$$

It is well-known that PSNR does not give objective estimate of image quality, but it correlates with the amount of distortion in the image and it can be used for comparing the quality of images compressed with algorithms causing similar distortion. As a measure for the amount of compression we use the number of *bits per pixel* (bpp).

3.2. The comic test image

The vqSPIHT algorithm was constructed for the needs of digital mammography. However, mammograms are rather smooth and thus easily hide compression artifacts. To better illustrate the effect of VQIC, we use a comic picture of size 420×480 with 8 bpp as the first test image (Fig. 3). GIF,[2] JPEG, SPIHT and vqSPIHT algorithms are used to compress the image having three ROIs covering 2.4% of the image marked on the Fig. 3. The compression results are presented in Table 1 and Fig. 4.

As seen in Table 1, the PSNR of SPIHT and vqSPIHT are similar. With less compression (large bpp), JPEG is also comparable in terms of PSNR, but its visual quality decreases rapidly with decreasing values of bpp (Fig. 4). IN JPEG, the resulting file size cannot be specified exactly in advance, and thus the bpp values of JPEG are slightly different from those of SPIHT and vqSPIHT. There are no big differences in the performance of these techniques, when only the overall PSNR is evaluated.

When considering the PSNR of ROIs, the situation changes radically. SPIHT performs significantly better than JPEG, but the improvement achieved with vqSPIHT is even greater. We have used quite high α values: 80% and 90% of the size of the output file. Even with these values, PSNR in the ROIs is considerably lower in the vqSPIHT compressed images than in the SPIHT compressed images, while good overall quality (PSNR of whole image) is still maintained. This is partly due to the fact that the coefficients that influence the ROIs also contribute to the areas outside the ROIs. Thus, the overall image quality is still improving outside the ROIs after the trigger value α has been reached.

Fig. 4 shows a 88×66 pixel region taken from the comic image and compressed with JPEG, SPIHT and vqSPIHT with different bpp values. The original part of the image is shown in the lower right corner. The selected part includes an ROI, marked on the original image.

Fig. 3. The original comic test image with three ROIs marked.

[2] A common lossless image compression method.

Table 1
Comparison between SPIHT, vqSPIHT and JPEG for the comic image (Fig. 3)

Method	bpp	PSNR	PSNR in ROIs
GIF	4.456	∞	∞
JPEG	1.01	26.01	23.76
JPEG	0.50	23.09	20.21
JPEG	0.29	20.43	17.81
SPIHT	1.00	27.56	26.58
SPIHT	0.50	24.06	22.13
SPIHT	0.25	21.00	18.63
vqSPIHT $\alpha = 90\%$	1.00	27.26	39.85
vqSPIHT $\alpha = 90\%$	0.50	23.83	29.76
vqSPIHT $\alpha = 90\%$	0.25	20.75	23.41
vqSPIHT $\alpha = 80\%$	1.00	26.72	42.46
vqSPIHT $\alpha = 80\%$	0.50	23.40	34.36
vqSPIHT $\alpha = 80\%$	0.25	20.37	26.85
vqSPIHT $\alpha = 80\%$	0.15	18.55	22.13
vqSPIHT $\alpha = 80\%$	0.10	17.45	18.94
vqSPIHT $\alpha = 80\%$	0.05	15.15	14.55

Fig. 4. A region of the comic test image containing an ROI compressed with JPEG, SPIHT and vqSPIHT. See the table below for the explanation of sub-regions of the figure:

JPEG 1.00 bpp	JPEG 0.50 bpp	JPEG 0.25 bpp
SPIHT 1.00 bpp	SPIHT 0.50 bpp	SPIHT 0.25 bpp
vqSPIHT α90% 1.00 bpp	vqSPIHT α90% 0.50 bpp	vqSPIHT α90% 0.25 bpp.
vqSPIHT α80% 1.00 bpp	vqSPIHT α80% 0.50 bpp	vqSPIHT α80% 0.25 bpp
vqSPIHT α80% 0.15 bpp	vqSPIHT α80% 0.10 bpp	ROI in original

The first row of Fig. 4 shows the limit bpp value, where JPEG clearly fails to produce acceptable quality. The image compressed to 0.50 bpp is still recognizable, but the 0.25 bpp image is not. Even the 1.00 bpp image compressed with JPEG has high-frequency noise around the sharp edges. In the 0.5 bpp and 0.25 bpp images the blocking effect introduces additional artifacts. In the 1.00 bpp SPIHT image there is no high-frequency noise. When the bpp value gets smaller, the image gets smoother, and it thus loses small high-frequency details. However, even the 0.25 bpp SPIHT image is recognizable.

The overall image quality of the 90% and 80% 1.00 bpp vqSPIHT images is very close to that of the 1 bpp SPIHT image. The visual quality of the

0.50 bpp SPIHT image is similar to the 0.25 bpp vqSPIHT ($\alpha = 80\%$) image on the ROI. Note that the ROI of the 0.10 bpp vqSPIHT image is visually better than the ROI on the 0.25 bpp JPEG image, and of comparable quality with ROI on the 0.25 bpp SPIHT image. In this image, 0.25 bpp corresponds to compression ratio 32:1. It should be noted that SPIHT is designed to perform well on natural images. A comic drawing is a difficult case for SPIHT and thus also for vqSPIHT.

The performance of the vqSPIHT was good when ROI covered only 2.4% of the picture. With the increase of ROI α must decrease to compensate the larger number of coefficients in the ROI in order to maintain the same quality. Because bits are coded in the order of their importance, the bits used in coding of ROI can add considerably less to whole image PSNR than the bits outside ROI. As seen in Fig. 5, the benefits of VQIC rapidly disappear with large ROI.

3.3. The mammogram test image

The second test image (Fig. 6) is a mammogram of size 2185×2925 with 12 bpp. The mammogram test image has been compressed only with vqSPIHT. However, the setting of α to 100% makes vqSPIHT function similarly to SPIHT.

In this example, we assume that the micro-calcifications are the only important diagnostic details of a mammogram that are easily lost in compression. Note that in a study of the applicability of

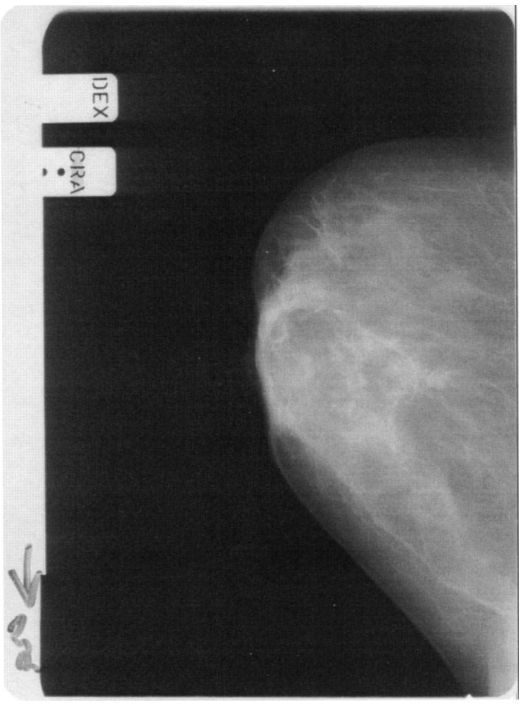


Fig. 6. Original mammogram test image.

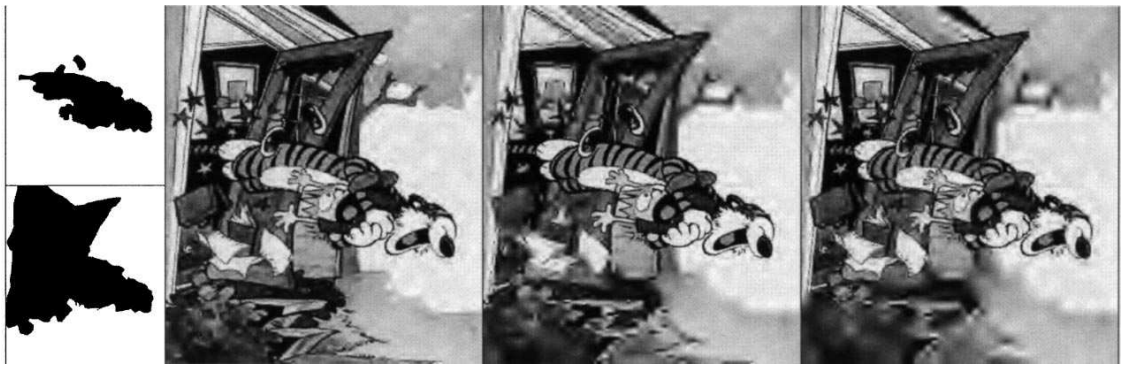


Fig. 5. The two images on the left are ROI masks used in the compression of the two rightmost images, where ROI covers 16% (the upper image) and 46% of the image. The first gray-scale image is compressed with without ROI ($\alpha = 100\%$), while 16% ROI ($\alpha = 50\%$) is used in the second image and 46% ROI ($\alpha = 50\%$) in the last image. All the images are compressed with same 0.25 bpp bit-rate.

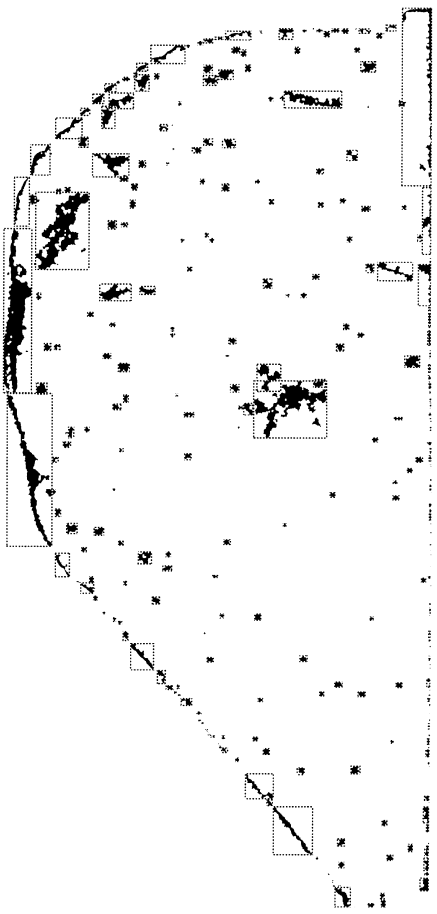


Fig. 7. Micro-calcifications found in the test mammogram (shown as black), and the ROIs (dotted rectangles around micro-calcifications).

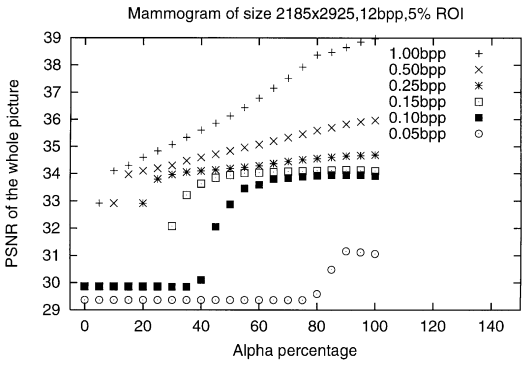


Fig. 8. PSNR of the whole mammogram for vqSPIHT as a function of α and bpp.

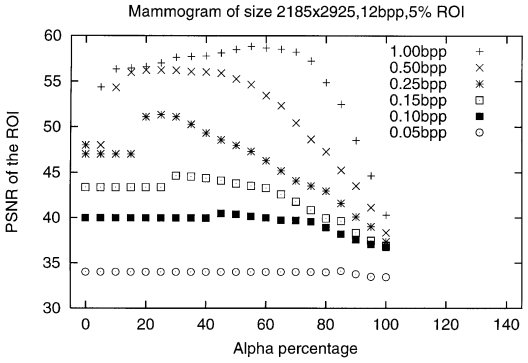


Fig. 9. PSNR of the ROIs in the mammogram for vqSPIHT as a function of α and bpp.

vqSPIHT to digital mammograms also other signs of cancer, like stellate lesions and nodules, should be considered. A micro-calcification location map, shown in Fig. 7, was generated with a micro-calcification detection algorithm slightly modified from the morphological segmentation algorithm of Dengler et al. [3]. The detection was tuned to be oversensitive to make sure that all micro-calcifications were detected. Because of this, the algorithm detected also a large number of false calcifications, including the skin-line of the breast. In this test case, there were 323 ROIs covering 5% of the whole mammogram.

We used the bpp values 0.05, 0.10, 0.15, 0.25, 0.50, 0.75, 1.00 and let α take values of 30, 40, 50, 60, 70, 80, 90 and 100 percent of the resulting file size. Fig. 8 shows the PSNR of the whole image as a function of α and the bpp. Lowering α decreases the PSNR of the whole image, but the effect remains moderate with reasonable α values.

Fig. 9 shows the PSNR calculated only on the ROI as a function of α and the bpp. The benefits of the VQIC on ROIs is clearly seen in comparison with the Fig. 8. To point out, the PSNR of ROIs in 1.00 bpp mammogram jumps from 40.29 to

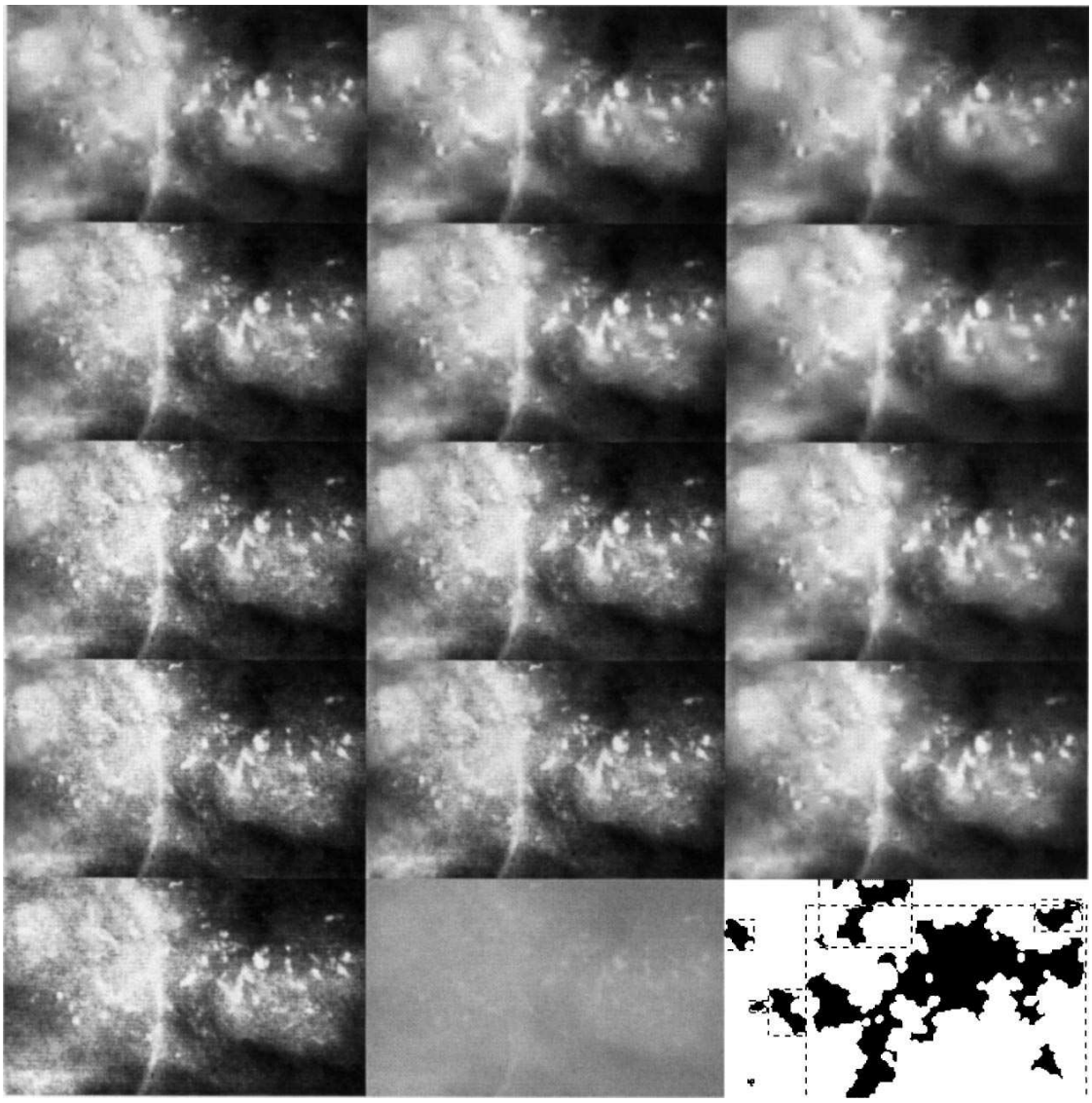


Fig. 10. A region of vqSPIHT compressed mammograms with different α and bpp rates. Bpp values on the columns from left to right are 1.00, 0.50 and 0.15. The values of α starting from the uppermost row are 100%, 90%, 70% and 50%. All the images have been histogram equalized to ease the evaluation. The three images in the last row from left to right are: the histogram equalized uncompressed region, original uncompressed region and a bit map of the detected micro-calcifications with the ROIs marked.

54.87 dB when α decreases from 100% (i.e. SPIHT) to 80%. This causes a very moderate change in the PSNR of the whole image, which decreases from 38.98 to 38.38.

Fig. 10 shows a region containing a micro-calcification cluster taken from a mammogram, that has been compressed using various bpp and α values. A visual comparison between SPIHT and

vqSPIHT shows that the mammogram can be compressed to a significantly lower bpp value with vqSPIHT than with SPIHT ($\alpha = 100\%$) to achieve similar preservation of micro-calcifications in the ROIs. The region in the upper left corner has been compressed with SPIHT to compression ratio 12:1. Even with this rather modest compression, a comparison with the original (lower left corner) reveals that the edges of the calcifications have become blurred, some small calcifications have disappeared and some have merged together. When keeping the same bpp 1.00, we notice that setting $\alpha = 70\%$, the micro-calcifications are virtually indistinguishable from the original. With this choice of α, the PSNR of whole image decreases from 38.98 to 37.51 dB. Now, keeping $\alpha = 70\%$ the bpp value 0.15 (compression ratio 1:80) gives a visually comparable reconstruction to the 1.00 bpp SPIHT image (compression ratio 1:12). In this case, the PSNR of the whole mammogram decreases to 34.08 dB. This is, however, virtually same as the PSNR of SPIHT 0.15 bpp compressed image, which is 34.10 dB.

3.4. Practical memory requirements of the implementation

We first implemented the algorithm using the list data structures of the original SPIHT algorithm [11], but found that this required a large amount of internal memory. The amount of memory needed was very dependent on the values of bpp and α. Typically, the compression of a 12 MB mammogram required at least 120 MB of internal memory during encoding, but with some combinations of bpp and α the memory requirement was considerably larger. The memory is mainly used for representing the coefficient table and the lists that are constantly scanned through. Thus, paging the memory to hard disk increases the execution time drastically. Memory requirements can be made independent of the bpp-ratio and α by reimplementing the algorithm using the matrix data structures presented previously. The working memory space dropped to about 50 MB and about 40% of that could be paged to disk without significant increase of the execution time. All of the needed memory could be allocated once, which made the memory management efficient in comparison to the slow per-node dynamical memory management of explicit list structures.

4. Summary and conclusions

The idea of VQIC is to use more bits for important details at the cost of unimportant details such as noise. The compression method can be applied in applications where certain small regions in the image are especially important. We have shown that in our target application, compression of digital mammograms, the variable quality compression scheme can improve the compression efficiency considerably. The variable quality property has been integrated into SPIHT, which is one of the best general-purpose compression techniques. We have also simplified the implementation of SPIHT and reduced working storage requirements significantly compared to the original implementation. Our version of the algorithm allows the compression of large images such as mammograms with a standard PC. A research on the clinical applicability of the VQIC techniques in the context of very large digital mammogram archive is planned.

Acknowledgements

The authors would like to thank M.Sc J. Näppi for providing the micro-calcification detection software.

References

[1] C.N. Adams, A. Aiyer, B.J. Betts et al., Image quality in lossy compressed digital mammograms, in: Proc. 3rd Internat. Workshop in Digital Mammography, Chicago, USA, 1996.
[2] V. Bhaskaran, K. Konstantinides, Image and Video Compression Standards, Kluwer Academic Publishers, Dordrecht, The Netherlands, 1995, Chapter 5.
[3] J. Dengler, S. Behrens, J.F. Beraga, Segmentation of micro calcifications in mammograms, IEEE Trans. Med. Imaging 12 (4) (December 1993).

[4] B.J. Erickson, A. Manduca, K.R. Persons, Clinical evaluation of wavelet compression of digitized chest X-rays, in: Proc. SPIE 3031 Medical Imaging: Image Display, Newport Beach, CA, 1997.
[5] M. Giger, H. MacMahon, Image processing and computer-aided diagnosis, Radiologic Clinics of North America 34 (3) (May 1996) 565–596.
[6] M. Hilton, B.D. Jawerth, A.N. Sengupta, Compressing still and moving images, Multimedia Systems 2 (December 1994) 218–227.
[7] A. Manduca, A. Said, Wavelet compression of medical images with set partitioning in hierarchical trees, in: Proc. SPIE 2704 Medical Imaging: Image Display, Newport Beach, CA, 1996.
[8] P.J. Meer, R.L. Lagendijk, J. Biemond, Local adaptive thresholding to reduce the bit rate in constant quality MPEG coding, in: Proc. Internat. Picture Coding Symp., Melbourne, Australia, 1996.
[9] S.M. Perlmutter, P.C. Cosman, R.M. Gray, R.A. Olshen, D. Ikeda, C.N. Adams, B.J. Betts, M.B. Williams, K.O. Perlmutter, J. Li, A. Aiyer, L. Fajardo, R. Birdwell, B.L. Daniel, Image quality in lossy compressed digital mammograms, Signal Processing 59 (2) (June 1997) 189–210.
[10] R. Plompen, J. Groenveld, F. Booman, D. Boekee, An image knowledge based video codec for low bitrates, in: Proc. SPIE 804 Advances in Image Processing, 1987.
[11] A. Said, W.A. Pearlman, A new fast and efficient image codec based on set partitioning in hierarchical trees, IEEE Trans. Circuits Systems Video Technol. 6 (June 1996) 243–250.
[12] J.M. Shapiro, Embedded image coding using zerotrees of wavelet coefficients, IEEE Trans. Signal Process. 31 (12) (December 1993).
[13] D. Shin, H. Wu, J. Liu, A region of interest (ROI) based wavelet compression scheme for medical images, in: Proc. SPIE 3031 Medical Imaging: Image Display, Newport Beach, CA, 1997.
[14] M. Vetterli, J. Kovačević, Wavelets and Subband Coding, Prentice-Hall, Englewood Cliffs, NJ, 1995.

Limiting distortion of a wavelet image codec

Joonas Lehtinen

Published in Acta Cybernetica, vol. 14, no. 2, pages 341–356, 1999.

Acta Cybernetica **14** (1999) 341-356

Limiting Distortion of a Wavelet Image Codec

Joonas Lehtinen *

Abstract

A new image compression algorithm, *Distortion Limited Wavelet Image Codec* (DLWIC), *is introduced*. The codec is designed to be *simple to implement*, *fast* and have *modest requirements for the working storage*. It is shown, *how the distortion of the result can be calculated while progressively coding a transformed image* and thus how the *mean square error of the result can be limited* to a predefined value. The DLWIC uses zerotrees for efficient coding of the wavelet coefficients. Correlations between different orientation components are also taken into account by binding together the coefficients on the three different orientation components in the same spatial location. The maximum numbers of significant bits in the coefficients of all subtrees are stored in two-dimensional heap structure that allows the coder to test the zerotree property of a subtree with only one comparison. The compression performance of the DLWIC is compared to the industry standard JPEG compression and to an advanced wavelet image compression algorithm, vqSPIHT. An estimation of execution speed and memory requirements for the algorithm is given. The compression performance of the algorithm seems to exceed the performance of the JPEG and to be comparable with the vqSPIHT.

1 Introduction

In some digital image archiving and transferring applications, especially in medical imaging, the quality of images must meet predefined constrains. The quality must be often guaranteed by using a lossless image compression technique. This is somewhat problematic, because the compression performance of the best known lossless image compression algorithms is fairly modest; the compression ratio ranges typically from 1:2 to 1:4 for medical images [5].

Lossy compression techniques generally offer much higher compression ratios than lossless ones, but this is achieved by losing details and thus decreasing the quality of the reconstructed image. Compression performance and also the amount of distortion are usually controlled with some parameters which are not directly connected to image quality, defined by *mean square error* [1] (MSE). If a lossy technique is used, the quality constrains can be often met by overestimating the control parameters, which results worse compression performance.

*Turku Centre for Computer Science, University of Turku, Lemminkäisenkatu 14 A, 20520 Turku, Finland, email: jole@jole.fi, WWW: http://jole.fi/

In this paper a new lossy image compression technique called *Distortion Limited Wavelet Image Codec* (*DLWIC*) is presented. The DLWIC is related to *embedded zerotree wavelet coding* (EZW) [9] technique introduced by J.M. Shapiro in 1993. Also some ideas from SPIHT [8] and vqSPIHT [3] have been used. DLWIC solves the problem of *distortion limiting* (DL) by allowing the user of the algorithm to specify the *MSE of the decompressed image as controlling parameter for the compression algorithm*.

The algorithm is designed to be *as simple as possible*, which is achieved by *binding together the orientation bands of the octave band composition* and *coding the zerotree structures and wavelet coefficient bits in the same pass*. A special auxiliary data structure called *two dimensional heap is introduced* to make the zerotree coding simple and fast. The DLWIC uses only little extra memory in the compression and is thus suitable for compression of very large images. The technique also seems to provide competitive compression performance in comparison with the vqSPIHT.

In the DLWIC, the image to be compressed is first converted to the wavelet domain with the orthonormal *Daubechies wavelet transform* [10]. The transformed data is then coded by bit-levels using a scanning algorithm presented in this paper. The output of the scanning algorithm is coded using *QM-coder* [7], an advanced binary arithmetic coder.

The scanning algorithm processes the bits of the wavelet transformed image data in decreasing order of their significance in terms of MSE, as in the EZW. This produces *progressive* output stream: the algorithm can be stopped at any phase of the coding and the already coded output can be used to construct an approximation of the original image. This feature can be used when a user browses images using slow connection to the image archive: The image can be viewed immediately after only few bits have been received; the subsequent bits then make it more accurate. The DLWIC uses the progressivity by stopping the coding when the quality of the reconstruction exceeds threshold given as a parameter to the algorithm. The coding can also be stopped when the size of the coded output exceeds a given threshold. This way both the MSE and *bits per pixel* (BPP) value of the output can be accurately controlled.

After the introduction, the structure of the DLWIC is explained. A quick overview of the octave band composition is given and it is shown with an example how the wavelet coefficients are connected to each other in different parts of the coefficient matrix.

Some general ideas of the bit-level coding are then explained (2.3) and it is shown how the unknown bits should be approximated in the decoder. The meaning of zerotrees in DLWIC is then discussed (2.4). After that an auxiliary data structure called two dimensional heap is introduced (2.5). The scanning algorithm is given as pseudo code (2.6).

The distortion limiting feature is introduced and the stopping of the algorithm on certain stopping conditions is discussed (2.7). Finally we show how separate probability distributions are allocated for coding the bits with the QM-coder in different contexts (2.8).

The algorithm is tested with a set of images and the compression performance

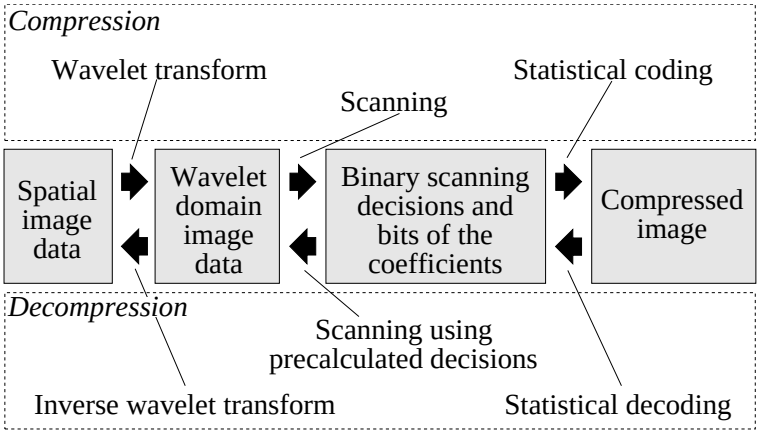


Figure 1: The structure of the DLWIC compression algorithm

is compared to the JPEG and the vqSPIHT compression algorithms (3). Variations in the quality achieved by the constant quantization in the JPEG is demonstrated with an example. Also an estimation of the speed and memory usage is given (3.2).

2 DLWIC algorithm

2.1 Structure of the DLWIC and the wavelet transform

The DLWIC algorithm consists of three steps (Figure 1): 1) the wavelet transform, 2) scanning the wavelet coefficients by bit-levels and 3) coding the binary decisions made by the scanning algorithm and the bits of the coefficients with the statistical coder. The decoding algorithm is almost identical: 1) binary decisions and coefficient bits are decoded, 2) the coefficient data is generated using the same scanning algorithm as in the coding phase, but using the previously coded decision information, 3) the coefficient matrix is converted to a spatial image with the inverse wavelet transform.

The original spatial domain picture is transformed to the wavelet domain using Daubechies wavelet transform [10]. The transform is applied recursively to the rows and columns of the matrix representing the original spatial domain image. This operation gives us an octave band composition (Figure 2). The left side (B) of the resulting coefficient matrix contains horizontal components of the spatial domain image, the vertical components of the image are on the top (A) and the diagonal components are along the diagonal axis (C). Each *orientation pyramid* is divided to levels, for example the horizontal orientation pyramid (B) consists of three levels (B0, B1 and B2). Each level contains details of different size; the lowest level (B0), for example, contains the smallest horizontal details of the spatial image. The three orientation pyramids have one shared top level (S), which contains

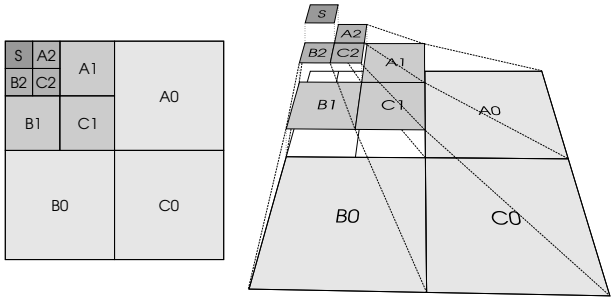


Figure 2: Octave band composition produced by recursive wavelet transform is illustrated on the left and the pyramid structure inside the coefficient matrix is shown on the right.

scaling coefficients of the image, representing essentially the average intensity of the corresponding region in the image. Usually the coefficients in the wavelet transform of a natural image are small on the lower levels and bigger on the upper levels (Figure 3). This property is very important for the compression: the coefficients of this highly skewed distribution can be coded using fewer bits.

2.2 Connection between orientation pyramids

Each level in the coefficient matrix represents certain property of the spatial domain image in its different locations. Structures in the natural image contain almost always both big and small details. In the coefficient matrix this means that if some coefficient is small, it is most likely that also the coefficients, representing smaller features of the same spatial location, are small. This can be seen in Figure 3: different levels of the same coefficient pyramid look similar, but are in different scales. The EZW takes advantage of this by scanning the image in depth first order, i.e. it scans all the coefficients related to one spatial location in one orientation pyramid before moving to another location. This way it can code a group of small coefficients together, and thus achieves better compression performance.

In natural image, most of the features are not strictly horizontal or vertical, but contain both components. The *DLWIC takes advantage of this by binding also all three orientation pyramids together*: The scanning is done only for the horizontal orientation pyramid (B), but bits of all three coefficients, representing the three orientations of the same location and scale, are coded together. Surprisingly this only slightly enhances the compression performance. The feature is however included in the DLWIC because of its advantages: it simplifies the scanning, makes the implementation faster and reduces the size of auxiliary data structures.

Figure 3: An example of the Daubechies wavelet transform. The original 512×512 sized picture is on the left and its transform is presented with absolute values of the coefficients in logarithmic scale on the right.

2.3 Bit-level coding

The coefficient matrix of size $W \times H$ is scanned by bit-levels beginning from the highest bit-level n_{max} required for coding the biggest coefficient in the matrix (i.e. the number of the significant bits in the biggest coefficient):

$$n_{max} = \lfloor \log_2(\max\{(|c_{i,j}|)|0 \leq i < W \wedge 0 \leq j < H\}) + 1 \rfloor, \tag{1}$$

where the coefficient in (i, j) is marked with $c_{i,j}$. The coefficients are represented using positive integers and the sign bits that are stored separately. The coder first codes all the bits on the bit-level n_{max} of all the coefficients, then all the bits on bit-level $n_{max} - 1$ and so on until the least significant bit-level 1 is reached or the scanning algorithm is stopped (Section 2.7). The sign is coded together with the most significant bit (the first 1-bit) of a coefficient. For example three coefficients $c_{0,0} = -19_{10} = -10011_2, c_{1,0} = 9_{10} = 01001_2, c_{2,0} = -2_{10} = -00010_2$ would be coded as

$$\underbrace{1100}_{5}\,\underbrace{0100}_{4}\,\underbrace{000}_{3}\,\underbrace{1011}_{2}\,\underbrace{110}_{1}, \tag{2}$$

where the corresponding bit-level numbers are marked under the bits coded on that level (without signs it would be $\underbrace{100}_{5}\,\underbrace{010}_{4}\,\underbrace{000}_{3}\,\underbrace{101}_{2}\,\underbrace{110}_{1}$).

Because of the progressivity, the code stream can be truncated at any position and the decoder can approximate the coefficient matrix using received information. The easiest way of approximating the unknown bits in the coefficient matrix would be to fill them with zeroes. In the DLWIC algorithm a more accurate estimation is

used, the first unknown bit of each coefficient, for which the sign is known, is filled with one and the rest bits are filled with zeroes. For example, if the first seven bits of the bit-stream (2) have been received, the coefficients would be approximated: $c_{0,0} = -20_{10} = -10100_2, c_{1,0} = 12_{10} = 01100_2, c_{2,0} = 0_{10} = 00000_2$.

2.4 Zerotrees in DLWIC

A bit-level is scanned by first coding a bit of a scaling coefficient (on the level S in the Figure 2). Then recursively the three bits of the coefficients in the same spatial location on the next level of the orientation pyramids (A2,B2,C2) are coded. The scanning continues to the next scaling coefficient, after all the coefficients in the previous spatial location in all the pyramid levels has been scanned.

We will define that a coefficient c is *insignificant* on a bit-level n, if and only if $|c| < 2^{n-1}$. Because the coefficients on the lower pyramid levels tend to be smaller than on the higher levels and different sized details are often spatially clustered, probability for a coefficient for being insignificant is high, if the coefficient on the higher level in the same spatial location is insignificant.

If an insignificant coefficient is found in the scanning, the compression algorithm will check if any of the coefficients below the insignificant one is significant. If no significant coefficients are found, all the bits in those coefficients on current bit-level are zeroes and thus can be coded with only one bit. This structure is called *zerotree*.

One difference to the EZW algorithm is that the DLWIC scans all the orientations simultaneously and thus constructs only one shared zerotree for the all the orientation pyramids. Also the significance information is coded at the same pass as the significant bits in the coefficients, whereas the EZW and SPIHT algorithms use separate passes for the significance information.

2.5 Two dimensional significance heap

It is a slow operation to perform a significance check for all the coefficients on a specific spatial location on all the pyramid levels. The DLWIC algorithm uses a new auxiliary data-structure, which we call *two dimensional significance heap*, to eliminate the slow significance checks.

The heap is a two dimensional data-structure of the same size (number of elements) and shape as the horizontal orientation pyramid in the coefficient matrix. *Each element in the heap defines the number of bits needed to represent the largest coefficient in any orientation pyramid in the same location on the same level or below it.* Thus the scanning algorithm can find out, whether there is a zerotree starting from a particular coefficient on a certain bit-level by comparing the number of the bit-level to the corresponding value in the heap.

Here and in the rest of this paper we denote the height of the coefficient matrix with H, the width with W and the number of levels in the pyramid excluding the scaling coefficient level (S) with L. Thus the dimensions of the scaling coefficient level are: $H_s = H/2^L$ and $W_s = W/2^L$. Furthermore the dimensions of the level

in the two-dimensional heap, where (x, y) resides are

$$\begin{aligned} W_{x,y} &= W_s 2^{\lfloor \log_2(\max\{1, \frac{y}{H_s}\}) \rfloor} \\ H_{x,y} &= H_s 2^{\lfloor \log_2(\max\{1, \frac{y}{H_s}\}) \rfloor} \end{aligned} \tag{3}$$

Now the heap elements $h_{x,y}$ can be defined with the functions $h_t(x, y)$, $h_c(x, y)$ and $h_s(x, y)$:

$$\begin{aligned} h_t(x,y) &= \max\{h_{x,y+H_s}, \lfloor \log_2(|c_{x,y}|) \rfloor + 1\} \\ h_c(x,y) &= \max\{h_{2x,2y}, h_{2x+1,2y}, h_{2x,2y+1}, h_{2x+1,2y+1}\} \\ h_s(x,y) &= \lfloor \log_2(\max\{|c_{x,y}|, |c_{x+W_{x,y},y}|, |c_{x+W_{x,y},y-H_{x,y}}|\}) \rfloor + 1 \\ h_{x,y} &= \begin{cases} h_t(x,y), & \text{if } x < W_s \wedge y < H_s \\ h_s(x,y), & \text{if } x \geq W/2 \wedge y \geq H/2 \\ \max\{h_s(x,y), h_c(x,y)\}, & \text{otherwise,} \end{cases} . \end{aligned} \tag{4}$$

Note that the definitions (3) and (4) are only valid for the elements in the heap, where $0 \leq y < H$ and $0 \leq x < W_s 2^{\lfloor \log_2(\max\{1, \frac{y}{H_s}\}) \rfloor}$. While the definition of the heap looks complex, we can construct the heap with a very simple and fast algorithm (Alg. 1).

2.6 Coding algorithm

The skeleton of the compression algorithm (Alg. 2) is straightforward: 1) the spatial domain image is transformed to wavelet domain by constructing the octave band composition, 2) the two dimensional heap is constructed (Alg. 1), 3) the QM-coder is initialized, 4) the coefficient matrix is scanned in bit-levels by executing scanning algorithm (Alg. 3) for each top level coefficient on each bit-level.

The decoding algorithm is similar. First an empty two dimensional heap is created by filling it with zeroes. Then the QM-decoder is initialized and the same scanning algorithm is executed in such way that instead of calculating the decisions, it extracts the decision information from the coded data.

The scanning algorithm (Alg. 3) is the core of the compression scheme. It tries to minimize correlations between the saved bits by coding as many bits as possible with zerotrees. In the pseudo-code, Bit(x,n) returns n:th bit of the absolute value of x and $s_{i,j}$ denotes the sign of the coefficient in the matrix element (i, j). Bits are coded with function QMCode(b,CONTEXT), where b is the bit to be coded and CONTEXT is the context used as explained in Section 2.8. Context can be either a constant or some function of variables known to both the coder and decoder. In both cases, the value of the context is not important, but it should be unique for each combination of parameters. Stopping of the algorithm is queried with function ContinueCoding(), which returns true, if the coding should be continued. In order to calculate the stopping condition, the quality of the approximated resulting image must be calculated while coding. This is achieved by calling function DLUpdate(n,x) every time after coding n:th bit of the coefficient x. Both calculations are explained in the Section 2.7. The dimensions of the matrix and its levels are noted in the same way as in the Section 2.5.

The scanning algorithm first checks the stopping condition. Then we check from the two dimensional heap, whether there is a zerotree starting from this location, and code the result. If the coefficient had become significant earlier, the decoder knows also that and thus we can omit the coding of the result. If we are coding a scaling coefficient ($l = 0$), we only process that coefficient and then recursively scan the coefficient in the same location on the next level below this one. If we are coding a coefficient below the top-level, we must process all three coefficients in three orientation pyramids in this spatial location and then recursively scan all four coefficients on the next level, if that level exists.

When a coefficient is processed using ScanCoeff algorithm (Alg. 4), we first check, whether it had become significant earlier. If that is the case, we just code the bit on the current bit-level and then do the distortion calculation. If the coefficient is smaller than 2^n, we code the bit on the current bit-level, and also check whether that was the first 1-bit of the coefficient. If that is true, we also code the sign of the coefficient and do the distortion calculation.

2.7 Stopping condition and distortion limiting

The DLWIC continues coding until some of the following conditions occur: 1) All the bits of the coefficient matrix have been coded, 2) The number of bits produced by the QM-coder reach a user specified threshold, or 3) the distortion of the output image, that can be constructed from sent data, decreases below the user specified threshold. The binary stopping decisions made before coding each bit of a coefficient are coded, as the decoder must exactly know when to stop decoding.

The first condition is trivial, as the main loop (Alg. 2) ends when all the bits have been coded. The second condition is also easy to implement: output routine of the QM-coder can easily count the number of bits or bytes produced. To check the third condition, the algorithm must know the MSE of the decompressed image. The MSE of the decompressed image could be calculated by doing inverse wavelet transform for the whole coefficient matrix and then calculating the MSE from the result. Unfortunately this would be extremely slow, because the algorithm must check the stopping condition very often.

The reason for using Daubechies wavelet transform is its orthonormality. For orthonormal transforms, the square sums of the pixel values of the image before and after the transform are equal:

$$\sum_{i,j}(x_{i,j})^2 = \sum_{i,j}(c_{i,j})^2 \tag{5}$$

where $x_{i,j}$ stands for the spatial domain image intensity and $c_{i,j}$ is the wavelet coefficient. Furthermore, the mean square error between the original image and some approximation of it can be calculated equally in the wavelet and spatial domains. Thus we do not have to do the inverse wavelet transform to calculate the MSE.

Instead of tracking the MSE, we track the current square error, *cse*, of the approximated image because it is computationally easier. The initial approximation

of the image is zero coefficient matrix, as we have to approximate the coefficients to be zero, when we do not know their signs. Thus the initial *cse* equals to the energy of the coefficient matrix.

$$cse \leftarrow \sum_{i,j} (c_{i,j})^2 \tag{6}$$

After sending each bit of a coefficient c, we must update *cse* by subtracting the error produced by the previous approximation of c and adding the error of its new approximation. The error of an approximation of c depends only on the level of the last known bit and the coefficient c itself. If we code the n:th bit of c, then the *cse* should be updated:

$$cse \leftarrow cse - \begin{cases} \left[(|c|\mathrm{AND}_2(2^n - 1)) - 2^{n-1}\right]^2 - & \\ \left[(|c|\mathrm{AND}_2(2^{n-1} - 1)) - 2^{n-2}\right]^2 & \text{if} \lfloor \log_2 c \rfloor > n-1 \\ c^2 - (2^{(n-1)} + 2^{(n-2)} - c)^2 & \text{if} \lfloor \log_2 c \rfloor = n-1 \\ 0 & \text{if} \lfloor \log_2 c \rfloor < n-1, \end{cases} \tag{7}$$

where AND_2 is bitwise and-operation. The first case defines the error reduced by finding out one bit of a coefficient, when the sign is already known. The second case defines the error reduced by finding out the sign of a coefficient and the last case states that *cse* does not change if only zero bit before the coefficients first one bit is found. The equation 7 holds only, when $n > 1$.

2.8 The use of contexts in QM-coder

The QM-coder is a binary arithmetic coding algorithm that tries to code binary data following some probability distribution as efficiently as possible. Theoretically an arithmetic coder compresses data according to its entropy [4], but the QM-coder uses a dynamical probability estimation technique [6, 7, 2] based on state automata, and its compression performance can even exceed the entropy, if the local probability distribution differs from the global distribution used in the entropy calculation.

The DLWIC codes different types of information with differing probability distributions. For example the signs of coefficients are highly random, that is the probability of plus sign is approximately 0.5, but the probability of finding the stopping condition is only $1/N$, where N is the number of stopping condition evaluations. If bits following the both distributions would be coded using the same probability distribution, the compression performance obviously would not be acceptable.

To achieve better compression performance the DLWIC uses separate *contexts* for binary data following different probability distributions. The contexts for coding the following type of data are defined: 1) signs, 2) stopping conditions, 3) the bits of the coefficients after the first one bit, 4) the bits of the scaling coefficients, 5) zerotrees on the different levels of the pyramid, 6) the significance check of the

Figure 4: Test images from top left: 1) barb (512×512), 2) bird (256×256), 3) boat (512×512), 4) bridge (256×256), 5) camera (256×256), 6) circles (256×256), 7) crosses (256×256), 8) france (672×496) and 9) frog (621×498).

insignificant coefficients on different pyramid levels on different orientation pyramids. The number of separate contexts is $4 * (l + 1)$, where l defines the number of levels in the pyramids. It would also be possible to define different contexts for each bit-level, but dynamical probability estimation in the QM-coder seems to be so efficient that this is not necessary.

3 Test results

The performance of the DLWIC algorithm is compared to the JPEG and the vqSPIHT [3] algorithms with a set (Fig. 4) of 8 bit grayscale test images. The vqSPIHT is an efficient implementation of the SPIHT [8] compression algorithm. The vqSPIHT algorithm uses the biorthogonal B97 wavelet transform [10], the QM-coder and a more complicated image scanning algorithm than the DLWIC. Image quality is measured in terms of *peak signal to noise ratio* [1] (PSNR), which is an inverse logarithmic measure calculated from MSE.

3.1 Compression efficiency

To compare the compression performance of the algorithms, the test image set is compressed with different BPP-rates from 0.1 to 3.0 and the PSNR is calculated as the mean for all the images. Because it is not possible to specify BPP as a parameter for JPEG compression algorithm, various quantization parameters are used and the BPP value is calculated as a mean value of the image set for each quantization value.

As can be seen in the Figure 5, the performance of the vqSPIHT and the DLWIC algorithms is highly similar. This is somewhat surprising, because of the greater complexity and better wavelet transform used in the vqSPIHT. The quality of the images compressed with the EZW variants seem to exceed the quality produced by

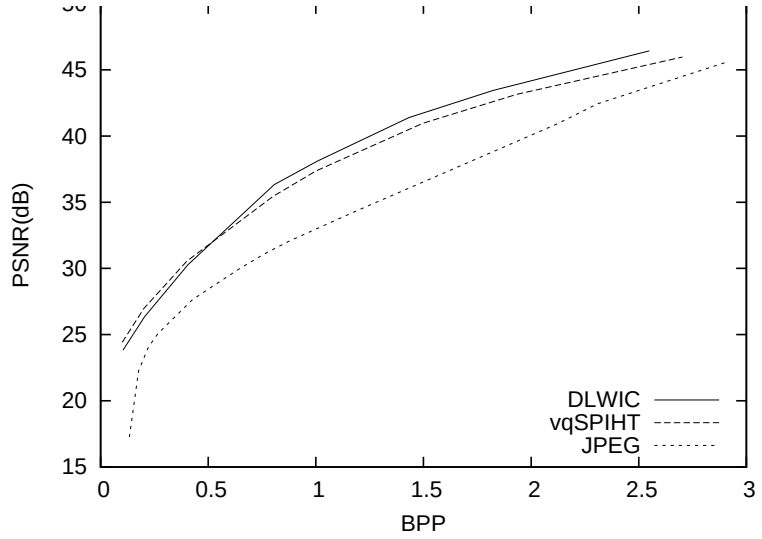


Figure 5: Compression performance comparison of DLWIC, vqSPIHT and JPEG. The PSNR-values correspond to mean value obtained from test image set (Fig. 4).

the JPEG. This is especially true when low bit-rates are used. Poor scalability of the JPEG to low BPP values is probably implied by the fixed block size used in the DCT transform of the JPEG as opposed to multi-resolution approach of the wavelet based methods.

One might expect that a conventional image compression algorithm such as the JPEG would give similar PSNR and BPP values for similar images when fixed quantization parameter is used. This is not the case as demonstrated in the Figure 6, where all the test images are compressed using the same quantization parameter (20) with the standard JPEG.

3.2 Speed and memory usage

The speed of the implementation is not compared to other techniques, because the implementation of the algorithm is not highly optimized. Instead an example of the time consumption of the different components of the compression process is examined using the GNU profiler. The frog test image is compressed using a 400MHz Intel Pentium II workstation running Linux and the algorithm is implemented in C language and compiled with GNU C 2.7.2.1 using "-O4 -p" options. The cumulative CPU time used in the different parts of the algorithm is shown in the Figure 7.

When the image is compressed with a low BPP-rate, most of the time is consumed by the wavelet transform. When the BPP-rate increases, the time used by the QM-coder, the scanning algorithm and the distortion calculations increases in

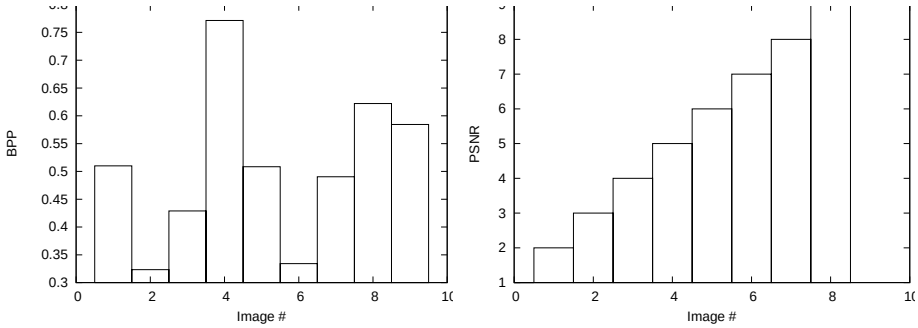


Figure 6: All the images of the test image set are compressed by JPEG with the same quantization value (20), and the BPP (left) and the PSNR (right) of the resulting images are shown.

somewhat linear manner. Construction of the two dimensional heap seems to be quite fast operation and distortion limiting is not very time consuming.

If we want to optimize the implementation of the DLWIC, the biggest problem would probably be the extensive use of the QM-coder, that is already highly optimized. One way to alleviate the problem would be to store the stopping condition is some other way than compressing the binary decision after each bit received. Also the transform would have to be optimized to achieve faster compression, because it consumes nearly half of the processing time, when higher compression ratios are used.

Probably the biggest advantage of the DLWIC over the SPIHT and even the vqSPIHT is its low auxiliary memory usage. The only auxiliary data-structure used, the two dimensional heap, can be represented using 8-bit integers and thus only consumes approximately $8 * N/3$ bits of memory, where N is the number of coefficients. If the coefficients are stored with 32-bit integers, this implies 8% auxiliary memory overhead, which is very reasonable, when compared to 32% overhead in the vqSPIHT or even much higher overhead in the SPIHT algorithm, which depends on the target BPP-rate.

4 Summary and conclusion

In this paper a new general purpose wavelet image compression scheme, DLWIC, was introduced. Also it was shown how the distortion of the resulting decompressed image can be calculated while compressing the image and thus how the distortion of the compressed image can be limited. The scanning algorithm in the DLWIC is very simple and it was shown, how it can be efficiently implemented using a two dimensional heap structure.

Compression performance of the DLWIC was tested with a set of images and the compression performance seems to be promising, when compared to a more complex

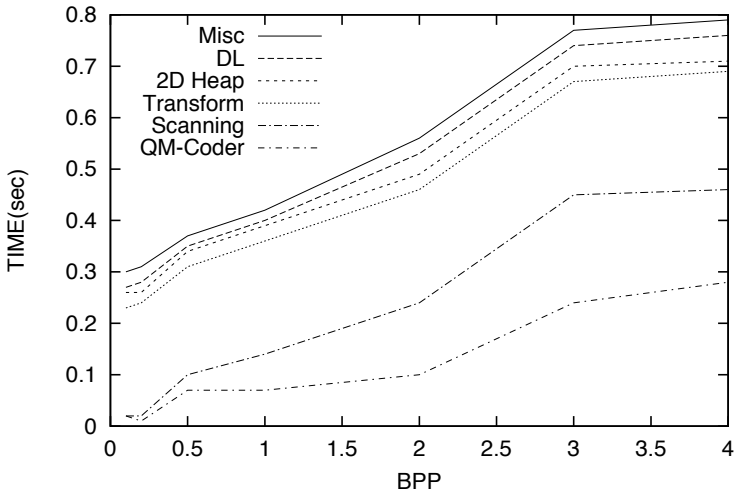


Figure 7: Running time of the different components in the DLWIC compression/decompression algorithm, when compressing the frog test image (Fig. 4). Graph shows the cumulative CPU time consumption when different BPP-rates are used.

compression algorithm, the vqSPIHT. Furthermore, the compression performance easily exceeds the performance of the JPEG, especially when high compression ratios are used.

Further research for extending the DLWIC algorithm to be used in lossless or nearly lossless multidimensional medical image compression is planned. Also the implementation of the DLWIC will be optimized and usage of some other wavelet transforms will be considered.

References

[1] R. Gonzalez and R. Woods. *Digital Image Processing*. Addison-Wesley Publishing Company, 1992.

[2] ITU-T. Progressive bi-level image compression, recommendation t.82. Technical report, International telecommunication union, 1993.

[3] A. Järvi, J. Lehtinen, and O. Nevalainen. Variable quality image compression system based on SPIHT. *to appear in Signal Processing: Image Communications*, 1998.

[4] Sayhood K. *Introduction to Data Compression*. Morgan Kaufmann, 1996.

[5] Juha Kivijrvi, Tiina Ojala, Timo Kaukoranta, Attila Kuba, László Nyúl, and Olli Nevalainen. The comparison of lossless compression methods in the case of a medical image database. Technical Report 171, Turku Centre for Computer Science, April 1998.

[6] W.B. Pennebaker and J.L. Mitchell. Probability estimation for the q-coder. *IBM Journal of Research and Development*, 32(6):737–752, 1988.

[7] William Pennebaker and Joan Mitchell. *Jpeg : Still Image Data Compression Standard*. Van Nostrand Reinhold, 1992.

[8] Amir Said and William A. Pearlman. A new fast and efficient image codec based on set partitioning in hierarchical trees. *IEEE Transactions on Circuits and Systems for Video Technology*, 6:243–250, June 1996.

[9] J.M. Shapiro. Embedded image coding using zerotrees of wavelet coefficients. *IEEE Trans. Signal Processing*, 31(12), December 1993.

[10] M. Vettereli and J. Kovačević. *Wavelets and Subband Coding*. Prentice Hall, Englewood Cliffs, NJ, 1995.

Algorithm 1 Construct2DHeap

for $l \leftarrow 1$ **to** $L+1$ **do**
  $H_t \leftarrow H/2^{\min\{l,L\}}$, $W_t \leftarrow W/2^{\min\{l,L\}}$
  for $j \leftarrow 0$ **to** $H_t - 1$ **do**
    for $i \leftarrow 0$ **to** $W_t - 1$ **do**
      if $\mathrm{l} = 1$ **then**
        $t \leftarrow 0$
        $u \leftarrow \lfloor 1 + \log_2(\max\{|c_{i,j+H}|, |c_{i+W,j}|, |c_{i+W,j+H}|\}) \rfloor$
      else if $l \leq L$ **then**
        $t \leftarrow \max\{h_{2x,2y}, h_{2x+1,2y}, h_{2x,2y+1}, h_{2x+1,2y+1}\}$
        $u \leftarrow \lfloor 1 + \log_2(\max\{|c_{i,j+H}|, |c_{i+W,j}|, |c_{i+W,j+H}|\}) \rfloor$
      else
        $t \leftarrow \max\{h_{i,j+H_s}, h_{i+W_s,j}, h_{i+W_s,j+H_s}\}$
        $u \leftarrow \lfloor 1 + \log_2(\max\{(|c_{i,j}|) | 0 \leq i < W \wedge 0 \leq j < H\}) \rfloor$
      $h_{i,j} \leftarrow \max\{t, u\}$

Algorithm 2 CompressDLWIC

Transform the spatial image with Daubechies wavelet transform constructing the octave band composition where the coefficients $c_{i,j}$ are represented with positive integers and separate sign bit.
Construct the two dimensional heap (Alg. 1).
Initialize QM-coder
Calculate initial distortion of the image (Section 2.7).
$n_{max} \leftarrow \max\{h_{i,j} | 0 \leq i < W_s \wedge 0 \leq j < H_s\}$
for $n \leftarrow n_{max}$ **to** 1 **do**
  for $j \leftarrow 0$ **to** $H_s - 1$ **do**
    for $i \leftarrow 0$ **to** $W_s - 1$ **do**
      Scan$(i, j, 0, n)$ (Alg. 3)

Algorithm 3 Scan(i, j, l, n)

if ContinueCoding() **then**
  if $h_{i,j} < n$ **then**
    QMCode(INSIGNIFICANT, SIGNIFICANCE-TEST(l))
  else
    if $h_{i,j} = n$ **then**
      QMCode(SIGNIFICANT, SIGNIFICANCE-TEST(l))
    if $l = 0$ **then**
      ScanCoeff$(i, j$,TOPLEVEL$, n)$
      Scan$(i, j + H_s, 1, n)$
    else
      ScanCoeff$(i, j$,HORIZONTAL$\left(\begin{array}{l} l, c_{(i+W_{i,j}),j} < 2^n, \\ c_{(i+W_{i,j}),(j-Hi,j)} < 2^n \end{array} \right), n)$
      ScanCoeff$(i + W_{i,j}, j$,DIAGONAL$\left(\begin{array}{l} l, c_{i,j} < 2^{n-1}, \\ c_{(i+W_{i,j}),(j-Hi,j)} < 2^n \end{array} \right), n)$
      ScanCoeff$(i + W_{i,j}, j - H_{i,j}$,VERTICAL$\left(\begin{array}{l} l, c_{i,j} < 2^{n-1}, \\ c_{(i+W_{i,j}),j} < 2^{n-1} \end{array} \right), n)$
      if $2 * y < H$ **then**
        Scan$(2 * i, 2 * j, l + 1, n)$
        Scan$(2 * i + 1, 2 * j, l + 1, n)$
        Scan$(2 * i, 2 * j + 1, l + 1, n)$
        Scan$(2 * i + 1, 2 * j + 1q, l + 1, n)$

Algorithm 4 ScanCoeff$(x, y$,CONTEXT$, n)$

if $c_{x,y} < 2^n$ **then**
  QMCode(Bit$(c_{x,y}, n)$,CONTEXT)
  if Bit$(c_{x,y}, n)$= 1 **then**
    QMCode($s_{x,y}$,SIGN)
    DLUpdate$(n, c_{x,y})$
else
  QMCode(Bit$(c_{x,y}, n)$,COEFFICIENTBIT)
  DLUpdate$(n, c_{x,y})$

Predictive depth coding of wavelet transformed images

Joonas Lehtinen

Published in Proceedings of SPIE: Wavelet Applications in Signal and Image Processing, vol. 3813, no. 102, Denver, USA, 1999.

Predictive depth coding of wavelet transformed images

J. Lehtinen[a]

[a]Turku Centre for Computer Science and
Department of Mathematical Sciences, University of Turku,
Lemminkäisenkatu 14 A, 20520 Turku, Finland

ABSTRACT

In this paper, a new prediction based method, *predictive depth coding* (PDC), for lossy wavelet image compression is presented. It compresses a wavelet pyramid composition by predicting the number of significant bits in each wavelet coefficient quantized by the universal scalar quantization and then by coding the prediction error with arithmetic coding. The adaptively found linear prediction context covers spatial neighbors of the coefficient to be predicted and the corresponding coefficients on lower scale and in the different orientation pyramids. In addition to the number of significant bits, the sign and the bits of non-zero coefficients are coded. The compression method is tested with a standard set of images and the results are compared with SFQ, SPIHT, EZW and context based algorithms. Even though the algorithm is very simple and it does not require any extra memory, the compression results are relatively good.

Keywords: wavelet, image, compression, lossy, predictive, depth, coding

1. INTRODUCTION

While the DCT-based JPEG[1] is currently the most widely used lossy image compression method, much of the lossy still image compression research has moved towards wavelet-transform[2] -based compression algorithms. The benefits of wavelet-based techniques are well-known and many different compression algorithms , which have good compression performance, has been documented in literature.[3–6]

A general system for lossy wavelet image compression usually consists of three steps: 1) signal decomposition, 2) quantization and 3) lossless coding. Signal decomposition separates different frequency-components from the original spatial domain image by filtering the image with a subband filter-bank. Quantization is the phase, where some of the image information is purposely lost to achieve better compression performance. In the third phase, the quantized wavelet coefficients are coded using as few bits as possible. Usually this is done in two phases: modelling the information to symbols and then entropy encoding the symbols. The entropy coding can be optimally implemented with arithmetic coding,[7] and thus the compression performance largely depends on the quantization and modelling algorithm.

Signal composition can be done in different ways using 1-dimensional wavelet transform for the columns and the rows of the image in some order or by directly using 2-dimensional transform.[8] One of the most popular compositions is to use 1-dimensional transforms for the rows and columns of the image and thus divide the image to four different orientation bands. Then the same signal-composition is done recursively to the upper left quadrant of the image to construct a pyramid (dyadic) composition (Fig.1). It is also possible to apply the signal-composition to other orientation subbands and this way to construct more complicated wavelet packet signal-compositions.[2]

The 1-dimensional discrete wavelet transform (DWT) is basically very simple: discrete signal is filtered using low- and high-pass filters, and the results are scaled down and concatenated. The filters are constructed from a mother wavelet-function by scaling and translating the function. Some of the most used functions in lossy image compression are biorthogonal 9-7 wavelet, Daubechies wavelet family and coiflets. Usually the transforms itself are not lossy, but produce floating point result, which is often rounded to integers for coding.

The quantization of the wavelet coefficient matrix can be done in many ways. The most obvious way of quantizing the matrix is to use uniform scalar quantizer for the whole matrix, which usually gives surprisingly good results. It is also possible to use different quantizers for the different subbands or even regions of the image, which leads to variable quality image compression.[9] A more sophisticated way of quantifying the matrix would be to use vector

Further author information: E-mail: jole@jole.fi, WWW: http://jole.fi/

quantization, that is computationally more intensive, but in some cases gives better results.[10] Many techniques also implement progressive compression by integrating the modelling and the scalar quantization.[3]

Many different approaches for modelling has been documented in literature. The most obvious one is to entropy code the quantized coefficients directly using some static, adaptively found or explicitly transmitted probability distribution. Unfortunately this leads to relatively poor compression performance, as it does not use any dependencies between the coefficients. Shapiros landmark paper[4] used quad-trees to code zero regions (called zerotrees) of the scalar-quantized coefficient matrix. A. Said and W. Pearlman developed the zerotree approach further by implicitly sending zerotrees embedded into code-stream and achieved very good compression performance with their SPIHT-algorithm.[3] Also spatial contexts has successfully been used to select the probability distribution for coding the coefficients.[5] One of the best methods is based on space-frequency quantization,[6] that tries to find optimal balance between scalar quantization and spatial quantization achieved with zerotree-like structures.

In lossless image compression, signal prediction and prediction-error coding are efficient and widely used methods. In this paper the usage of these methods in lossy wavelet image coding is evaluated and *a new method for modelling quantized pyramid composition is introduced*. The modelling algorithm is evaluated by implementing a wavelet image compression scheme with the following components: 1) biorthogonal 9-7 transform is used to construct a pyramid composition, 2) uniform scalar quantization is used for the quantization of the coefficients, 3) an adaptive prediction model is used to estimate the coefficient values and 4) the prediction errors are coded with an adaptive arithmetic coder.

It is possible to try directly predict the values of the wavelet coefficients, but it is probable that the prediction results would then be very inaccurate. Because the goal is to code the quantized coefficients accurately and thus the prediction error would be quite big, the efficient error coding would be impossible. Instead of trying to predict the exact coefficients, the PDC-algorithm estimates the number of bits needed to represent the coefficients and then codes the coefficients separately.

In this paper the number of significant bits in an coefficient is called *depth*. Each coefficient is coded as a triplet of the depth-prediction error, the sign and the actual bits. The prediction is done with a simple linear predictor, which covers six spatial neighbors, a coefficient on the lower scale band and two coefficients on the different orientation bands. The weights of the predictor are adaptively trained in the course of the compression. The signs are coded using simple context model. The coding of the depths predicted to be near zero utilizes several context.

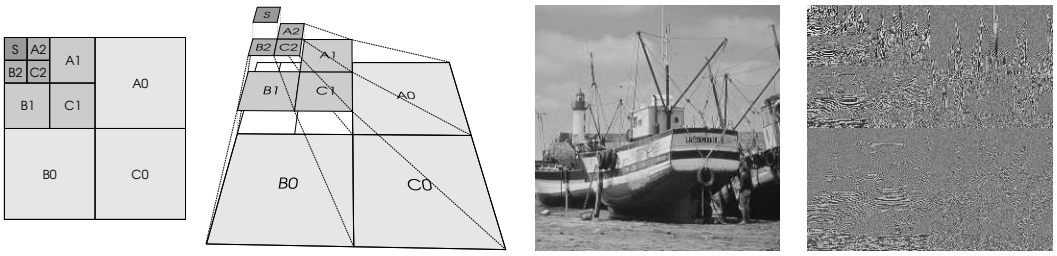
In section 2 we give a brief introduction to all components of the PDC compression scheme. The original signal is processed by the biorthogonal 9-7 wavelet transform. As a result we get a pyramid composition of the image. We analyze the dependencies between the coefficients and use finally scalar quantization and entropy coding to the coefficients. Section 3 explains the modelling algorithm in detail. First we define the concept of coefficient depth and a depth prediction context. Then we give an adaptive method for the coefficient estimation and discuss the modelling of the estimation errors. Finally a spatial sign coding context is defined and the coding of absolute values of the actual coefficients are explained. The compression performance of PDC is evaluated experimentally and compared to some existing image compressing techniques in section 4. Conclusions are shown in section 5.

2. BACKGROUND

2.1. The structure of the compression method

The PDC-compression algorithm consists of six steps:

1. The spatial domain image is transformed to a pyramid composition structure (section 2.2).
2. The coefficient matrix of step 1 is quantized using scalar quantization (section 2.3)
3. Quantized coefficients are converted to depth representation and average values of the depth are calculated for the different subbands (section 3.2)
4. The significant bits of the absolute coefficient values (section 3.5) and the signs (section 3.4) are entropy-coded (section 2.4) using several different coding contexts.
5. The depth of each coefficient is predicted using the same information as will be available at the moment of decompression in the same stage when predicting the same coefficient (section 3.2)



1 2 3

Figure 1. Images from left to right: 1) pyramid composition structure and the three orientation pyramids A,B and C, 2) spatial domain test image and 3) the same image in wavelet domain: the absolute values of the coefficients are represented in logarithmic scale.

6. The prediction errors are calculated and entropy-coded (section 3.3)

The decompression is done similarly, but in the reverse order: 1) the depth-prediction errors are decoded, 2) depths are predicted and the correct depth values are calculated using the decoded prediction errors, 3) the signs and the absolute values of the coefficients are decoded 4) different components of the coefficients are combined and 5) the reverse transform is applied to produce the decompressed spatial domain image.

2.2. Pyramid composition structure

In the PDC, the image is transformed to a subband pyramid-composition using biorthogonal 9-7^2 wavelet functions. The transform is done hierarchically by filtering the rows with low- and high-pass filters and then combining the down-scaled results side by side. Then the same operation is done to the columns of the image. The result of these two operations is a matrix with the same dimensions as the original image, but it is divided into four parts: S) Top left part contains a scaled down low-frequency components of the image. This part is also called scaling coefficients and basically it looks like a scaled down version of the image. A) Top right part contains the vertical high-frequency details of the image. B) Bottom left corner contains the horizontal details. C) Bottom right corner of the image contains the diagonal high-frequency components.

Pyramid composition (Fig. 1) is created by doing the transformation described above recursively to the scaling coefficients on the top left corner of the coefficient matrix. This can be repeated while the dimensions of the scaling coefficients are even. The three pyramids can be seen in the pyramid composition: A,B,C. Pyramid levels correspond to different scale details of the spatial domain image and the different pyramids correspond to details having different orientations. Usually the details in spatial image have features of multiple orientations and scales and thus they affect to the coefficients on all pyramids on multiple levels. These dependencies imply that coefficients are not independent and their values or magnitudes can be tried to predict from the other coefficients.

2.3. Scalar quantization

Scalar quantization of the coefficients is defined by a mapping from the set of all possible coefficients to a smaller set of representative coefficients, which approximates the original coefficients. Uniform scalar quantization is a simplification of general case, where each quantized coefficient represents a uniform set of original coefficients. Thus the uniform scalar quantization Q of the coefficients c can be simply implemented as

$$Q(c, q) = \lfloor q(c + 0.5) \rfloor / q,$$

where the quantization parameter q determines the accuracy of the mapping.

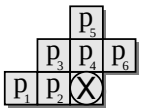


Figure 2. Spatial context of the predicted coefficient (marked with X) includes the depths of the six surrounding coefficients.

2.4. Entropy coding

Entropy H gives a lower bound[7] on the average code length of a symbol having probability p :

$$H(p) = p \log_2(1/p).$$

This implies that if our modelling system can model the image to a stream of N symbols having independent probabilities, $p_0, p_1, \ldots, p_{N-1}$, an optimal entropy coder can compress it to $\sum_{i=0}^{N-1} p_i \log_2(1/p_i)$ bits.

The assumption here is that the probabilities of the symbols are known. It is of course possible first to do all the modelling and calculate the probabilities for the symbols, but usually adaptive probability estimation gives very good results. A benefit of the adaptive approach is that the estimate for the probability distribution can be calculated from the symbols sent earlier.

An arithmetic coder[7] represents a stream of symbols as a number from the range $[0, 1)$. This is done by recursively dividing the range in the proportion to the symbol probabilities and selecting a subrange corresponding to the symbol to be coded. The output of the algorithm is the binary representation of some number in the final range. The compression performance is very close to the optimum defined by the entropy.

3. PREDICTIVE DEPTH CODING

3.1. PDC algorithm

In the PDC-algorithm, the coefficients are compressed one by one starting from the highest pyramid level and continuing downwards levelwise in all orientation pyramids at the same time. More precisely the scaling coefficients (on the level S in Fig. 1) are compressed first in row major order, then similarly the bands on the next pyramid level in the order C,B,A (that is C2,B2,A2 in the example) and finally similarly all the other levels below it.

For each coefficient, we predict it's depth and calculate the prediction error as the difference of the actual and predicted values. The prediction error is then coded together with the sign and the actual absolute value of the coefficient. In addition to an individual coefficient, the compression algorithm considers coefficients, which are on the spatial neighbourhood of the coeffient, on different orientation pyramids on the same spatial location and on the parent level. Also the information on the mean depth of coefficients on current level and its parent level is utilized in the prediction.

3.2. Depth prediction

The depth D of the coefficient c is defined as the number of bits needed for expressing its absolute value:

$$D(c) = \begin{cases} \lfloor \log_2(|c|) + 1 \rfloor, & \text{if } c > 0 \\ 0, & \text{if } c = 0 \end{cases}$$

Thus the coefficient can be represented as the depth, followed by the absolute value and the sign. This representation avoids the insignificant bits.

In PDC, the prediction is made with linear predictor $P(p_1, p_2, \ldots, p_N, w_1, w_2, \ldots, w_N) = \sum_{i=1}^{N} w_i p_i$, where p_i forms the prediction context, which is weighted by corresponding w_i ($sum_{i=1}^{N} w_i = 1$). The prediction context is divided into three parts: 1) spatial context describing the depths of six surrounding coefficients, 2) orientational context consisting of the depths of two orientational coefficients on the same level and at same spatial location, 3) a depth estimate calculated from the depth of the parent coefficient and the mean depths of the levels.

This gives us a prediction context of 9 coefficients. The six spatial neighbours (Fig. 2) are selected so that they give the best results with the PDC-algorithm. Experiments with different combinations of the orientational context advocated us to use a context of two coefficients at the same spatial location on the different orientation pyramids in the context. One coefficient is used on the upper level in the same spatial location and in the same orientation pyramid as the coefficient to be predicted. Because of the mean-depths of the levels are very different, the depth of the parent coefficient is divided by the mean-depth of its level and multiplied by the mean-depth of the current level.

In the compression phase, we can use only the depths known to the decompressor and for the rest of the prediction context we apply zero weights. For example for the first coefficient of a band the whole spatial context is undefined and only the coefficients on the orientational context and a coefficient on the previous level, can be used (if they are known).

The weights W_i for the linear prediction-context are calculated adaptively in the prediction algorithm:

- $\forall i \in [1,9] : W_i \leftarrow 1$
- while not finished
  - $\forall i \in [1,9] : w_i \leftarrow W_i$
  - $\forall i \in \{\text{unknown,undefined}\} : w_i \leftarrow 0$
  - $s \leftarrow \sum_{i=1}^{9} w_i$
  - $P \leftarrow \begin{cases} (\sum_{i=1}^{9} w_i D_i)/s & , \text{if } s > 0 \\ 0 & , \text{if } s = 0 \end{cases}$
  - $\forall i \in [1,n] : W_i \leftarrow \begin{cases} W_i * \alpha & , \text{if } |p_i - D| \leq |P - D| \\ W_i/\alpha & , \text{if } |p_i - D| > |P - D| \end{cases}$
  - $\forall i \in [1,n] : W_i \leftarrow \begin{cases} \beta & , \text{if } W_i > \beta \\ \gamma & , \text{if } W_i < \gamma \\ W_i & \text{otherwise} \end{cases}$

One good setting for the parameters is $\alpha = 1.1, \beta = 100$ and $\gamma = 0.3$. The symbol D denotes the depth of the current coefficient and D_i the depth of the neighbour i. The resulting prediction is denoted with P. For the first prediction, s is 0 and thus the first coefficient should not be predicted at all.

3.3. Coding the prediction errors

The linear prediction produces a floating-point value which must be rounded before the error calculation. The prediction error can then be directly coded with arithmetic coder.

On the lower pyramid levels most of the quantized coefficient values are zero and thus also the zero prediction is very common. We can increase the performance of the compression scheme for small predictions by adjusting the probability distribution of the error coding. In the PDC this is implemented by using six independent probability distributions. The distribution is selected by classifying the predictions to the classes: $[0, 0.01)$, $[0.01, 0.05)$, $[0.05, 0.13)$, $[0.13, 0.25)$, $[0.25, 0.5)$ and $[0.5, \infty)$. This selection of coding contexts seem to work quite well for natural images, but the compression performance is rather insensitive to the number of classes and their boundaries.

3.4. Coding the sign

Although the signs of the coefficients seem to be random and their distribution is almost even, some benefit can be gained by using two neighbours (north and west) as a coding context. Thus nine different probability distributions is used, as each sign can be either $+$, $-$ or 0 (if the quantized coefficient is zero, the sign is unknown and insignificant).

3.5. Coding the quantized coefficients

In addition to the sign and the depth of the coefficient, the actual quantized absolute value must be coded. The value could be coded simply with arithmetic coding using the depth as a context, but in the PDC we take an even simpler approach: the bits of the absolute value are coded as such one by one. No that we could as well apply the binary arithmetic coder can be used, but its benefit is marginal.

As the depth of the coefficient is known, only the bits after the first one-bit of the absolute value must be coded. Even thought the distribution of these bits is almost even, the distribution of absolute values of the coefficients is skew. For this reason the distribution of the bits right after the first one-bits is also somewhat skew and thus entropy coding is used to code these bits.

4. TESTING

4.1. Compression performance

As the purpose of the PDC is to apply the prediction coding principles in wavelet-coding, the absolute compression performance results are compared to some advanced wavelet compression techniques. Furthermore, as the compression scheme consists of many different sub-components, their influence on compression performance is evaluated separately. The peak signal to noise ratio (PSNR)[11] is used as image quality measure.

The compression performance (Table 4.1) of the PDC is measured by compressing three 512×512, 8-bit test images: lena, barbara and goldhill, using different number of bits per pixel (bpp). The performance is compared to the compression results of the standard JPEG,[1] a context based wavelet compression technique (C/B),[5] space frequency quantization (SFQ)[6] and a wavelet image compression technique based on set partitioning in hierarchical trees (SPIHT).[3]

The JPEG codes the image in fixed 8×8 blocks by first transforming the image blocks using discrete cosine transform (DCT) and then applying scalar quantization to the coefficients. The quantized coefficients are coded in zig-zag order using Huffman-coding. Context based technique compresses the scalar quantized coefficients of the wavelet pyramid composition using independent probability distributions that depend on the spatial context. The technique is relatively simple, yet very efficient. SFQ is based on the usage of zerotrees[4] in wavelet pyramid composition coding. The idea of the SFQ is to find optimal balance between spatial quantization done with zerotrees and scalar quantization. The resulting compression performance is very good. The SPIHT is also based on the idea of zerotrees, but it develops the zerotree structure coding further by coding the tree information implicitly as the decision information of the set partitioning algorithm, which constructs the zerotree. Although the method is somewhat complicated, the results are very good and moreover the modelling algorithm is so powerful, that the entropy-coding can be omitted at least in some applications. The algorithm can also be easily optimized with minor modifications.[9]

The compression performance graphs (Fig. 3) show that the PDC algorithm is not as good as the current state of the art algorithms, but the differences in the compression performance are small. The fixed sized block structure of the JPEG restricts the scalability to the smaller bit-rates and thus the compression performance is poor when comparing it to wavelet-transform based algorithms.

BPP	PSNR(dB)		
	lena	barbara	goldhill
0.50	49.8	49.8	49.8
0.25	45.8	45.8	45.8
0.15	42.3	42.5	42.1
0.10	39.9	40.0	39.2
0.05	36.6	35.8	34.9
0.01	29.5	26.4	27.6

Table 1. The PDC compression results

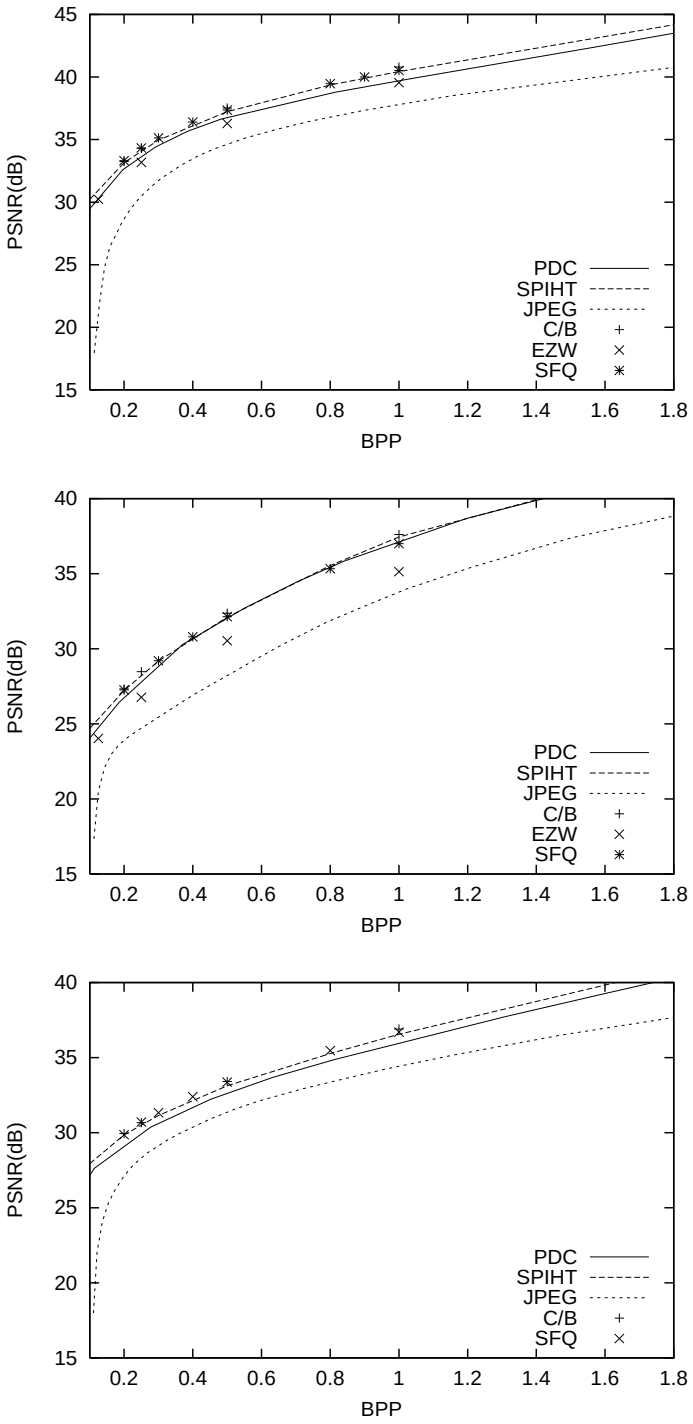


Figure 3. Compression performance of the PDC with three different 512×512 8-bit test images is measured. The test images used are from top to bottom: lena, barbara and goldhill.

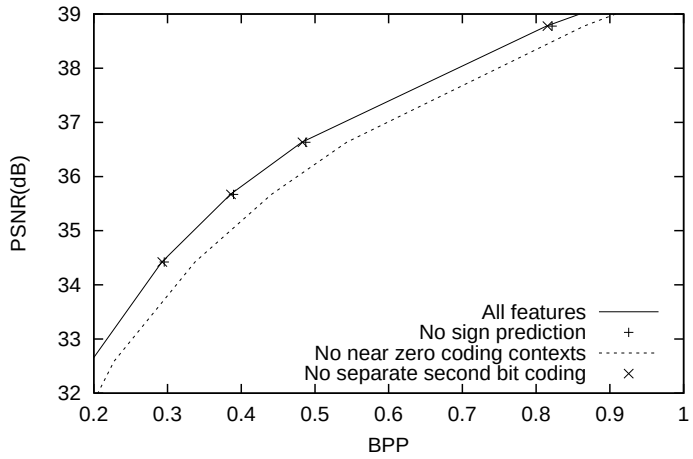


Figure 4. The compression performance of the PDC on the lena test-image, when leaving out some of the features.

4.2. The effect of different components

The modeller used in the PDC uses several different components to gain better compression performance. Many of these components can be left out from the PDC or integrated to other wavelet image compression systems. The coding gain can be measured by leaving the corresponding coding component out from the PDC and comparing the compression performance the results (Fig. 4) with the complete algorithm.

It turns out that the sign prediction and the use of a separate context for the second bit coding are not all necessary. If one would like to reduce the amount of arithmetic coding done, the performance loss from directly saving the sign bits and the coefficient bits is not very big. On the other hand it seem to be important to use several contexts for coding the coefficients that are predicted to be near zero.

4.3. Efficiency

One of the advantages of the PDC is that the algorithm does not require any extra memory, unlike many other wavelet transform coding algorithms. Still the wavelet transformed image itself must reside in memory at the prediction step. New methods for wavelet transforms having reduced memory requirements have been documented in literature,[5] and the PDC algorithm could probably be easily changed to work with them, as only the prediction context must be changed.

The speed of the PDC depends of three different steps: 1) the wavelet transform, 2) modelling and 3) arithmetic coding. The two most time consuming parts of the modelling step are coefficient prediction and prediction context weight adjustment. In the prediction one must calculate each estimate as a linear combination of the prediction context. This requires 9 floating point multiplications, additions and several memory address calculations and reads. The learning phase is about three to five times more complex than the prediction. One way to speed up the learning phase could be the use of approximate updates. In this approach we would update the weights of the prediction context only once for every n predictions, where the parameter n would determine the speed of learning.

5. CONCLUSIONS

A new method for wavelet transform coding has been proposed. The predictive depth coding technique demonstrates that the conventional prediction coding principles, originally used in lossless image-coding, can be applied to lossy wavelet transform coding. We have found that the number of significant bits in the wavelet coefficients can be predicted with small linear prediction context. A simple method for learning the weights of the linear prediction context is introduced. The PDC also demonstrates that sign compression with simple two-sign context is possible, yet relatively inefficient.

A significant benefit of the new compression technique is that it is very simple and does not need any extra memory. As the prediction context is very small, the applicability of the technique for use with reduced memory wavelet transfers should be researched. Also the possibilities of integrating the prediction coding techniques into some other wavelet coefficient modeling technique for construction of a hybrid-method should be inspected.

REFERENCES

1. W. Pennebaker and J. Mitchell, *Jpeg : Still Image Data Compression Standard*, Van Nostrand Reinhold, 1992.
2. M. Vettereli and J. Kovačević, *Wavelets and Subband Coding*, Prentice Hall, Englewood Cliffs, NJ, 1995.
3. A. Said and W. A. Pearlman, "A new fast and efficient image codec based on set partitioning in hierarchical trees," *IEEE Transactions on Circuits and Systems for Video Technology* **6**, pp. 243–250, June 1996.
4. J. Shapiro, "Embedded image coding using zerotrees of wavelet coefficients," *IEEE Trans. Signal Processing* **31**, December 1993.
5. C. Chrysafis and A. Ortega, "Efficient context-based entropy coding for lossy wavelet image compression," in *DCC, Data Compression Conference*, (Snowbird, UT), March 25 - 27 1997.
6. Z. Xiong, K. Ramchandran, and M. T. Orchard, "Space-frequency quantization for wavelet image coding," *IEEE Trans. Image Processing* , 1997.
7. S. K., *Introduction to Data Compression*, Morgan Kaufmann, 1996.
8. M. Kopp, "Lossless wavelet based image compression with adaptive 2d decomposition," in *Proceedings of the Fourth International Conference in Central Europe on Computer Graphics and Visualization 96 (WSCG96)*, pp. 141–149, (Plzen, Czech Republic), February 1996.
9. A. Järvi, J. Lehtinen, and O. Nevalainen, "Variable quality image compression system based on SPIHT," *to appear in Signal Processing: Image Communications* , 1999.
10. A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression*, Kluwer Publishers, 1996.
11. R. Gonzalez and R. Woods, *Digital Image Processing*, Addison-Wesley Publishing Company, 1992.

Clustering context properties of wavelet coefficients in automatic modelling and image coding

Joonas Lehtinen and Juha Kivijärvi

Published in Proceedings of IEEE 11th International Conference on Image Analysis and Processing, pages 151–156, Palermo, Italy, 2001.

Clustering context properties of wavelet coefficients in automatic modelling and image coding

Joonas Lehtinen and Juha Kivijärvi
Turku Centre for Computer Science (TUCS)
Lemminäisenkatu 14 A, 20520 Turku, Finland
Joonas.Lehtinen@cs.utu.fi

Abstract

An algorithm for automatic modelling of wavelet coefficients from context properties is presented. The algorithm is used to implement an image coder, in order to demonstrate its image coding efficiency. The modelling of wavelet coefficients is performed by partitioning the weighted context property space to regions. Each of the regions has a dynamic probability distribution stating the predictions of the modelled coefficients. The coding performance of the algorithm is compared to other efficient wavelet-based image compression methods.

1. Introduction

Although the amount of available bandwidth and storage space is continuously increasing, there is still growing demand for more efficient image coders in the industrial and medical applications. Most recent image coders are based on the coding of *wavelet representation* of the spatial image [8]. This is because wavelet transforms represent images very efficiently and the applicability of *transform based coders* is much better than that of alternative efficient methods, such as fractal image coders. The compression efficiency of a lossy wavelet image coder depends on four things: 1) the efficiency of *image transformation* method, 2) the *wavelet coefficient quantization* method, 3) the *coefficient modelling* algorithm, and 4) the *entropy coding* method. The image transformation converts the pixel representation of an image from the spatial domain to *wavelet domain*. The quantization method maps the wavelet coefficients to their approximations by reducing the number of possible values of the coefficient. This reduces the number of bits needed, but also introduces some errors into the image. The modelling step maps the quantized coefficients into a set of symbols with certain probabilities. The entropy coder encodes the symbols using these probabilities from the modelling step. The result of this process is a binary representation of the image data.

Several efficient wavelet image transforms have been proposed. In the context of lossy image compression, one of the most popular of these is recursive filtering with *Daubechies 9-7 wavelet filters* to produce *Mallat composition* [8]. Many coders use simple scalar quantization which gives fairly good results, but more advanced *rate distortion quantization* has also been used [2]. Arithmetic coding [1] can be used to perform entropy coding optimally, and thus it is widely used in high-performance image coders. Coefficient modelling algorithms vary from coder to coder, but most of them try to exploit the prior knowledge from the previously coded parts of the image to achieve a more efficient representation of the coefficients.

We introduce in this paper a new method for modelling the wavelet coefficients. The modelling decision is based on a set of *context properties* calculated from the previously coded coefficients. Context properties are expected to supply information relevant to the prediction of a coefficient. Some possible choices for context properties are individual previously coded coefficients, their absolute values and magnitudes, mean and variance of the neighbouring coefficients, measures for local gradient and other values describing the local properties of the image.

The method is independent of the context properties. In this way the problem of designing modelling heuristics is reduced to the problem of finding the most relevant context property measures. Because the other key components of a wavelet coding algorithm can be effectively implemented using existing techniques, one can construct efficient compression algorithm by selecting a good set of context properties.

The quantized coefficients are modelled by partitioning the weighted context property space with a clustering algorithm [4]. A dynamic probability distribution is assigned to each cluster. Such a distribution assigns a probability to each possible value of a coefficient mapped to the corresponding cluster. The coefficient is coded by encoding the

probability of the corresponding symbol by arithmetic coding.

Suitable weights for the context properties are usually selected by hand. In order to support automatic modelling, our method calculates the weights automatically from the correlations of the properties. This allows the coder to adapt more efficiently to different types of images, as the optimal selection of weights is image-dependent.

Rate distortion criterion [2] is used in the quantization of coefficients. Because the quantization depends on the partitioning of the context property space and the rate distortion quantization is used to construct the coefficient property space, optimal partitioning is not achievable. Furthermore, the quantization depends on the target bitrate, which makes selecting good quantization parameters and clustering difficult. Good quantization parameters can be found iteratively, but at the cost of a large time consumption.

We test the modelling algorithm and the optimization framework with a set of coefficient property selections, and compare the results to existing wavelet image compression algorithms. We also demonstrate the effect of different parameter selections on the compression performance.

The rest of the paper is organised as follows. In Section 2 we present the image coding algorithm used. Section 3 explains the clustering algorithm, and Section 4 summarizes the results of the test runs. Finally in the Section 5 we make the conclusions and discuss further research possibilities.

2. Image encoding algorithm

Our image encoding framework consists of three steps: 1) wavelet image transform, 2) finding good coding parameters and 3) coding the wavelet transformed image by using the parameters. In the first step the image is transformed to wavelet domain using Daubechies 9-7 wavelet filter recursively in order to construct a Mallat composition of the image.

The coding algorithm quantizes, models and encodes the *wavelet coefficient matrix* M considering the *coding parameter set* (Δ, W, C), where Δ is the *scalar quantization step*, $W = \{w_1, w_2, \ldots, w_m\}$ is a set of *weights for* m *context properties* and $C = \{c_1, c_2, \ldots, c_n\}$ is a set of n *context property partition centroids*. The context properties are defined as $P_{i,j} = \{p_1^{(i,j)}, p_2^{(i,j)}, \ldots, p_m^{(i,j)}\}$, where (i, j) are the indices of the corresponding coefficient $x_{i,j}$ in matrix M.

In the second step we try to find such a parameter set (Δ, W, C) that the compressed signal quality S measured with *peak signal to noise ratio* (PSNR) is maximized for the selected bitrate. We calculate the weights as

$$w_k = \frac{|r_k|^4}{\sqrt{\frac{1}{|M|}\sum_{i,j}(p_k^{(i,j)} - \bar{p}_k)^2}}, \qquad (1)$$

where r_k is the Pearson's correlation between p_k and the coefficients, $|M|$ is the number of elements in M, and $\bar{p}_k$ is the mean value of the property p_k. The context properties are calculated directly from M. The denominator normalizes the context properties by removing the effect of possibly different value ranges. The nominator assigns a larger weight to properties which have a strong correlation with the coefficients.

In order to find suitable Δ and C, we must determine both of them simultaneously, as they depend on each other. In step i we first search for a Δ_i, that satisfies the bitrate constraint by calculating the entropy of coding M with C_{i-1}. Then the coefficient properties of all the coefficients coded with parameter set (Δ_i, M, C_{i-1}) are used as a training set for the clustering algorithm (see Section 3) to find C_i. The steps for finding Δ and C are iterated in order to find a good solution. The initial selections Δ_0 and C_0 are not critical, as the algorithm quickly finds a reasonable solution.

The coding algorithm (see Alg. 2) can be used for performing two different tasks. First, we can calculate the entropy H, peak signal quality S, and context parameter space T, which can be used as a training set for the clustering algorithm. Second, we can encode the wavelet coefficient matrix M with arithmetic encoder by usind the described modelling algorithm.

The coding starts with an empty reconstruction pyramid R and a training set T. The coefficient matrix M is traversed starting from the top of the pyramid, subband by subband. For each subband, the dynamic distributions are initialized to contain zero probabilities. For each wavelet coefficient $x_{i,j}$ of a subband, the context properties $P_{i,j}$ are calculated from the reconstruction pyramid R and the set of property weights W. These parameters are known in the decoding step as the decoder must be able to repeat the calculations.

We define a distinct *zero-property set* $\hat{P}$ in order to more efficiently model the coefficients, which have highly quantized neighbours. If $P_{i,j} = \hat{P}$, the zero-property distribution D_0 is selected. Otherwise, the distribution D_k connected to the partition centroid c_k nearest to $P_{i,j}$ is chosen and $P_{i,j}$ is inserted into the training set T, which is used by the clustering algorithm for finding the set of partition centroids C.

Coefficient $x_{i,j}$ is quantized using rate distortion quantization $s = Q(|x_{i,j}|, \Delta, D_k)$, in which the idea is to find such a symbol s that the rate distortion criterion $|\frac{|x_{i,j}|}{\Delta} - s| - \frac{\ln 2}{4}\log_2 \mathbf{P}(s|D_k)$ is minimized. $\mathbf{P}(s|D_k)$ is the probability of symbol s in probability distribution D_k.

The symbol s is finally coded by arithmetic coding using the probability $\mathbf{P}(s|D_k)$. Furthermore, the sign of the coefficient must be coded if $s \neq 0$. The encoder keeps the reconstruction pyramid R up-to-date by inserting the new

encoded value $\text{sgn}(x_{i,j})\Delta s$ into R. The increase of the entropy is calculated to support the search for Δ, and the selected probability distribution D_k is adjusted to include the encoded symbol s. At the end of the coding process the signal quality of the compressed image is calculated from M and R. Calculations can be made by transforming the matrices to spatial domain or by calculating the error directly from the wavelet representations, if the applied filters are orthonormal.

Algorithm 1 The coding algorithm: $(\Delta, M, C, W) \rightarrow (T, S, H, B)$, where Δ is the quantization step size, M is the wavelet coefficient matrix to be coded, C is a set of context property partition centroids, W is a set of weights for properties, T is the training set containing the context properties, S is the signal quality, H is the resulting entropy and B is the generated bitstream.

▷ Zero entropy H, bitstream B, training set T and reconstruction pyramid R
for each pyramid level l starting from the top **do**
  for each subband b in level l **do**
    ▷ Clear distributions $D_0, D_1, \ldots, D_n$
    for each coefficient $x_{i,j}$ in the subband b **do**
      ▷ Calculate the context property vector $P_{i,j}$ from R using weights W
      if $P_{i,j} = \hat{P}$ **then**
        ▷ Select the zero-property probability distribution D_0
      else
        ▷ Select the probability distribution D_k with the corresponding centroid c_k nearest to the context property vector $P_{i,j}$
        ▷ Add context $P_{i,j}$ to the training set T
      ▷ Find the representative symbol s by rate distortion quantizing the absolute value of the coefficient $|x_{i,j}|$ using the selected distribution D_k and Δ
      ▷ Code the symbol s with arithmetic coder to B using the probability given by the selected distribution D_k
      ▷ If the value of the symbol is non-zero, encode the sign of the coefficient to B
      ▷ Insert the quantized coefficient value $\text{sgn}(x_{i,j})\Delta s$ to R
      ▷ Increase the entropy H by the number of bits used for coding the s and the sign
      ▷ Increase the probability of the symbol s in distribution D_k
▷ Calculate the signal quality S from M and R

3. Partitioning the context property space

Clustering means dividing a set of data objects into *clusters* in such a way that each cluster contains data objects that are similar to each other [5]. This idea can be applied in selection of a suitable probability distribution for coding the coefficient $x_{i,j}$. We want to partition the context property space by performing the clustering for a representative training set. This is feasible because clustering also defines a partitioning of the entire space. In this paper we are interested in clustering the context property vectors P_i, since we assume that the values of the coefficients with similar context properties are more likely to be close to each other than coefficients with dissimilar context properties.

In particular, we cluster a given set of N property vectors $\{P_1, \ldots, P_N\}$ into n clusters. Since each vector is assigned to exactly one cluster, we can represent a clustering by a *mapping* $U = \{u_1, \ldots, u_N\}$, where u_i defines the index of the cluster to which P_i is assigned. Furthermore, each cluster k has a *representative data object* c_k.

There are three selections to make before clustering: we should select the evaluation criterion, the number of clusters and the clustering algorithm. The evaluation criterion specifies which solutions are desirable. In this paper we aim to find solutions giving high compression ratios. Unfortunately, using the final compression ratio as the evaluation criterion is not a practical solution, because it would take too long to compute. Thus, we have assumed that minimizing the simple and frequently used *mean square error* (MSE) of the clustering will serve as an acceptable evaluation criterion. Given a mapping U and a set of cluster representatives $C = \{c_1, \ldots, c_n\}$, the evaluation criterion is calculated as:

$$f(U, C) = \frac{1}{Nm} \sum_{i=1}^{N} d(P_i, c_{u_i})^2 \tag{2}$$

where d is a distance function. We let d to be the *Euclidean distance* since it is the most commonly used distance function in clustering.

Thereafter, given a mapping U, the optimal choices for the cluster representatives C minimizing the function (2) are the cluster *centroids*:

$$c_k = \frac{\sum_{u_i=k} P_i}{\sum_{u_i=k} 1}, 1 \leq k \leq n \tag{3}$$

This formula is an integral part of the widely used *k-means* algorithm [6]. In this method, an initial solution is iteratively improved by mapping each vector to the cluster which has the nearest representative and then recalculating the cluster representatives according to formula (3). This never increases the MSE of the solution. The selection of the number of clusters is discussed in Section 4.

The problem of finding the clustering with minimal MSE is too complex to be performed exactly. However, many efficient heuristical algorithms have been developed for clustering. Thus, even though the optimal solution is unreachable, satisfactory solutions can usually be found.

We use *randomised local search* algorithm (RLS-2) as presented in [4]. The reason for this selection is that RLS-2 effectively utilizes the computation time. It finds reasonably good solutions quickly, but also carries on the optimization process and finally ends up with a very good solution.

RLS-2 starts with a random initial solution and iteratively tries to find a better one until the desired number of iterations has been reached. In each iteration, a new solution is created from the current solution by replacing a randomly selected cluster centroid with a randomly selected vector from the training set and applying two iterations of the k-means algorithm. The new solution is accepted as the new current solution if it is better than the old solution.

4. Test results

The above algorithm for clustering the context properties was tested by creating a coder (automatic context property based coding, ACPC) utilizing it. We used rate distortion quantization, Daubechies 9-7 filtering, Mallat composition and arithmetic coder as discussed in previous sections. To represent the context of the coefficient to be coded, we used the absolute values of one coefficient from the previous pyramid level in the Mallat composition and the 12 nearest neighbouring coefficients of the same subband as context properties.

If the number of context property partitions is too small, the clustering algorithm is unable to effectively model the different properties of the image, which results to poor compression performance. On the other hand, if the number of partitions is too large, the dynamic distributions do not have enough data to adapt accurately. The amount of extra header information required to represent the partitioning of the context property space also increases. In Figure 1 we demonstrate this behaviour by showing the results of compressing the test image *Lenna* using 0.25 bits per pixel and several selections of the number of partitions. One can see that a good choice of this number depends only slightly on the compression bitrate and seems to be between 10 and 20.

Partitioning of the context property space is demonstrated in Figure 2. Lenna is coded using 16 clusters and only two context properties: average magnitude and variability. We observe that the clustering algorithm has allocated more clusters (i.e. smaller partitions) to high density areas. Furthermore, the properties are not independent of each other.

To test the compression performance of our coder, we compared the compression results for three standard test images against three state of the art algorithms: SPIHT [7], SFQ [9] and C/B [3], see Table 1. Our algorithm uses 16 partitions for context property classification.

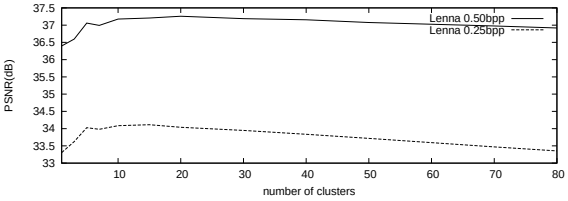


Figure 1. The signal quality for Lenna in PSNR using two bitrates and different numbers of clusters

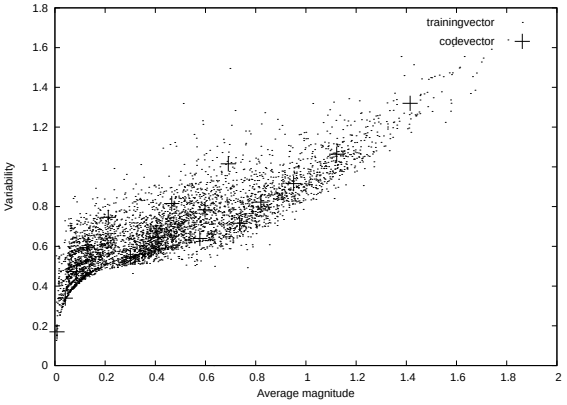


Figure 2. The average magnitude and variability context properties with partition centroids for Lenna, 0.25 bpp

The SPIHT algorithm is based on partitioning of hierarchical trees. It models the scalar quantized coefficients so efficiently that a separate entropy coding step is not necessarily required. Still, the results given here for SPIHT include an arithmetic compression step. Space frequency quantization (SFQ) includes a powerful modelling method which balances between spatial and scalar quantization and achieves excellent compression performance.

Context-based method (C/B) is somewhat similar to our method since it uses a set of dynamic discrete distributions for modelling the coefficients, and selects the proper distribution by evaluating the context. The partition selection method is straightforward in C/B: a weighted sum of nearby coefficients is quantized and the result is used to identify the distribution. The weights and the quantizer are carefully se-

Barbara 

bpp	SPIHT	SFQ	C/B	ACPC
0.20	26.64	26.26	27.33	**27.71**
0.25	27.57	27.20	28.48	**28.79**
0.50	31.39	31.33	32.37	**32.73**
1.00	36.41	36.96	37.61	**37.82**

Lenna 

bpp	SPIHT	SFQ	C/B	ACPC
0.20	33.16	**33.32**	33.24	33.12
0.25	34.13	34.33	**34.31**	34.13
0.50	37.24	37.36	**37.52**	37.23
1.00	40.45	40.52	**40.80**	40.62

Goldhill 

bpp	SPIHT	SFQ	C/B	ACPC
0.20	29.84	29.86	**29.94**	29.81
0.25	30.55	**30.71**	30.67	30.55
0.50	33.12	33.37	**33.41**	33.20
1.00	36.54	36.70	**36.90**	36.59

Table 1. Performance of SPIHT[7], SFQ[9], C/B[3] and our automatic context property based coding (ACPC) algorithm measured as PSNR for different bitrates.

lected.

The results show that the compression performance of our method is comparable with the other algorithms. It performs very well with test image *Barbara* which has a lot of high contrast edges. The performance in lower contrast images is slightly weaker.

Unfortunately the combination of the clustering algorithm and iterative search of the coding parameters is computationally very intensive. Our unoptimized coder implementation is much slower than the comparative programs and thus it is not practical for most real world applications. The speed of the coding algorithm without parameter search and clustering is comparable to C/B method. Each iteration in the parameter search requires the coder to be rerun with the new parameters and thus the actual coding time depends on the constraints set for the parameters and clustering. We believe that a faster implementation of the compression system can be made by relaxing the constraints and using faster search methods. The decoding process is much faster because it does not include the clustering and parameter search steps.

5. Conclusions

A new algorithm for automatic modelling the wavelet coefficients from arbitrary context properties was introduced. The algorithm was applied for implemention of an image coder. The coder achieves results comparable to state of the art methods. However, the encoding is rather slow, so the practical applications are limited to cases where the compression speed is irrelevant. Speed could be enhanced by a better implementation of the parameter optimization step.

The method is able to automatically find usable weights for context properties by utilizing correlation calculations. The presented compression framework allows one to easily evaluate different context property selections and thus helps to develop new efficient compression methods.

References

[1] V. Bhaskaran and K. Konstantinides. *Image and Video Compression Standards*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1995.

[2] C. Chrysafis. *Wavelet Image Compression Rate Distortion Optimizations and Complexity Reductions*. PhD thesis, December 1999.

[3] C. Chrysafis and A. Ortega. Efficient context-based entropy coding for lossy wavelet image compression. In *DCC, Data Compression Conference*, Snowbird, UT, March 1997.

[4] P. Fränti and J. Kivijärvi. Randomised local search algorithm for the clustering problem. *Pattern Analysis & Applications*, 3:358–369, 2000.

[5] L. Kaufman and P. J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley & Sons, New York, 1990.

[6] J. B. McQueen. Some methods of classification and analysis of multivariate observations. In *Proc. 5th Berkeley Symposium Mathemat. Statist. Probability*, volume 1, pages 281–296, University of California, Berkeley, CA, 1967.

[7] A. Said and W. A. Pearlman. A new fast and efficient image codec based on set partitioning in hierarchical trees. *IEEE Transactions on Circuits and Systems for Video Technology*, 6:243–250, June 1996.

[8] M. Vettereli and J. Kovačević. *Wavelets and Subband Coding*. Prentice Hall, Englewood Cliffs, NJ, 1995.

[9] Z. Xiong, K. Ramchandran, and M. T. Orchard. Space-frequency quantization for wavelet image coding. *IEEE Trans. Image Processing*, 1997.

A parallel genetic algorithm for clustering

Juha Kivijärvi, Joonas Lehtinen and Olli Nevalainen

To appear in Kay Chen Tan, Meng Hiot Lim, Xin Yao and Lipo Wang (editors), *Recent Advances in Simulated Evolution and Learning*, World Scientific, Singapore, 2004.

CHAPTER 3

A PARALLEL GENETIC ALGORITHM FOR CLUSTERING

Juha Kivijärvi, Joonas Lehtinen and Olli S. Nevalainen
Turku Centre for Computer Science (TUCS)
Department of Information Technology
University of Turku, 20014 Turku, Finland
E-mail: juha.kivijarvi@utu.fi

Parallelization of genetic algorithms (GAs) has received considerable attention in recent years. Reasons for this are the availability of suitable computational resources and the need for solving harder problems in reasonable time. We describe a new parallel self-adaptive GA for solving the data clustering problem. The algorithm utilizes island parallelization using a genebank model, in which GA processes communicate with each other through the genebank process. This model allows one to implement different migration topologies in an easy manner. Experiments show that significant speedup is reached by parallelization. The effect of migration parameters is also studied and the development of population diversity is examined by several measures, some of which are new.

1. Introduction

The objective in clustering is to divide a given set of data objects into a number of groups called *clusters* in such a way that similar objects belong to the same cluster whereas dissimilar objects are in different ones[1,2]. The problem appears as many variations in numerous fields of science such as data compression, pattern recognition, image analysis, medical data analysis, data mining, social sciences, bioinformatics, etc. The problem instances are commonly large in several respects: the dimensionality of data objects may be high, their number may be thousands or millions, and the number of clusters may be several hundreds. Thus the amount of computation needed for finding satisfactory solutions is often high, even if the hope of finding a true global optimum is abandoned.

In the present study we consider the case of *Euclidean* clustering. In

particular, we assume that the data objects can be considered as points in a Euclidean space and calculation of artificial cluster centers is meaningful. Furthermore, the number of clusters is expected to be known. This situation is met for example in vector quantization[3].

There exists a great number of algorithms for clustering[4,5]. These can be classified as *partitional* and *hierarchical*. Partitional algorithms aim to divide the given data into a number of clusters whereas hierarchical methods generate a hierarchy of clusterings of different sizes. Partitional algorithms are commonly *iterative*, i.e. they start with an initial solution and iteratively improve it. *Hierarchical* methods can be divided into *divisive* and *agglomerative* methods. They apply split and merge operations, respectively, until a clustering with the desired number of clusters has been reached[6].

General heuristic search techniques[7] have gained popularity in solving hard combinatorial optimization problems and clustering is not an exception. High quality results have been reported for e.g. *simulated annealing*, *tabu search*[8] and especially *genetic algorithms* (GAs)[9,10]. In the present study we concentrate on GAs since they are very effective while still conceptually simple and have been shown to achieve excellent results in clustering problems[11].

GAs perform *stochastic optimization* by applying stochastic evolution inspired operators to a set of candidate solutions. These operations include *mutation*, *crossover* and *selection*. There are several properties which have increased the popularity of GAs as a general framework for solving hard optimization problems. The quality of solutions found by GAs is in many cases excellent. The method is also easy to understand and an exact mathematical formulation is not needed; it suffices to determine a suitable representation for the individuals and a pertinent crossover operator.

All the above benefits, however, are not earned for free: GAs often suffer from very long running times so that a common complaint on their usefulness deals with the practicality of the approach. This drawback is underlined in many practical applications of GAs which include complicated objective functions or time constraints for problem solving. In addition, there are many design alternatives to choose from, and the final efficiency often strongly depends on details of the design and parameter values.

To overcome the latter difficulty, adaptive GAs have been developed[12,13]. A *self-adaptive genetic algorithm for clustering* (*SAGA*) is described in Ref. 14. In this algorithm, each individual contains several parameter values in addition to the actual solution. SAGA was demonstrated to be very robust and to achieve excellent results. The main drawback of

the method is the long running time. Fortunately, GAs are known to be easily parallelizable. Thus, using several interconnected processors one would expect to be able to reduce the actual running time considerably. Our primary goal is to speed up SAGA, but it is also interesting to see whether parallelization leads to algorithmic benefits as occasionally suggested. For a discussion of different models of parallelizing GAs and a literary survey, see Ref. 15. In Ref. 16 one can find in-depth mathematical analysis on different aspects of parallel GAs.

2. Clustering Problem

The clustering problem is defined as follows. Given a set of N data objects x_i, partition the data set into M clusters in such a way that similar objects are grouped together and dissimilar objects belong to different groups. Each object x_i has K *features* $x_i^{(k)}$. The features are assumed to be numerical and of the same scale. Mapping P defines a clustering by giving for each data object x_i the index p_i of the cluster it is assigned to. Furthermore, each cluster j has a *cluster representative* c_j.

We measure the *dissimilarity* (*distance*) between objects x_i and x_j by the *Euclidean distance*

$$d(x_i, x_j) = \sqrt{\sum_{k=1}^{K} (x_i^{(k)} - x_j^{(k)})^2}. \tag{1}$$

Our representation of a solution to the clustering problem includes both mapping and cluster representatives, i.e. a solution is of the form $\omega = (P, C)$ where $P = (p_1, \ldots, p_N)$ and $C = (c_1, \ldots, c_M)$. The objective is to find a solution with minimal *mean square error* (MSE), which is calculated as

$$e(\omega) = \frac{1}{NK} \sum_{i=1}^{N} d(x_i, c_{p_i})^2. \tag{2}$$

Given a mapping P, the optimal cluster representatives are the cluster *centroids*

$$c_j = \frac{\sum_{P_i=j} x_i}{\sum_{P_i=j} 1}, 1 \leq j \leq M, 1 \leq i \leq N. \tag{3}$$

The optimality of centroids leads to a simple and widely used clustering method, the *k-means* algorithm[17]. It improves an initial solution by repeatedly recalculating the cluster representatives using Eq. 3 and refining the mapping by assigning each object with the cluster which has the nearest

representative. Even though the results of the k-means are usually modest, it is highly useful as a hill-climbing method in more complicated algorithms.

3. Self-Adaptive Genetic Algorithm for Clustering

The self-adaptive genetic algorithm (SAGA) applied here[14] uses *individual level self-adaptation*[12,13], where each individual consists of a candidate solution to the problem and a set of *strategy parameters*. An individual ι is of the form $\iota = (\omega_\iota, \gamma_\iota, \psi_\iota, \mu_\iota)$, where $\omega_\iota = (P_{\omega_\iota}, C_{\omega_\iota})$ is a solution. The inclusion of both mapping and representatives allows us to make several speed optimizations. The strategy parameters included in ι are *crossover method* γ_ι, *mutation probability* ψ_ι and *noise range* μ_ι.

The general structure of SAGA is show in Alg. 1. Six crossover methods are available: random multipoint, centroid distance, largest partitions, multipoint pairwise, one-point pairwise and pairwise nearest neighbor crossover. All these methods exploit some problem-specific knowledge instead of considering the solutions as plain bit strings. A detailed description and discussion of the algorithm can be found in Ref. 14.

(1) Generate S random individuals to form the initial generation.
(2) Iterate the following T times.
  (a) Select S_B surviving individuals for the new generation.
  (b) Select $S - S_B$ pairs of individuals as the set of parents.
  (c) For each pair of parents (ι_a, ι_b) do the following:
    i. Determine the strategy parameter values $(\gamma_{\iota_n}, \psi_{\iota_n}, \nu_{\iota_n})$ for the offspring ι_n by inheriting each of them randomly from ι_a or ι_b.
    ii. Mutate the strategy parameter values of ι_n with the probability Ψ (a predefined constant).
    iii. Create the solution ω_{ι_n} by crossing the solutions of the parents. The crossing method is determined by γ_{ι_n}.
    iv. Mutate the solution of the offspring with the probability ψ_{ι_n}.
    v. Add noise to ω_{ι_n}. The maximal noise is ν_{ι_n}.
    vi. Apply k-means iterations to ω_{ι_n}.
    vii. Add ι_n to the new generation.
  (d) Replace the current generation by the new generation.
(3) Output the best solution of the final generation.

Algorithm 1: Self-adaptive genetic algorithm for clustering (SAGA).

4. Parallel Self-Adaptive GA for Clustering

Our parallel GA (*ParSAGA*) uses the *island parallelization* model[18], where Q GAs run independently and communicate with each other. The processes are seen as islands which occasionally send individuals ("emigrants") to other islands. We have implemented island parallelization using a *genebank model*. In the genebank model, instead of sending emigrants directly to other islands, islands communicate only with the *genebank process*. The genebank process maintains a *genebank*, a population of the best B individuals received from islands. For the communication purposes, three steps need to be added to SAGA, see Alg. 2. The genebank process, see Alg. 3, requires very little processor time and thus if e.g. Q processors are available, it could be run in side of one of the island processes.

(2) (e) Send an individual to the genebank.
(f) Receive an individual from the genebank and add it to the current population.
(g) Remove an individual from the current population.

Algorithm 2: Steps added to SAGA for island processes.

(1) Select coordinates κ_q for each island q.
(2) Repeat the following steps until a stopping condition is fulfilled.
(a) Sleep until an island process r makes a communication request.
(b) Receive an individual ι_r from r.
(c) Select an individual ι_s from the genebank.
(d) Send ι_s to island r.
(e) Add ι_r to the genebank.
(f) If the genebank contains $B + 1$ individuals, remove the worst individual from the genebank.
(3) Return the solution of the best individual in the genebank.

Algorithm 3: Genebank process.

Each island process is assigned a two-dimensional coordinate $\kappa_i = (x_i, y_i)$, where $0 \leq x_i, y_i \leq 1$, corresponding to the "location of the island

i". The cost of traveling from location κ_i to κ_j is defined as:

$$d_t(\kappa_i, \kappa_j) = \sqrt{\begin{array}{l}\min(|y_i - y_j|, 1 - |y_i - y_j|)^2 + \\ \max\left(\begin{array}{l}\min(\frac{x_i - x_j}{w}, 1 - x_i + x_j), \\ \min(x_j - x_i, \frac{1 + x_i - x_j}{w})\end{array}\right)^2\end{array}}. \quad (4)$$

The *direction control parameter* $w \in [0, 1]$ controls how much traveling from left to right is favored. The smaller the value of w, the stronger the imbalance in directions. Setting $w = 1$ gives no emphasis on the direction and $w = 0$ (which should be interpreted so that $\min(\frac{a}{0}, b) = b$) completely forbids traveling from right to left.

Note that if all the individuals in the genebank originate from the island making the communication request, the genebank process informs the island about this and sends no individual. As a result of this, the island process skips the steps 2(f) and 2(g).

The default, resulting in *island topology*, is to choose the coordinates κ_i randomly, let $w = 1$ and use *roulette wheel selection* with weights $\frac{1}{d_t(\kappa_s, \kappa_r)}$ for selecting an individual ι_s to be sent to island r from the genebank. Here κ_s is the location of the island ι_s originates from. The individuals originating from island r are not considered in selection.

This model is sufficiently general to allow us to employ several alternative network topologies. For example, the traditional *ring topology* is achieved by setting $\kappa_i = (\frac{i-1}{Q-1}, 0)$ for $i = 1, \ldots, Q$. The direction control parameter w controls whether the ring model is unidirectional ($w = 0$) or bidirectional ($w = 1$). Furthermore, island i is allowed to receive an individual from island j only if $d_t(\kappa_i, \kappa_j) \leq \sqrt{\frac{1}{Q-1}}$ (the distance between neighboring islands). *Torus topology*, where each processor is connected to four neighboring processors, can be realized by a similar setting in two dimensions.

Regarding the classification of parallel island GAs given by S.-C. Lin *et al.*[19], our parallel SAGA is an *asynchronous heterogeneous island GA with a static connection scheme*. The following settings for the migration parameters are applied:

- migration rate: one individual migrates
- migration frequency: migration occurs once in each generation
- migration topology: adjustable
- migration policy: several policies have been implemented, see Sec. 6.

Our model somewhat resembles the island model GA with a master pro-

cessor used by F. J. Marin *et al.*[20]. However, they differ in three important aspects. First, our model allows emulating several different topologies by adjusting the activities of the control process. It also applies asynchronous emigration, which simplifies the management of the island processes and adds to efficiency. Finally, the GAs on the islands are self-adaptive. The adaptive parallel GA by S. Tongchim and P. Chongstitvatana[21] is also quite different. Their algorithm utilizes population level adaptation whereas ParSAGA adapts on individual level. Furthermore, ParSAGA applies a very flexible neighborhood topology.

Even though the method scales up well, one could consider a case with a very large number of objects, fast processors and a slow communications network. Then, problems may be caused by the transmission of the object-to-cluster mapping of size $\Theta(N)$ which happens in each communication. Fortunately, one can speed up the communication simply by discarding the mapping from the sent individual and recalculating it each time an island receives an individual.

5. Statistical Measures for the Island Model

The operation of a parallel GA can be evaluated by *genotypic measures* or *phenotypic measures*. Genotypic measures consider the data representation of objects whereas phenotypic measures consider properties of solutions, in practice usually the fitness. M. Capcarrère *et al.*[22] have considered the case of cellular parallel GAs and defined several measures for describing the advancement of an evolutionary algorithm. The genotypic measures, including frequency of transitions, entropy of population and diversity indices, are related to the number of duplicate solutions. The phenotypic measures include performance (i.e. average error of solutions), diversity and ruggedness, which measures the dependency of individual's fitness from its neighbor's fitness.

The genotypic measures seem not to be applicable to an island model with very few duplicates, and we therefore give new measures for the island model. On the other hand, the phenotypic measures are applicable also for the coarse-grained case and we recall them shortly.

We assume that there are Q islands, and island q has a population of size s_q. The individuals on island q are $I_q = \{\iota_{q,i} | i = 1, \ldots, s_q\}$.

5.1. *Genotypic measures*

Genotypic measures deal with the representation of individuals. They are therefore specific to the particular problem in question and to the coding of individuals. In order to define our genotypic measures we first need to define the dissimilarity of two individuals ι_1 and ι_2. We ignore the strategy parameter values and concentrate on calculating the difference between solutions ω_{ι_1} and ω_{ι_2}. By defining a bijective *assignment* $a_{1\leftrightarrow 2}(\omega_{\iota_1}, \omega_{\iota_2}) = \langle i, \alpha_{1\leftrightarrow 2}(i)\rangle (i = 1, \ldots, M)$ for the clusters in the solutions, we can calculate the dissimilarity of individuals ι_1 and ι_2 simply by summing up the distances between the representatives of the associated clusters:

$$\delta_b(\iota_1, \iota_2) = \sum_{i=1}^{M} d(c_{\omega_{\iota_1}, i}, c_{\omega_{\iota_2}, \alpha_{1\leftrightarrow 2}(i)}) \tag{5}$$

where $c_{\omega_{\iota_1}, i}$ is the representative of the ith cluster in the solution of ι_1. The problem in using Eq. 5 is the proper selection of the assignment. The natural choice would be the assignment resulting to the smallest dissimilarity. Unfortunately, the problem of finding the optimal assignment is difficult. One could settle for a heuristically selected suboptimal assignment, but this would make the dissimilarity measure depend on the selection of the heuristic.

This problem can be solved by abandoning the demand for bijectivity. We define assignment $a_{1\to 2}$ so that each cluster of ω_{ι_1} is assigned with the nearest cluster in ω_{ι_2} measured by the distance between cluster representatives. Assignment $a_{2\to 1}$ is defined correspondingly. Now we can define the dissimilarity between ι_1 and ι_2 as the average of the distances calculated using these two assignments:

$$\begin{aligned}\delta_i(\iota_1, \iota_2) = \frac{1}{2}\Bigg[&\sum_{i=1}^{M} d(c_{\omega_{\iota_1}, i}, c_{\omega_{\iota_2}, \alpha_{1\to 2}(i)}) \\ &+ \sum_{i=1}^{M} d(c_{\omega_{\iota_2}, i}, c_{\omega_{\iota_1}, \alpha_{2\to 1}(i)})\Bigg].\end{aligned} \tag{6}$$

This is a computationally feasible definition since the two assignments can be determined in $\mathcal{O}(M^2K)$ time.

A completely different approach to defining dissimilarity is to concentrate on mappings instead of cluster representatives. A straightforward way to define dissimilarity using mappings is to define an $N \times N$ binary matrix $\mathbf{B}^{(\iota)}$ for mapping P_{ω_ι} so that $\mathbf{B}^{(\iota)}_{i,j} = 1$ iff $p_{\omega_\iota, i} = p_{\omega_\iota, j}$, i.e. objects i and j are mapped to the same cluster in solution ω_ι. The mapping dissimilarity

of two individuals is the number of differing elements in the corresponding matrices:

$$\delta_m(\iota_1, \iota_2) = \sum_{i=1}^{N}\sum_{j=1}^{N} |\mathbf{B}_{i,j}^{(\iota_1)} - \mathbf{B}_{i,j}^{(\iota_2)}|. \tag{7}$$

Due to the size of the matrices the calculation of this dissimilarity measure takes $\mathcal{O}(N^2)$ time.

Now we can define the *average dissimilarity* $A^{(\delta)}(q)$ of island q as

$$A^{(\delta)}(q) = \sum_{i=1}^{s_q}\sum_{j=1}^{i-1} \frac{2\delta(\iota_{q,i}, \iota_{q,j})}{s_q(s_q-1)} \tag{8}$$

and the *average dissimilarity of the island model* as

$$A^{(\delta)} = \frac{\sum_{q=1}^{Q} s_q(s_q-1)A^{(\delta)}(q)}{\sum_{q=1}^{Q} s_q(s_q-1)}. \tag{9}$$

Here any suitable dissimilarity measure δ can be applied resulting to e.g. *average assignment dissimilarity* $A^{(\delta_i)}$ and *average mapping dissimilarity* $A^{(\delta_m)}$. The average dissimilarity measures the diversity of the individuals. Obviously, it tends to zero as the population converges to several copies of a single individual. By observing the development of $A^{(\delta)}$ we can get an understanding on the speed of convergence.

5.2. *Phenotypic measures*

Since Eq. 2 is the optimization criterion of the solution, we can evaluate the population of an island by calculating the *average distortion on island* q

$$A^{(e)}(q) = \frac{1}{s_q}\sum_{i=1}^{s_q} e(\omega_{\iota_{q,i}}) \tag{10}$$

and the *standard deviation of distortion on island* q:

$$\sigma^{(e)}(q) = \sqrt{\frac{1}{s_q}\sum_{i=1}^{s_q}\left(A^{(e)}(q) - e(\omega_{\iota_{q,i}})\right)^2}. \tag{11}$$

Furthermore, we can define the *average distortion of the island model*

$$A^{(e)} = \frac{\sum_{q=1}^{Q} s_q A^{(e)}(q)}{\sum_{q=1}^{Q} s_q} \tag{12}$$

and the *standard deviation of distortion of the island model*

$$\sigma^{(e)} = \sqrt{\frac{\sum_{q=1}^{Q} s_q \left[(\sigma^{(e)}(q))^2 + \left(A^{(e)}(q) - A^{(e)} \right)^2 \right]}{\sum_{q=1}^{Q} s_q}}. \tag{13}$$

The average distortion gives information on the progress of the algorithm and it can be compared to the distortion of the best solution for a rough view on diversity. The standard deviation of diversity measures phenotypic diversity more accurately.

6. Results

6.1. *Test setting*

We have used four different test problems, see Table 1. Three of them originate from the field of vector quantization and one from a biological application of clustering. *Bridge* consists of 4×4 pixel blocks sampled from a gray-scale image with image depth of 8 bits per pixel. *Bridge2* has the blocks of *Bridge* after a BTC-like quantization into two values according to the average pixel value of a block. The cluster representatives for *Bridge2* are rounded to binary vectors. *Lates mariae* contains data from pelagic fishes of Lake Tanganyika. The data originates from a research, in which the occurrence of 52 different DNA fragments was tested for each fish sample using RAPD analysis and a binary decision was obtained whether the fragment was present or absent. The cluster representatives are real vectors. *Miss America* has been obtained by subtracting two subsequent image frames of a video image sequence and constructing 4×4 pixel blocks from the residuals.

Table 1. Dimensions of the test problems

data set	attributes	objects	clusters
Bridge	16	4096	256
Bridge2	16	4096	256
Lates mariae	52	215	8
Miss America	16	6480	256

The parameters for SAGA are the default parameters from Ref. 14: parameter mutation probability $\Psi = 5\%$, number of k-means iterations $G = 2$, population size $S = 45$ and the roulette wheel selection method.

The tests were run on a single two-processor (400 MHz each) computer using 10 island processes thus emulating the situation of 10 interconnected

computers. The communication costs are very low compared to the computational costs, so the results should be well comparable to the situation with an actual computer network. The amount of processor time consumed by each process was considered instead of the real time.

The statistical significance of differences has been verified by Student's t-test, $p < 0.05$.

6.2. *Test results*

ParSAGA was compared to seven other clustering algorithms, see Table 2. The results of k-means[17,23] and *stochastic relaxation* (SR)[24] are averages of 100 runs with random initializations. Divisive and agglomerative hierarchical methods are respectively represented by *splitting method with local repartitioning* (SLR)[25] and *Ward's method*[26]. These results were not repeated since the methods are deterministic. The results of *randomised local search* (RLS-2)[27], genetic algorithm (GA)[11], self-adaptive genetic algorithm (SAGA)[14] and ParSAGA are averages of 20 independent runs. GA and SAGA were run 1000 generations with a population of 45 individuals. ParSAGA was run 100 generations with 10 islands of 45 individuals. The result of a single ParSAGA run is the MSE of the best solution in the genebank after all islands have completed their run.

We observe that ParSAGA achieves results similar to SAGA, i.e. results of excellent quality whereas simpler methods give considerably weaker results. An exception here is *Lates mariae* for which, in addition to the GAs, RLS-2 arrives each time at the same result. While there is no significant difference between the results of ParSAGA and SAGA, the difference between ParSAGA and GA is significant on other sets than *Lates mariae* ($p = 4.6 * 10^{-10}$, $7.3 * 10^{-10}$, $3.0 * 10^{-9}$ for *Bridge*, *Bridge2* and *Miss America*, respectively). When comparing to other methods, the significance of difference is obvious. However, Ward's method reached almost as good a result for the easy problem *Lates mariae*. For the other problems Ward's method was less successful.

The rest of the results are for *Bridge* only and they are averages of 20 runs of 100 generations and 10 islands of 45 individuals as above. See Sec. 4 for the default migration topology parameters. Table 3 compares different migration policies. Best results are achieved by sending the best individuals and replacing the worst, as one would expect. Sending the best gives constantly better results than sending random individuals ($p = 8.3 * 10^{-8}$, 0.036, $6.5 * 10^{-5}$ for replacing worst, sent and random, respectively).

Also, replacing the sent individual seems to be a bad policy when the best is sent ($p = 0.0077$).

Table 2. Comparison of clustering methods. The best result and the results not statistically significantly worse ($p < 0.05$) are boldfaced.

	Bridge		*Bridge2*	
	av. MSE	st. dev	av. MSE	st. dev
k-means	180.073	1.442	1.489	0.015
SLR	170.221	0.000	1.362	0.000
Ward's method	169.253	0.000	1.429	0.000
SR	162.607	0.275	1.469	0.014
RLS-2	164.220	0.251	1.264	0.004
GA	161.403	0.089	1.263	0.004
SAGA	**161.183**	0.102	**1.252**	0.003
ParSAGA	**161.153**	0.100	**1.254**	0.003
	Lates mariae		*Miss America*	
	av. MSE	st. dev	av. MSE	st. dev
k-means	0.0703	0.0054	5.963	0.056
SLR	0.0662	0.0000	5.398	0.000
Ward's method	0.0627	0.0000	5.507	0.000
SR	0.0683	0.0045	5.265	0.013
RLS-2	**0.0626**	0.0000	5.262	0.019
GA	**0.0626**	0.0000	5.108	0.004
SAGA	**0.0626**	0.0000	**5.100**	0.003
ParSAGA	**0.0626**	0.0000	**5.099**	0.002

Table 3. Comparison of different migration policies.

send	replace	av. MSE	st.dev.
best	worst	161.153	0.100
best	sent	161.262	0.140
best	random	161.195	0.098
random	worst	161.357	0.095
random	sent	161.347	0.105
random	random	161.344	0.111

Table 4 compares three different migration topologies. The topology doesn't seem to have a great effect on results, at least with this small a number of islands, since the differences are not statistically significant.

The effect of the direction control parameter can be seen in Table 5. It

Table 4. Comparison of different migration topologies.

	av. MSE	st.dev.
island	161.153	0.100
ring	161.170	0.133
torus (2x5)	161.204	0.096

turns out that restricting the direction of emigration ($w = 0$) is disadvantageous ($p = 0.040, 0.043$ for island and ring topology, respectively, for $w = 1$ versus $w = 0$).

Table 5. Effect of the direction control parameter.

	island topology		ring topology	
w	av. MSE	st.dev.	av. MSE	st. dev.
1	161.153	0.100	161.170	0.133
0.5	161.210	0.107	161.192	0.119
0	161.236	0.144	161.249	0.103

Increasing the genebank size from the default 20 to 100 did not give significant improvement. However, the difference between this result (161.117 with standard deviation of 0.098) and the result of SAGA (see Table 2) is significant ($p = 0.044$).

Figure 1 shows the speedup of ParSAGA in comparison to SAGA at various moments of time in two cases: 1000 generations of SAGA with a population of 45 individuals and 100 generations of SAGA with a population of 450 individuals. Speedup is calculated as a function of time so that speedup

$$s(t) = \frac{t}{T_{ParSAGA}(R_{SAGA}(t))} \tag{14}$$

where $R_{SAGA}(t)$ is the MSE of the best result found by SAGA after running t seconds (averaged over 20 runs) and $T_{ParSAGA}(R)$ is the time ParSAGA needs to find such a solution ω that $e(\omega) \leq R$ (also averaged over 20 runs). This approach has been selected because the methods are able to keep on finding better results and thus a single point of time for speedup observations can not be chosen justifiably. Since ParSAGA is run with 10 islands, speedup of 10 would be linear.

We observe that ParSAGA is very fast in comparison to SAGA with a large population even though this SAGA setting resembles more closely the setting of ParSAGA. This is because SAGA would need many more genera-

tions to successfully handle this large a population with the roulette-wheel selection. After 100 generations the average result is only 161.963 with standard deviation of 0.188. In ParSAGA, the large population is essentially divided into smaller intercommunicating populations thus resulting to excellent speedup. On the other hand, since ParSAGA in practice applies a considerably larger population than SAGA with a population of 45 individuals, it can preserve genetic variation longer and thus is still able to achieve regular progress when SAGA can only find better solutions occasionally. This is illustrated by the almost linear final portion of the 1000*45 curve.

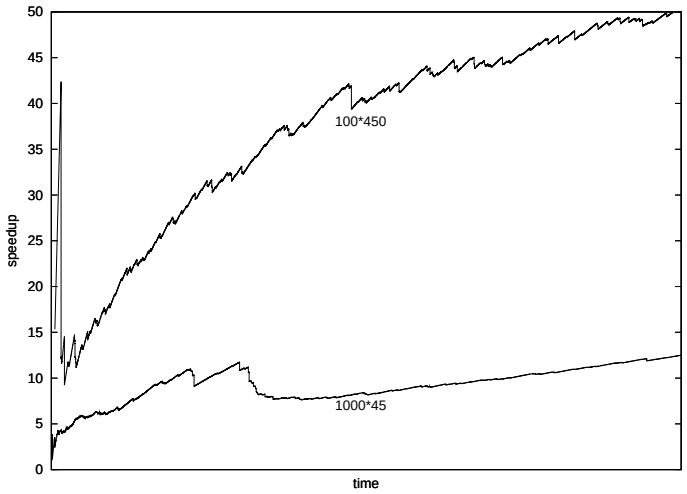


Fig. 1. Speedup of ParSAGA over 1000 generations of SAGA with a population of 45 individuals and 100 generations of SAGA with a population of 450 individuals as a function of time spent by SAGA.

Figures 2 – 4 show the development of several statistical measures in three different cases. In Fig. 2 the default parameters are used, in Fig. 3 migration is performed only once every 10 generations (instead of every generation) and in Fig. 4 20 k-means iterations are applied to each solution instead of default 2. The diversity measures clearly show that reducing the migration frequency slows down the decline of diversity. This is probably most apparent in the graph of average assignment dissimilarity ($A^{(\delta_i)}$), but

also the difference between average distortion ($A^{(e)}$) and best distortion as well as the standard deviation of distortion ($\sigma^{(e)}$) portray the same behavior. Increasing the number of k-means iterations leads to rapid decline of diversity in the beginning, as one would expect. However, after a while the diversity sets on roughly the same level as with the default parameters. It should be noted that due to the smart k-means implementation[23], 20 k-means iterations per solution is only slightly slower than the default 2. The increase of k-means iteration count does not change the quality of results significantly (161.122 with standard deviation of 0.104).

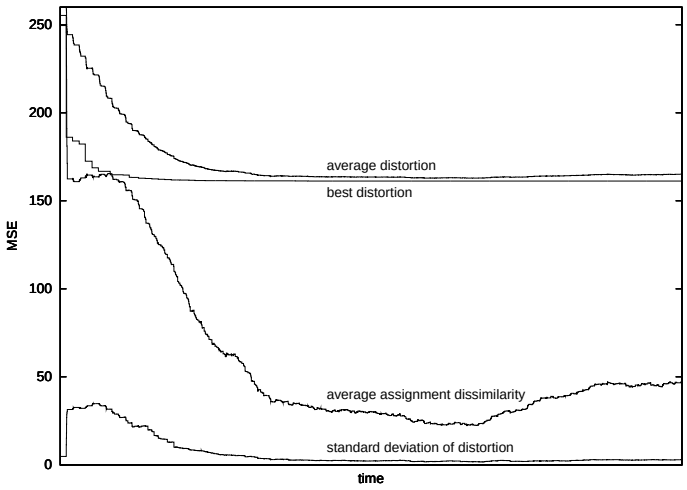


Fig. 2. Development of several measures for default parameters.

The development of the average mapping dissimilarity ($A^{(\delta_m)}$) is shown in Fig. 5. Here, all the previous cases have been plotted in a single figure. Same observations as above can also be made from this figure.

One further thing to notice about the measures is the fact that even though the phenotypic diversity declines steadily, genotypic diversity can still occasionally increase noticeably. This might suggest that the search has found new promising areas of problem space to study closer.

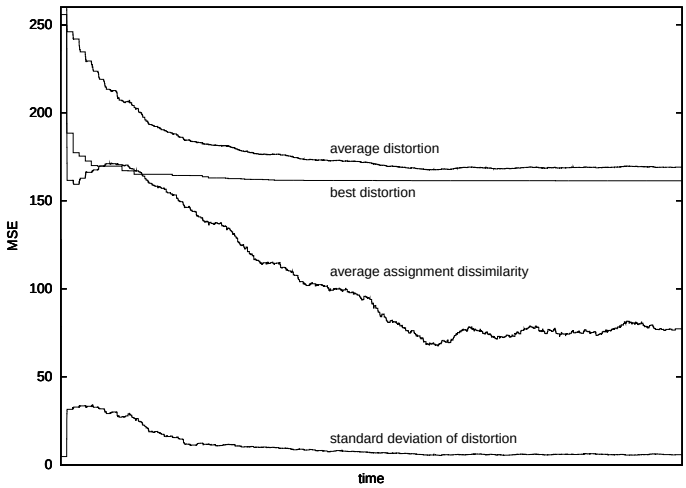


Fig. 3. Development of several measures for migration frequency once in 10 generations.

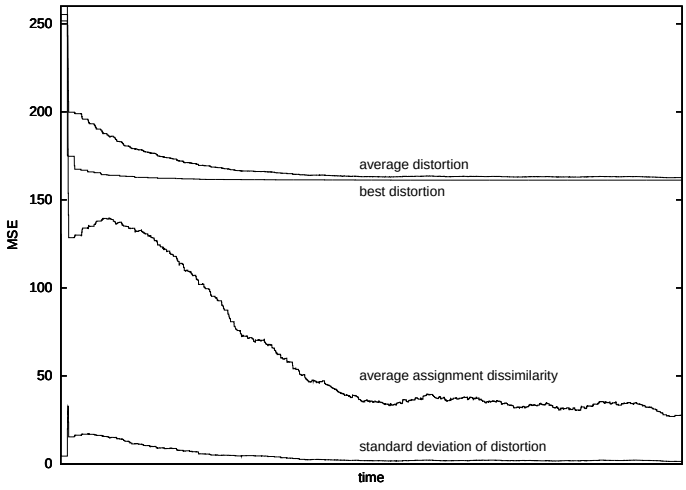


Fig. 4. Development of several measures for 20 k-means iterations per solution.

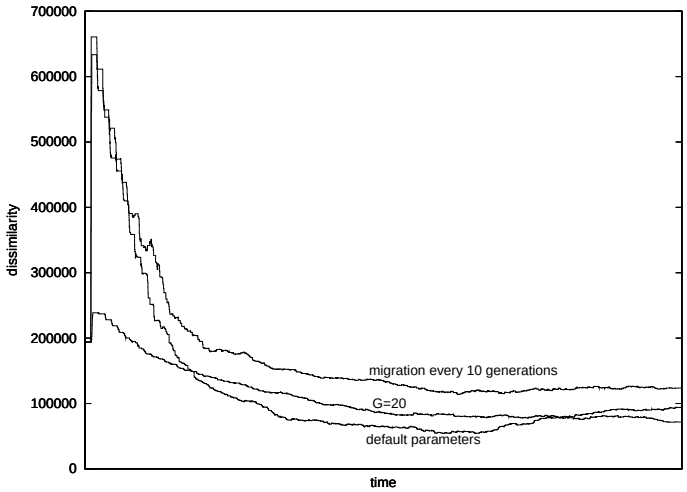


Fig. 5. Development of the average mapping dissimilarity for default parameters, for migration frequency once in 10 generations and for 20 k-means iterations per solution.

7. Conclusions

Parallelization of a self-adaptive genetic algorithm for clustering was studied. Our parallel algorithm applied the genebank model for organizing the emigration. In the model, all the communication between SAGA processes is directed through the genebank process, which maintains a collection of the best individuals received. This general model has the advantage of allowing one to implement different topologies flexibly by simple parameter adjustments.

The parallel SAGA achieved results of the same quality as the sequential SAGA but in considerably shorter time. SAGA and ParSAGA outperform the other tested methods in all the cases except for the easy problem *Lates mariae* where several algorithms consistently reached the same solution (Table 2).

Speedup of ParSAGA was studied against two different SAGA setups (Fig. 1). When ParSAGA is compared to SAGA with a population of corresponding size, i.e. the number of islands times the population size of an island, ParSAGA is remarkably efficient. Superlinear speedup could be claimed here, even though it is obviously due to different functioning of the algorithms. This efficiency is explained by SAGA's inability to handle such

a large population reasonably fast. On the other hand, when the population size of SAGA is set to the population size of a single island and the difference in the amount of work is compensated by increasing the number of generations, the speedup is close to linear. However, in this case the larger effective population size of ParSAGA allows it to retain more diversity and thus keep on finding better solutions more efficiently in the later stages of the search process. Thus, slight superlinearity could also be claimed here when the speedup is observed near the end of the search time chosen.

Comparison of different migration policies (Table 3) showed that sending the best and replacing the worst individuals is the most effective migration policy. This policy causes the highest selection pressure among the methods studied. Restricting the direction of migration turned out to be disadvantageous (Table 5) even though the selection of actual topology was found insignificant (Table 4).

We gave two new genotypic diversity measures for the parallel GA. Average assignment dissimilarity measures the average distances between matching cluster centroids of two individuals and average mapping dissimilarity compares two mappings.

Several things can be learned by observing the development of the statistical measures (Figs. 2 – 5). First, the measures clearly demonstrate that by reducing the frequency of migration, the decline of genetic variation can be effectively slowed down. Furthermore, the presented genotypic measures show that even though there are no considerable changes in diversity measured by phenotypic measures, genotypic diversity may still increase noticeably. This may be due to finding new promising areas to search.

Figure 4 demonstrates another interesting phenomenon. When the number of k-means iterations per solutions is increased to 20, diversity drops very fast in the beginning, as expected. However, even though phenotypic measures suggest that genetic variation stays very low, genotypic measures reveal that the diversity of solutions is actually similar to the default case. This also explains why this setting does not lead to inferior results even though the genetic variation seems to decrease rapidly when examined by ordinary phenotypic measures.

Finally, the usefulness of the statistical measures is not limited to learning important things about the behavior of the algorithm and the effect of different parameter settings. They could also be used in guiding the algorithm. Parameters controlling the operation could be adjusted according to the values of these measures. However, this would be more appropriate with other adaptation schemes, see Ref. 12. Since the calculation of mea-

sures is rather slow, one might consider calculating an approximation from a randomly selected sample instead of applying full measures.

References

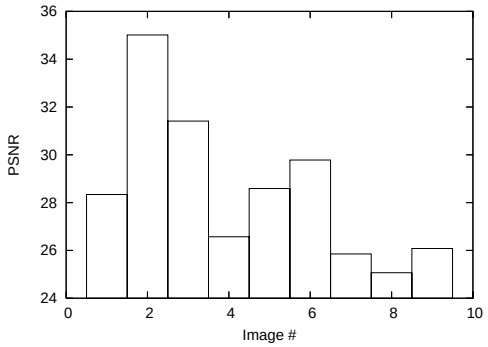
1. B. S. Everitt, *Cluster Analysis (3rd ed.)* (Edward Arnold / Halsted Press, London, 1993).
2. L. Kaufman and P. J. Rousseeuw, *Finding Groups in Data: An Introduction to Cluster Analysis* (John Wiley & Sons, New York, 1990).
3. A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression* (Kluwer, Dordrecht, 1992).
4. A. K. Jain and R. Dubes, *Algorithms for Clustering Data* (Prentice Hall, Englewood Cliffs, 1988).
5. A. K. Jain, M. N. Murty and P. J. Flynn, *ACM Comp. Surv.* **31**, 264 (1999).
6. T. Kaukoranta, *Iterative and Hierarchical Methods for Codebook Generation in Vector Quantization* (Turku Centre for Computer Science, Turku, 1999).
7. C. R. Reeves, Ed., *Modern Heuristic Techniques for Combinatorial Problems* (Blackwell, Oxford, 1993).
8. P. Fränti, J. Kivijärvi and O. Nevalainen, *Patt. Rec.* **31**, 1139 (1998).
9. J. H. Holland, *Adaptation in Natural and Artificial Systems* (University of Michigan Press, Ann Arbor, 1975).
10. D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning* (Addison-Wesley, Reading, 1989).
11. P. Fränti, J. Kivijärvi, T. Kaukoranta and O. Nevalainen, *Comp. J.* **40**, 547 (1997).
12. R. Hinterding, Z. Michalewicz and A. E. Eiben, in *Proc. 1997 IEEE International Conference on Evolutionary Computation* (IEEE, New York, 1997), p. 65.
13. G. Magyar, M. Johnsson and O. Nevalainen, *IEEE Trans. Evol. Comp.* **4**, 135 (2000).
14. J. Kivijärvi, P. Fränti and O. Nevalainen, *J. Heur.* **9**, 113 (2003).
15. J. Kivijärvi, J. Lehtinen and O. Nevalainen, TUCS Technical Report 469 (Turku Centre for Computer Science, Turku, 2002).
16. E. Cantú-Paz, *Efficient and Accurate Parallel Genetic Algorithms* (Kluwer, Boston, 2000).
17. J. B. McQueen, in *Proc. 5th Berkeley Symposium on Mathematical Statistics and Probability*, Eds. L. M. Le Cam and J. Neyman (University of California Press, Berkeley, 1967), p. 281.
18. M. Tomassini, in *Evolutionary Algorithms in Engineering and Computer Science*, Eds. K. Miettinen, M. M. Mäkelä, P. Neittaanmäki and J. Periaux (John Wiley & Sons, Chichester, 1999), p. 113.
19. S.-C. Lin, W. F. Punch and E. D. Goodman, in *Proc. 6th IEEE Symposium on Parallel and Distributed Processing* (IEEE, New York, 1994), p. 28.
20. F. J. Marin, O. Trelles-Salazar and F. Sandoval, in *Parallel Problem Solving from Nature - PPSN III, International Conference on Evolutionary Com-*

putation, Eds. Y. Davidor, H.-P. Schwefel and R. Männer (Springer-Verlag, New York, 1994), p. 534.
21. S. Tongchim and P. Chongstitvatana, in *Proc. 1st International Conference on Intelligent Technologies*, Eds. V. Kreinovich and J. Daengdej (Assumption University, Bangkok, 2000), p. 94.
22. M. Capcarrère, A. Tettamanzi, M. Tomassini and M. Sipper, *Evol. Comp.* **7**, 255 (1999).
23. T. Kaukoranta, P. Fränti and O. Nevalainen, *IEEE Trans. Image Proc.* **9**, 1337 (2000).
24. K. Zeger and A. Gersho, *Electronics Lett.* **25**, 896 (1989).
25. P. Fränti, T. Kaukoranta and O. Nevalainen, *Optical Engineering* **36**, 3043 (1997).
26. J. H. Ward, *J. American Stat. Ass.* **58**, 236 (1963).
27. P. Fränti and J. Kivijärvi, *Pattern Analysis & Appl.* **3**, 358 (2000).

Publication errata

Distortion limited wavelet image codec

Page 88: The PSNR values given in the Figure 6 are incorrect. The correct values are given in the diagram below.



Predictive depth coding of wavelet transformed images

Page 100: The column marked as BPP in the Table 1 contains scalar quantization constant q, not BPP values. The corresponding BPP values for each q are given in the table below.

q	BPP		
	lena	barbara	goldhill
0.50	3.28	3.66	3.87
0.25	2.29	2.65	2.90
0.15	1.57	1.92	2.15
0.10	1.05	1.42	1.59
0.05	0.48	0.84	0.81
0.01	0.10	0.19	0.11

Turku Centre for Computer Science
TUCS Dissertations

24. **Linas Laibinis**, Mechanised Formal Reasoning About Modular Programs
25. **Shuhua Liu**, Improving Executive Support in Strategic Scanning with Software Agent Systems
26. **Jaakko Järvi**, New Techniques in Generic Programming : C++ is More Intentional than Intended
27. **Jan-Christian Lehtinen**, Reproducing Kernel Splines in the Analysis of Medical Data
28. **Martin Büchi**, Safe Language Mechanisms for Modularization and Concurrency
29. **Elena Troubitsyna**, Stepwise Development of Dependable Systems
30. **Janne Näppi**, Computer-Assisted Diagnosis of Breast Calcifications
31. **Jianming Liang**, Dynamic Chest Images Analysis
32. **Tiberiu Seceleanu**, Systematic Design of Synchronous Digital Circuits
33. **Tero Aittokallio**, Characterization and Modelling of the Cardiorespiratory System in Sleep-disordered Breathing
34. **Ivan Porres**, Modeling and Analyzing Software Behavior in UML
35. **Mauno Rönkkö**, Stepwise Development of Hybrid Systems
36. **Jouni Smed**, Production Planning in Printed Circuit Board Assembly
37. **Vesa Halava**, The Post Correspondence Problem for Marked Morphisms
38. **Ion Petre**, Commutation Problems on Sets of Words and Formal Power Series
39. **Vladimir Kvassov**, Information Technology and the Productivity of Managerial Work
40. **Franck Tétard**, Managers, Fragmentation of Working Time, and Information Systems
41. **Jan Manuch**, Defect Theorems and Infinite Words
42. **Kalle Ranto**, Z_4-Goethals Codes, Decoding and Designs
43. **Arto Lepistö**, On Relations between Local and Global Periodicity
44. **Mika Hirvensalo**, Studies on Boolean Functions Related to Quantum Computing
45. **Pentti Virtanen**, Measuring and Improving Component-Based Software Development
46. **Adekunle Okunoye**, Knowledge Management and Global Diversity - A Framework to Support Organisations in Developing Countries
47. **Antonina Kloptchenko**, Text Mining Based on the Prototype Matching Method
48. **Juha Kivijärvi**, Optimization Methods for Clustering
49. **Rimvydas Rukšėnas**, Formal Development of Concurrent Components
50. **Dirk Nowotka**, Periodicity and Unbordered Factors of Words
51. **Attila Gyenesei**, Discovering Frequent Fuzzy Patterns in Relations of Quantitative Attributes
52. **Petteri Kaitovaara**, Packaging of IT Services – Conceptual and Empirical Studies
53. **Petri Rosendahl**, Niho Type Cross-Correlation Functions and Related Equations
54. **Péter Majlender**, A Normative Approach to Possibility Theory and Soft Decision Support
55. **Seppo Virtanen**, A Framework for Rapid Design and Evaluation of Protocol Processors
56. **Tomas Eklund**, The Self-Organizing Map in Financial Benchmarking
57. **Mikael Collan**, Giga-Investments: Modelling the Valuation of Very Large Industrial Real Investments
58. **Dag Björklund**, A Kernel Language for Unified Code Synthesis
59. **Shengnan Han**, Understanding User Adoption of Mobile Technology: Focusing on Physicians in Finland
60. **Irina Georgescu**, Rational Choice and Revealed Preference: A Fuzzy Approach
61. **Ping Yan**, Limit Cycles for Generalized Liénard-type and Lotka-Volterra Systems
62. **Joonas Lehtinen**, Coding of Wavelet-Transformed Images

Turku
Centre *for*
Computer
Science

Lemminkäisenkatu 14 A, 20520 Turku, Finland | www.tucs.fi

University of Turku

- Department of Information Technology
- Department of Mathematics

Åbo Akademi University

- Department of Computer Science
- Institute for Advanced Management Systems Research

Turku School of Economics and Business Administration

- Institute of Information Systems Sciences

ISBN 952-12-1568-2
ISSN 1239-1883