Adaptive Objectbased Image Compression With Wavelet Methods

Diplomarbeit

zur Erlangung des akademischen Grades
Diplom-Ingenieur
an der Naturwissenschaftlichen Fakultät
der Universität Salzburg

eingereicht von
Dominik Engel

Betreuer
Ao.Univ.-Prof. Mag. Dr. Andreas Uhl

Salzburg, im November 2002

Abstract

With the emergence of objectbased image coding standards and the successful use of adaptive image coding techniques for various applications, the combination of both fields in objectbased adaptivity is only logical, as it promises not only better compression efficiency, but also increased functionality.

In this work, we present various approaches of objectbased and adaptive image compression using wavelets. After discussing the basic principles of the wavelet transform and zerotree coding in general, we present recent propositions how to adapt these techniques to arbitrarily shaped objects and compare different methods and implementations. We show how objectbased adaptivity can be achieved and how well it performs compared to other methods in an objectbased framework. Finally, we present an image coding framework that uses objectbased adaptive wavelet transformation in order to increase compression performance and functionality in a non-objectbased context.

Contents

Acknowledgements

This thesis evolved in the environment of the research group *Ganesh*, which is based at the University of Salzburg. I am indebted to the Austrian Science Fund (FWF) for funding my research while I was a member of *Ganesh* (project no. P13732).

I want to thank Andreas Uhl, the head of *Ganesh*, for supervising my thesis. Special thanks to Rade Kutil, Andreas Pommer and Thomas Schell for their help with wavelets in general and the *Ganesh++* codec in particular, and to all members of the *Ganesh* group for their help, the interesting discussions and – not to forget – the good time at all of the *Ganesh* social events.

I want to thank my family and friends for their great support, advice and emotional backup.

Intellectual work is always a social process. Many people contributed to this thesis directly and indirectly. I want to thank all who are not mentioned here explicitly.

Chapter 1

Introduction

1.1 Digital Image Compression

With the advent of broadband networking infrastructure, the art of compressing data seems to have lost most of its importance. However, with more demanding applications, the amount of data to be transferred increased tremendously compared to the increase in networking bandwidth. For example, a 2 hour video sequence in raw data with 24 frames per second of size 512×512 in RGB with 8bpp per channel would use $7200 * 512^2 * 24 * 24$ bit = 129600 MB. Image databases, like the FBI database for fingerprints, containing such a vast amount of data that storage without compression would be impossible, are another example for the need of good compression techniques.

For image and video applications, data compression is more important than ever and has to be adapted to new forms of use. Desired improvements are not only higher compression results, but an increase in functionality at the same time.

1.1.1 Lossless Versus Lossy Data Compression

We distinguish two basic types of data compression: *lossless* and *lossy*. In lossless compression the original data can be restored exactly from the compressed bitstream, whereas with lossy compression, the restoration is not exact. The use of either of the two types depends on the field of application. For medical or financial application, for example, loss of data is not acceptable. In the case of medical image processing, the accuracy of a region in a tomographical image may be vital for a patient's fate. On the other hand there is a large field of applications where loss of data is acceptable. Video conferencing, for example, does not depend on the exact restoration of the original data but can easily trade absolute accuracy in data restoration for low delay and fast throughput.

Lossy compression makes use of lossless compression. In lossy compression, the data is transformed to a "perspective" more favorable for compression. It is the transformed data that is then quantized. And it is the quantized data that is compressed losslessly in the last step. Summing up, the principal parts of a lossy compression are in most cases

- Transformation

  A transformation of the signal decorelates the signal and compacts its energy.

- Quantization

  In this step virtually all of the data loss occurs. The quantization step simplifies the signal in order to make it better suited for lossless compression

- Coding

  In this step a lossless compression scheme is applied to the quantized bitstream.

Lossy compression produces superior results, because quantization lowers the amount of information in the signal. The crux of successful compression hereby is to leave the essential information in the bitstream in a compact form. This step is aided by the transform which redistributes the energy of the original signal.

A tricky question is, what part of the information is important and what criteria should be applied to measure importance. Successful transformation schemes in the area of audiovisual signal processing take the human sensory system into account. For example, the ubiquitous MPEG1-Audio-Layer-III audio compression standard makes use of masking effects in human acoustic perception. The wavelet transform (WT), which is our main topic of interest here, used for image compression, takes into account the fact that most natural images consist of low frequential components to a major part. The WT takes a "multiresolution viewpoint" of image data which corresponds to the the way images are perceived by the human visual system. Furthermore, it provides a perspective of the data that is self-similar to a high degree and it is thus predestined for efficient compression, as we will see.

1.2 Motivation for Adaptive Objectbased Image Compression

There are different areas where objectbased compression may prove to be of advantage. We are mainly concerned with the following motivations.

Motivation 1 One obvious application is in objectbased environments like MPEG-4. In such a framework, it makes sense to deal with objects directly, rather than using an indirect approach like a bounding-box technique to compress and store the objects. The overhead introduced by the increased functionality can be balanced by using adaptivity for increased compression efficiency.

Motivation 2 On the other hand, adaptive objectbased encoding of images can lead to better compression results in contexts that are not innately objectbased. The idea here is to segment an image into objects of different characteristics and then adaptively encode each object separately, thus increasing compression performance.

We will propose approaches for both of these areas of application and compare them to other approaches which do not use adaptive objectbased techniques.

1.3 Outline

In chapter 2 we deal with wavelet methods in general. We will present the classical, pyramidal wavelet transformation and then go on to the wavelet packet transformation (WPT). Adaptivity is an important feature of WPT that we will also introduce in this chapter, followed by a set of test results comparing the pyramidal decompositions to WPT-based schemes. In chapter 3 we will discuss how object shapes can be efficiently represented and stored for image coding. The shape adaptive wavelet packet transformation (SA-WPT) is discussed in chapter 4. We will develop SA-WPT from the one-dimensional case to the two-dimensional case, and discuss various issues that apply to either case, especially border extensions for different types of filters as well as subsampling techniques. Results related to *Motivation 1* will also be presented in this chapter. In chapter 5 we introduce the concept of zerotree coding (ZTC) and present a couple of approaches that use it. We will then present an extension for objectbased use of ZTC, and discuss the different possible parameters and their impact on coding efficiency. Chapter 6 deals with *Motivation 2*, the framework we developed for this purpose and the related results. Finally, chapter 7 contains the conclusion.

Test results will be given where applicable and related to the current topic. Note, however, that some of the test results in the earlier chapters may have been produced using methods from later chapters. Especially for zerotree-coding, which is presented in full detail only as late as chapter 5, this is the case.

1.4 Notation And Measures Used

1.4.1 Notation

For easy reading, we use different fonts to denote various structures, where ambiguities are eliminated by the context.

- Two-dimensional arrays $I(i,j)$ are denoted with a single character $\mathbf{I}$ in bold font.
- The structure consisting of a two-dimensional array $O(i,j)$ and a bitmask $M(i,j)$ of the same dimension will also be denoted with a single character $\mathbf{O}$.
- Reconstructed signals are denoted with a hat, e.g. $\hat{\mathbf{I}}, \hat{\mathbf{O}}$.
- Sets are denoted in calligraphic font, e.g. $\mathcal{H}$.

We will refer to an equation by citing its number in brackets, e.g. (2.4).

1.4.2 Measures

As a measure for the quality of a reconstructed image we use peak signal to noise ratio (PSNR).

This measure is based on the mean square error

$$\mathrm{MSE}(\mathbf{I}, \hat{\mathbf{I}}) = \frac{\| \mathbf{I} - \hat{\mathbf{I}} \|}{nm} = \frac{1}{nm} \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} (I(i,j) - \hat{I}(i,j))^2 \tag{1.1}$$

where $\mathbf{I}$ is the original image with width n and height m and $\hat{\mathbf{I}}$ is the reconstructed image. For an object of arbitrary shape, we define the MSE as

$$\mathrm{MSE}(\mathbf{O}, \hat{\mathbf{O}}) = \frac{\| \mathbf{O} - \hat{\mathbf{O}} \|}{p} = \frac{1}{p} \sum_{(i,j) \in \mathbf{M}} (O(i,j) - \hat{O}(i,j))^2 \tag{1.2}$$

where p is the number of pixels in the object and $\mathbf{M}$ is the bitmask defining the object shape. We define PSNR as

$$\mathrm{PSNR}(\mathbf{I}, \hat{\mathbf{I}}) = 20 \log_{10} \frac{255^2}{\mathrm{MSE}(\mathbf{I}, \hat{\mathbf{I}})} \text{ dB}. \tag{1.3}$$

1.5 Remark on testruns

Most of the testresults were produced using the *Ganesh++*-wavelet-packet codec that was developed at the University of Salzburg. All the testimages are 512 × 512 pixels portable grey-map (pgm) images at 8 bpp. Where not stated otherwise, the testruns use level 5 wavelet decompositions. We use the *biorthogonal 7,9* filter for our standard testruns, which is also the default filter for JPEG-2000, but we will also investigate the performance of other filters for selected test images, especially *biorthogonal 3,9*, which is the default filter for Visual Texture Coding module (VTC) of MPEG-4. Where not stated otherwise, global even-odd subsampling (see sec. 4.3.2) and zerotree-coding (see chapter 5) are used, and for objectbased testruns, the cost for encoding the bitmask for each object is excluded from the results. The standard method for objectbased testruns using more than one object is MPT (see sec. 6.3.1.2). For example, a testrun marked as "Objectbased:NORM" is produced using objectbased adaptivity with costfunction ℓ-Norm, SA-WPT with filter *biorthogonal 7,9* and global:even-odd subsampling followed by ZTC using SMAWZ (with MPT in the case of more than one object) .

Chapter 2

Image Transformation with Wavelet Methods

2.1 Introduction

Image and video coding with wavelet methods have been paid much attention in the scientific community in recent years. Since the finalization of the JPEG2000 standard [1, 2] at the latest, wavelet methods are now well established in image compression. The MPEG-4 standard [3] makes use of the wavelet transform in its visual texture coding (VTC) module [4]. Especially for low bitrates, wavelet methods outperform longer established techniques like the discrete cosine transform (DCT) [5, 6].

Wavelet methods for image and video coding, maybe even more than other methods, comprise a large field of research, ranging from the actual transform itself via closely related topics like zerotree coding to highly specialized applications like medical image coding. The wavelet transform itself is not a single algorithm, but a whole set of different approaches, ranging from the classical pyramidal Mallat decomposition to adaptive transformations, like the wavelet packet transform (WPT) using best basis algorithms, each with its area of application. The WPT, for example, has been successfully adopted for the compression of oscillatory signals like digital fingerprints [7].

This chapter gives a basic introduction to wavelet methods in general and presents the Mallat and WP decompositions in more detail. The methods of chapter 4 use these principles for developing actual algorithms for the shape adaptive wavelet transform, which is a generalization of the rectangular wavelet transform. In this framework, the introduction will only provide the most essential foundations. For more thorough introductions to the topic that also cover the underlying mathematical principles in much more detail see [8, 9, 10, 11, 12, 13].

2.2 The Wavelet Transform (WT)

Although digital image compression deals with discrete signals, the continuous approach is the pool of ideas where a lot of the concepts for discrete transformations come from.

We will develop the basics of the continuous wavelet transform (CWT) and then show the transition to the discrete wavelet transform (DWT).

2.2.1 Definitions

Let $L_2(\mathbb{R})$ denote the space of all square integrable functions f

$$\int_{-\infty}^{\infty} |f(x)|^2 dx < \infty \tag{2.1}$$

The *inner product* of functions f, g in $L_2(\mathbb{R})$ is defined as

$$\langle f, g \rangle = \int_{-\infty}^{\infty} f(x)\overline{g(x)}dx. \tag{2.2}$$

The *norm* in $L_2(\mathbb{R})$ is defined as

$$\|f\|_2 = \langle f, f \rangle^{\frac{1}{2}}. \tag{2.3}$$

The *Kronecker symbol* is defined on $\mathbb{Z} \times \mathbb{Z}$ as

$$\delta_{j,k} = \begin{cases} 1 & \text{for } j = k \\ 0 & \text{for } j \neq k \end{cases} \tag{2.4}$$

A *biorthogonal* basis in $L_2(\mathbb{R})$ is a basis $\{f_i\}$ for which a dual basis $\{g_i\}$ exists such that

$$\langle f_i, g_j \rangle = \delta_{i,j}. \tag{2.5}$$

An *orthogonal* basis is a biorthogonal basis that is its own dual.

2.2.2 Continuous Wavelet Transform (CWT)

Let f be a function in the function space of square integrable functions, $f \in L_2(\mathbb{R})$. The CWT is essentially a unitary map (up to a factor C_ψ [13]) on $L_2(\mathbb{R}) \to L_2(\mathbb{R})$.

A unitary transform $\mathcal{T}$ preserves dot products

$$\langle \mathcal{T}f, \mathcal{T}g \rangle = \langle f, g \rangle. \tag{2.6}$$

In consequence, the norm of a function is preserved

$$\|\mathcal{T}f\|_2 = \|f\|_2. \tag{2.7}$$

Intuitively, a unitary transformation just alters the perspective in which a function is observed. The advantage for image coding is that for images certain unitary transforms provide a perspective that compacts the energy in a small portion of coefficients.

The idea of the CWT is to describe any $f \in L_2(\mathbb{R})$ as the combination of different basis functions ψ_{ab}. $\{\psi_{ab}\}$ are all dilated and translated versions of one function ψ with

compact support, often called the *mother wavelet*. a is the *dilatation* parameter and b is the *translation* parameter.

The only requirements we formulate for ψ so far are that

$$\int \psi(t)dt = 0 \tag{2.8}$$

and that ψ has mean zero. ψ_{ab} is formed from ψ by

$$\psi_{ab}(x) = |a|^{-\frac{1}{2}}\psi(\frac{x-b}{a}). \tag{2.9}$$

Note that if ψ has unit length, then also all ψ_{ab} have unit length:

$$\|\psi\|_2 = 1 = \|\psi_{ab}\|_2, \ a, b \in \mathbb{R}. \tag{2.10}$$

We can write $f^*(a,b)$ as follows

$$\begin{aligned} f^*(a,b) &= \langle f, \psi_{ab}\rangle & (2.11)\\ &= \int f(x)\psi_{ab}(x)dx & (2.12)\\ &= |a|^{-\frac{1}{2}} \int f(x)\psi(\frac{x-b}{a})dx. & (2.13) \end{aligned}$$

For the CWT, a and b are varied continuously over $\mathbb{R}$ to obtain a representation of $f(x)$ in the wavelet domain. The important fact in our brief discussion of the CWT is that it constitutes a (unitary) mapping of $f(x) \rightarrow f^*(a,b)$.

Of course, transforming a one-variable function into a two-variable function is not useful for compression. In fact, the CWT itself is not suited for compression at all, but is very useful in the area of signal analysis. The wavelet transform becomes interesting for image compression in its discrete form, which uses the idea of the CWT, but works in a rather different way, as will be seen.

2.2.3 Discrete Wavelet Transform (DWT)

For processing digital images the discrete wavelet decomposition is of interest. In the DWT, the dilatation and translation parameters are limited to discrete values. Further, by setting dilation to 2^j (*binary partitions*) and translation to $\frac{k}{2^j}$, the original signal is partitioned into subbands of different frequency ranges (note that this is only one of many possible discretization of the CWT). $\psi_{jk}(x)$ is thus defined as

$$\psi_{jk}(x) = 2^{-\frac{j}{2}}\psi(2^j x - k), \ j, k \in \mathbb{Z}. \tag{2.14}$$

As ψ stays the same, we can state that for the DWT the CWT is evaluated at the dyadic positions $b = k/j^2$ and binary dilations $a = 2^{-j}$ [9]. Analogous, the wavelet coefficients c_k^j are produced by

$$c_k^j = \langle f, \psi_{jk}\rangle.$$

Thus, every wavelet ψ constructs a wavelet series for a given function $f \in L_2(\mathbb{R})$,

$$f(x) = \sum_{j,k=-\infty}^{\infty} c_k^j \psi_{jk}(x). \tag{2.15}$$

As this is true for every $f \in L_2(\mathbb{R})$, $\{\psi_{jk},\ \forall j,k \in \mathbb{Z}\}$ span $L_2(\mathbb{R})$. Now consider leaving j fixed and only varying k over $\mathbb{Z}$. We will denote the space that is spanned by $\{\psi_{jk}, \forall k \in \mathbb{Z}\}$ with $j \in Z$ fixed as W_j. It is obvious that by forming the direct sum of all W_j, $\forall j \in \mathbb{Z}$ we again obtain $L_2(\mathbb{R})$. Thus, every function $f \in L_2(\mathbb{R})$ can be decomposed uniquely as:

$$f(x) = \cdots + g_{-1}(x) + g_0(x) + g_1(x) + \cdots \tag{2.16}$$

where $g_j \in W_j, \forall j \in \mathbb{Z}$. $W_j, j \in \mathbb{Z}$ form a partitioning of $L_2(\mathbb{R})$

$$L_2(\mathbb{R}) = \bigoplus_j W_j. \tag{2.17}$$

As g_j lives in W_j, there must be a wavelet series representation for g_j in W_j (which is spanned by ψ_{jk}). Consider that by taking all subspaces W_k for $k < j$, we get a series of nested subspaces V_j

$$V_j = \bigoplus_{k<j} W_k. \tag{2.18}$$

This introduces the concept of multiresolution analysis (MRA).

2.2.4 Multiresolution Analysis (MRA)

We can deduce the following properties for this series of subspaces $V_j, j \in \mathbb{Z}$ [9, 13]:

(1) $V_j \subset V_{j+1},\ \forall j \in \mathbb{Z}$

(2) $\bigcup_{j\in\mathbb{Z}} V_j = L_2(\mathbb{R})$

(3) $\bigcap_{j\in\mathbb{Z}} V_j = 0$

(4) $V_{j+1} = V_j \oplus W_j,\ \forall j \in \mathbb{Z}$

(5) $f(t) \in V_j \iff f(2t) \in V_{j+1},\ \forall j \in \mathbb{Z}$

We now introduce a so-called *scaling function* $\phi \in V_0$ which generates the V_j series of subspaces. For this purpose, take

$$\phi_{jk}(t) = 2^{j/2}\phi(2^j t - k) \tag{2.19}$$

with ϕ_{0k} being an orthonormal basis for V_0 and ϕ_{lk} being an orthonormal basis for V_l. From $V_0 \subset V_1$ follows that also $\phi \in V_1$. Thus, as ϕ_{1k} is a basis of V_1,

$$\phi(t) = \sum_k h(k)\phi_{1k}(t) \tag{2.20}$$

$$\overset{(2.19)}{=} \sum_k h(k)\sqrt{2}\phi(2t - k) \tag{2.21}$$

for some coefficients $h(k)$. As ϕ_{1k} is an orthonormal basis of V_1

$$h(k) = \langle \phi_{1k}, \phi \rangle \qquad \text{and} \tag{2.22}$$

$$\sum_k |h(k)|^2 = 1 \qquad \text{(normality)}. \tag{2.23}$$

Further, we can represent ϕ_{0k} as a sum of translated versions of ϕ_{1k} multiplied by coefficients $h(k)$

$$\phi_{0k} \overset{(2.19)}{=} \phi(t-k) \tag{2.24}$$

$$\overset{(2.20)}{=} \sum_k h(k)\phi_{1k}(t-k). \tag{2.25}$$

Now that we have defined the *scaling function* ϕ, we construct a suited *wavelet* ψ to span the wavelet spaces W complementary to V. We generate ψ from ϕ_{1k} in a way similar to (2.20)

$$\psi(t) = \sum_k g(k)\phi_{1k}(t) \tag{2.26}$$

where the coefficients $g(k)$ are derived from $h(k)$

$$g(k) = (-1)^k h(-k+1). \tag{2.27}$$

Analogous to (2.22), $g(k)$ can be written as

$$g(k) = \langle \phi_{1k}, \psi \rangle \tag{2.28}$$

As ϕ_{0k} is a basis of V_0, a function $f \in V_0$ can be represented as:

$$f = \sum_k c_k^0 \phi_{0k}. \tag{2.29}$$

Further, it is possible to write the following decomposition f

$$f = \gamma^{-1} + \delta^{-1} \in V_{-1} \oplus W_{-1}. \tag{2.30}$$

(2.30) can be applied repeatedly to the resulting γ and the original function f is still perfectly recoverable:

$$f = \gamma^{-k} + \delta^{-k} + \delta^{-k+1} + \cdots + \delta_{-1} \in V_{-k} \oplus W_{-k} \oplus \cdots \oplus W_{-1} \tag{2.31}$$

where γ^{-k} is the *approximation* signal and $\{\delta^{-k}, \ldots, \delta^{-1}\}$ are the *detail* signals.

Without loss of generality, we show the development of γ^{-1} and δ^{-1} [13].

$$\gamma^{-1} = \sum_k \langle \phi_{-1k}, f \rangle \phi_{-1k} \tag{2.32}$$

$$\overset{(2.29)}{=} \sum_{kl} c_l^0 \langle \phi_{-1k}, \phi_{0l} \rangle \phi_{-1k} \tag{2.33}$$

$$\overset{(2.19)}{=} \sum_{kl} c_l^0 \phi_{-1k} \int 2^{-\frac{1}{2}} \phi(2^{-1}t - k)\phi(t - l)dt \tag{2.34}$$

$$\overset{s:=2^{-1}t-k}{=} \sum_{kl} c_l^0 \phi_{-1k} \int \phi(s)\sqrt{2}\phi(2s - (l - 2k))ds \tag{2.35}$$

$$\overset{\sqrt{2}\phi(2s-(l-2k)) = \phi_{1(l-2k)}}{=} \sum_{kl} c_l^0 \phi_{-1k} \langle \phi_{1(l-2k)}, \phi \rangle \tag{2.36}$$

$$\overset{(2.22)}{=} \sum_{kl} c_l^0 h(l - 2k)\phi_{-1k} \tag{2.37}$$

$$=: \sum_k c_k^1 \phi_{-1k} \tag{2.38}$$

For δ^{-1} the development is similar:

$$\delta^{-1} = \sum_k \langle \psi_{-1k}, f \rangle \psi_{-1k} \tag{2.39}$$

$$\overset{(2.29)}{=} \sum_{kl} c_l^0 \langle \psi_{-1k}, \phi_{0l} \rangle \psi{-}1k \tag{2.40}$$

$$\overset{(2.14),(2.19)}{=} \sum_{kl} c_l^0 \psi_{-1k} \int 2^{-\frac{1}{2}} \psi(2^{-1}t - k)\phi(t - l)dt \tag{2.41}$$

$$\overset{s:=2^{-1}t-k}{=} \sum_{kl} c_l^0 \psi_{-1k} \int \psi(s)\sqrt{2}\phi(2s - (l - 2k))ds \tag{2.42}$$

$$\overset{\sqrt{2}\phi(2s-(l-2k)) = \phi_{1(l-2k)}}{=} \sum_{kl} c_l^0 \psi_{-1k} \langle \phi_{1(l-2k)}, \psi \rangle \tag{2.43}$$

$$\overset{(2.28)}{=} \sum_{kl} c_l^0 g(l - 2k)\psi_{-1k} \tag{2.44}$$

$$=: \sum_k d_k^1 \psi_{-1k} \tag{2.45}$$

2.2.5 Filters

So far, we have only dealt with functions in $L_2(\mathbb{R})$, whereas with digital image processing we usually work in the sequence space $\ell_2(\mathbb{R})$. However, there is a mapping from $L_2(\mathbb{R})$ to $\ell_2(\mathbb{R})$ for this purpose [13]. For an orthonormal basis $\{e_l(t)\}$ in $L_2(\mathbb{R})$, a sequence $\{(c_l^0)\}$

of coefficients can be mapped to the function that is defined uniquely by $\{(c_l^0)\}$.

$$\{(c_l^0)\} \to f = \sum_l c_l^0 e_l(t) \tag{2.46}$$

Further, as described above, we can map the sequence $\{(c_l^0)\}$ to its decomposition $\{(c_k^1), (d_k^1)\}$

$$\{(c_l^0)\} \to \{(c_k^1), (d_k^1)\} \overset{(2.38),(2.45)}{=} \{(\sum_l h(l-2k)c_l^0), (\sum_l g(l-2k)c_l^0)\} \tag{2.47}$$

Convolution in $\ell_2(\mathbb{R})$ between a filter $\{f(j),\ j = l, \ldots, L\}$ and a sequence $\{x(k)\}$ producing an output signal $y(k)$ is defined as

$$y(k) = (f * x)(k) = \sum_{i=l}^{L} f(i)x(k+i). \tag{2.48}$$

(2.37) and (2.44) can be interpreted as convolutions (followed by subsampling) in the digital domain. Thus, the above equations form the transition of the integral wavelet transform to digital image processing. In this interpretation, $\{h(k)\}$ serves as a lowpass filter and $\{g(k)\}$ serves as a highpass filter.

Also, a subsampling process is performed. If the original signal has N coefficients then the approximation and detail signal of one decomposition together have N coefficients as well. For applying the transformation to finite sequences, the input signal has to be extended in a suitable way. We will enlarge on these points when we discuss the shape adaptive wavelet transformation in chapter 4.

2.3 2D Pyramidal Wavelet Transform

The wavelet transform is separable, i.e. for applying it in higher dimensions, the wavelet transform can be used in each dimension separately. For transforming digital images, the row vectors can be transformed first (without loss of generality) followed by a transformation applied to the column vectors of the resulting coefficients. The result is a level one wavelet decomposition consisting of four subbands – one approximation subband and three detail subbands.

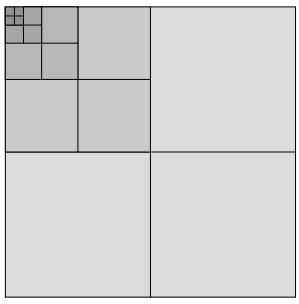


Figure 2.1: Pyramidal wavelet decomposition, level 5

For the pyramidal wavelet transformation, only the approximation subband is decomposed further. Thus, the pyramidal transform is unique in that it creates the same structure for every image for a certain level of decomposition (fig. 2.1).

The advantage is that for most natural images, a large portion of the energy is contained in the approximation subbands. By decomposing the approximation subband further, the energy can be concentrated into fewer coefficients, a process which is called *energy compaction*. As will be seen, this representation is well suited for various methods of entropy coding. Each subband corresponds to coefficients

of a certain scale and orientation. From property (4) above follows that the structures in the subband is similar across the scales. This *self-similarity* is exploited for *zerotree coding*, which will be discussed in chapter 5.

For some images, especially images containing oscillatory patterns, the pyramidal wavelet decomposition fails to concentrate the energy in just a few coefficients. Instead, the energy is spread over the detail subbands. In such cases it makes sense to also decompose the detail subbands further, leading us to the *wavelet packet transform* (WPT).

2.4 Wavelet Packet Transform (WPT)

The mathematical foundations developed above can of course be extended to WPT. The discussion here will be limited to a subband and signal processing point of view. The mathematical details can be found in [10].

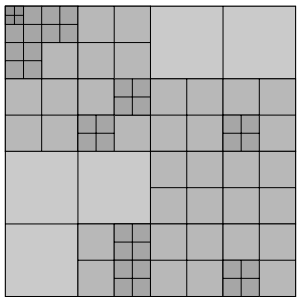


Figure 2.2: A possible WPT structure, level 5

The wavelet packet transform is a generalization of the pyramidal wavelet decomposition, where the approximation subband is not the only subband to be decomposed further (fig. 2.2). Thus, for a specific level of maximum decomposition depth, there are many possible WP-structures – the WPT is an *overcomplete* library of bases. Each structure represents its own wavelet basis and therefore a specific frequency subband decomposition. These decompositions can be adapted to the properties of the image to be transformed and to take high frequential components into account. Thus, the WPT has been successfully used for compressing images with oscillatory patterns, like the famous testimage "*Barbara*" [14, 15, 16] or fingerprint images [7].

A tree representation is well suited to describe a single decomposition. The *"decomposition tree"* is a quadtree whose nodes represent the binary decisions whether the subband associated with this node is decomposed further or not. Note that it makes sense to distinguish the tree representing the decomposition structure from the actual coefficients. In our implementation the coefficients are stored separately from the decomposition information. We will generically call the structure holding the coefficients *"coefficient tree"* (although the actual implementation does not necessarily have to be a tree). When we discuss zerotree coding for WPT, we will introduce another tree, the *"similarity tree"*, which orders the subbands in the decomposition tree for zerotree coding.

2.5 Adaptive Transformation

As the bases of the WPT are overcomplete, a way has to be found to choose the best suited basis for each image. The best basis algorithm (BBA) is the approach used in our testruns. It optimizes additive information cost functions to achieve this purpose. From a rate-distortion point of view there are superior methods [17], yet the BBA in conjunction with additive costfunction has low computation complexity.

2.5.1 Best Basis Algorithm (BBA)

In the BBA [10], a full wavelet packet decomposition for the target level is generated. The BBA uses costfunctions to decide where the decomposition structure should be modified. Various costfunctions can potentially be used – we limit our testruns to additive costfunctions. Additive costfunction have the advantage that for a set of subbands $\mathbf{S}_1, \ldots, \mathbf{S}_n$ and a costfunction f

$$\sum_{i=1}^{n} f(\mathbf{S}_i) = f(\bigcup_{i=1}^{n} \mathbf{S}_i) \tag{2.49}$$

holds.

The following additive costfunctions are used in our tests

- Logarithm of Energy (LogE)

$$cost_{LogE}(\mathbf{S}) = \sum_{(u,v)\in\mathcal{I}_w\times\mathcal{I}_h} \log^*(t) \qquad |_{t=S(u,v)^2} \tag{2.50}$$

- ℓ-Norm (Norml)

$$cost_{Norm}(\mathbf{S}) = \sum_{(u,v)\in\mathcal{I}_w\times\mathcal{I}_h} |S(u,v)| \tag{2.51}$$

- Entropy Information Cost (EIC)

$$cost_{EIC}(\mathbf{S}) = -\sum_{(u,v)\in\mathcal{I}_w\times\mathcal{I}_h} t * \log^*(t) \qquad |_{t=S(u,v)^2} \tag{2.52}$$

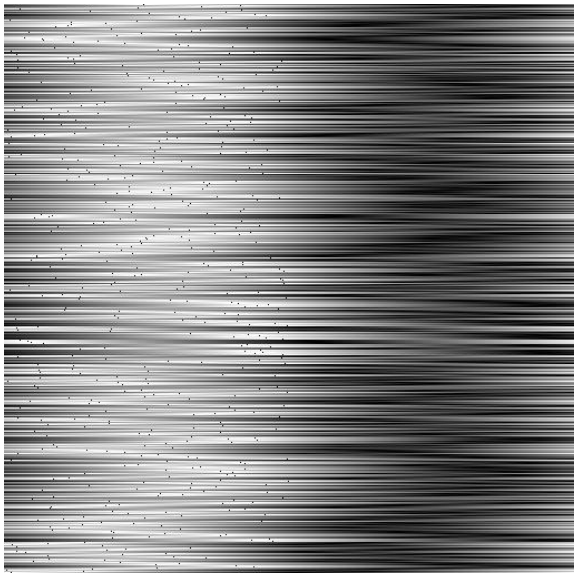
where $\mathbf{S}$ is a subband of width w and height h with row-indices defined by $\mathcal{I}_h$ and column-indices defined by $\mathcal{I}_w$, and

$$\log^*(t) = \begin{cases} \log(t) & \text{for } t \neq 0 \\ 0 & \text{else.} \end{cases} \tag{2.53}$$

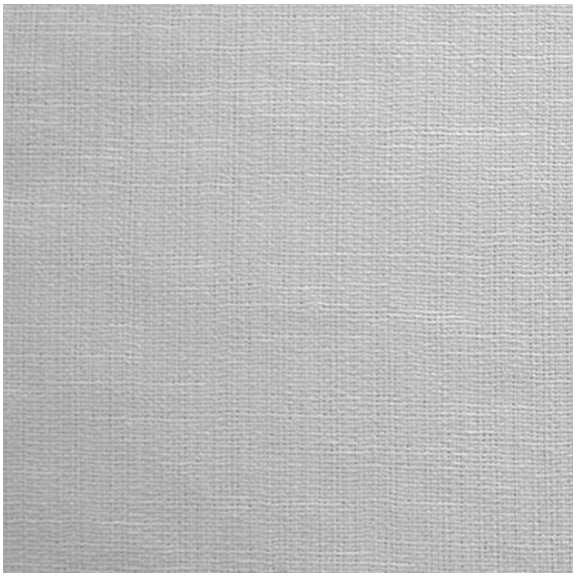
In the perspective of *decomposition trees*, the BBA does nothing else than working its way upwards recursively in the decomposition tree, comparing additive cost information for each subband in its undecomposed state with the four subbands resulting from decomposing and pruning the tree wherever the cost for decomposition is higher than for leaving the subband undecomposed. In that way, the BBA always produces an optimal basis with regard to the used measure.

2.6 Results

As the WP decomposition is known to work well with oscillatory patterns, we chose a set of testimages containing textures with particular high frequency components. This is taken to the extreme for the artificially generated texture of fig. 2.3.a, where high frequency patterns only run in one direction. Other test images are comprised of Brodatz textures [18] and images of textile fabric. In the following, we explore the performance of the BBA using different additive costfunctions, namely *LogE*, *Norm* and *EIC*.

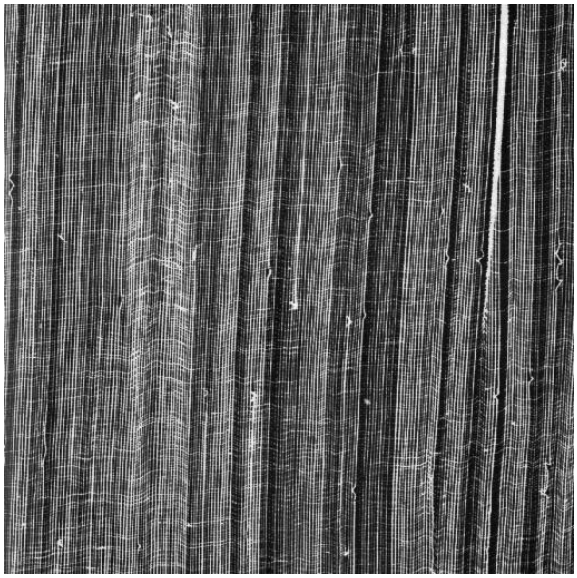


(a) Artificial test texture *artificial*

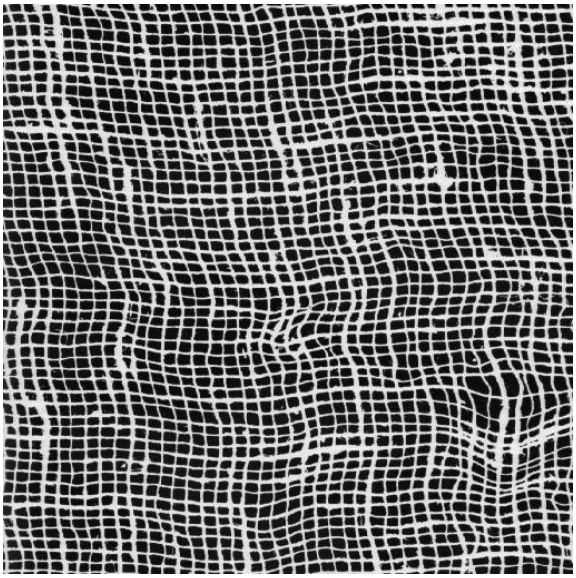


(b) Fabric texture *poly_ linen*

Figure 2.3: Test images for WP decomposition I



(a) Brodatz texture *D105*



(b) Brodatz texture *D103*

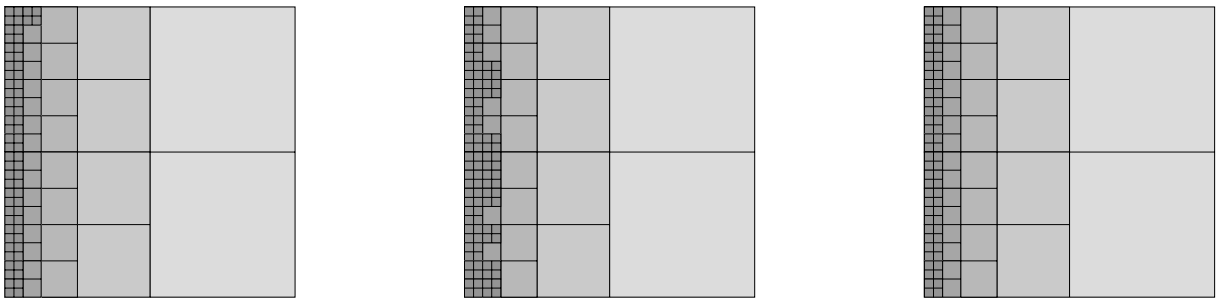
Figure 2.4: Test images for WP decomposition II

2.6.1 Costfunctions

Fig. 2.6.a shows that for the artificial texture, the pyramidal wavelet transform is outperformed by the WP-transform over the whole bitbudget considered, regardless of the used costfunction. However, as each costfunction represents a different measure, the decomposition structure varies for each of the costfunctions (fig. 2.5). For the fabric image *poly_linen.pgm* and the Brodatz texture *D105.pgm*, which both show traits in their frequential patterns similar to the artificial test image, the pyramidal wavelet transform is also outperformed for all costfunctions (fig. 2.6.b, 2.7.a).

For textures that do not contain oscillatory patterns to such an extent, the performance of the WP-decomposition is not that superior. As can be seen in fig. 2.7.b, the WP-decomposition using the *LogE*-costfunction is even outperformed by the pyramidal decomposition for the lowest bitrate considered.

Comparing the performance of the various costfunctions, we observe that, as each of the costfunction addresses a different measure, there is not just one single costfunction that guarantees to always produce the best decomposition with regard to PSNR-performance.



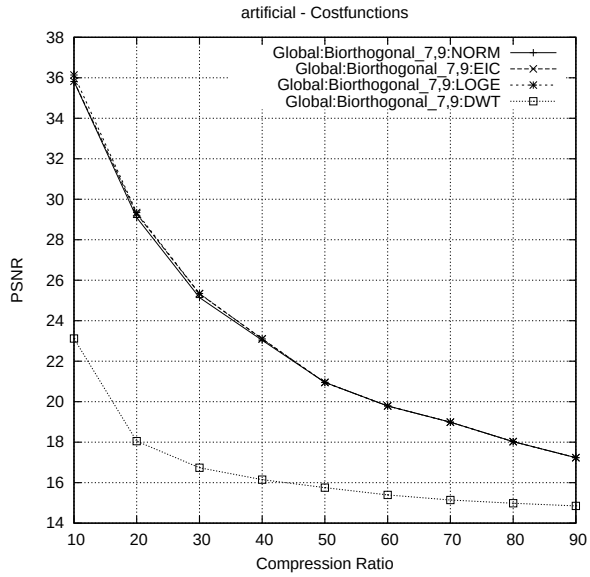
(a) Norm (b) EIC (c) LogE

Figure 2.5: Decomposition structures for fig. 2.3.a

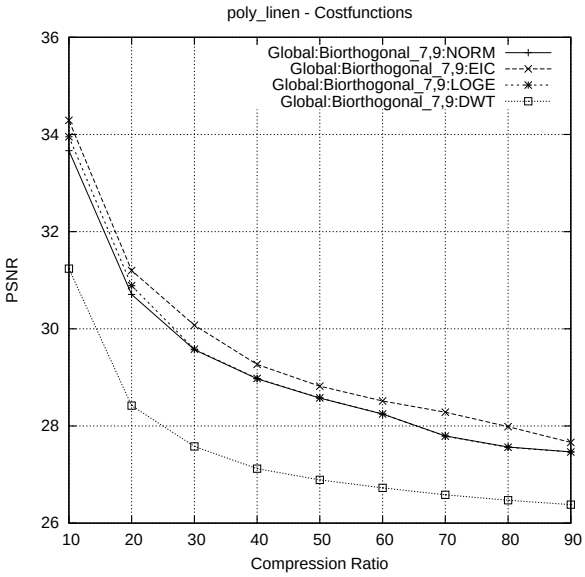
2.6.2 Filters

Fig. 2.9 shows results comparing different filters. The 12-tap orthogonal *symlet* filter is known to be well suited for compressing high frequential patterns. Further we test the 6-tap orthogonal filter by Coifman, and another filter from the family of biorthogonal filters, *biorthogonal 3,9*, which is the standard filter for MPEG-4 VTC. Note that as biorthogonal filters have less restrictive requirements as regards orthogonality, the BBA in this case should rather be termed *Near Best Basis Algorithm* (nBBA).

Biorthogonal 3,9 does not produce competitive results for our oscillatory test images and is outperformed by the other filters as can be well seen in fig. 2.9.a. The results for all of the other filters range very closely together, with *symlet 12* often gaining on higher compression ratios, and *biorthogonal 7,9* performing better for lower compression ratios.

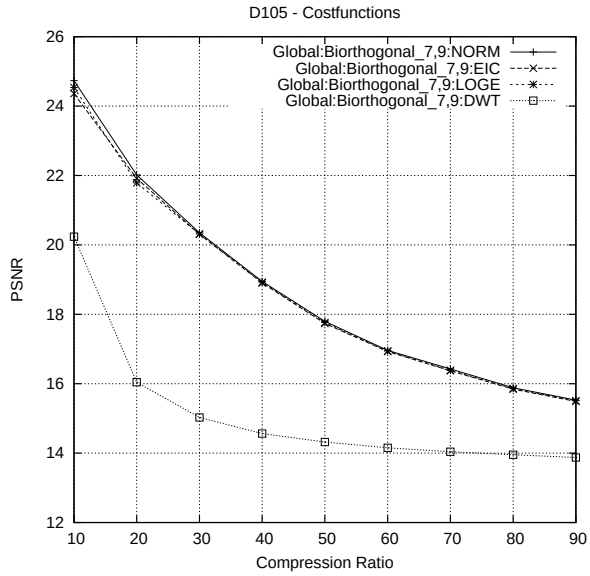


(a) Results for fig. 2.3.a and filter Biorthogonal 7,9

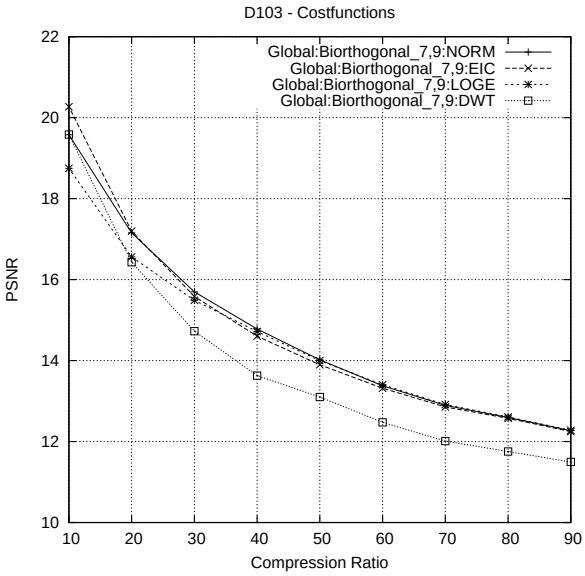


(b) Results for fig. 2.3.b and filter Biorthogonal 7,9

Figure 2.6: Comparison of costfunctions for fig. 2.3



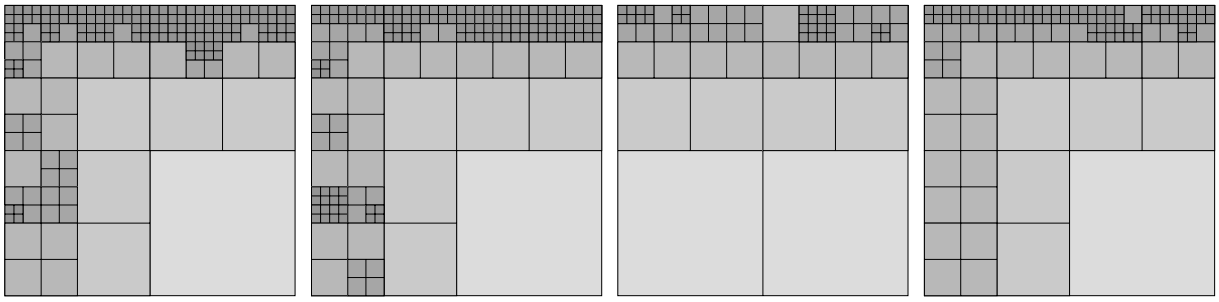
(a) Results for fig. 2.4.a and filter Biorthogonal 7,9



(b) Results for fig. 2.4.b and filter biorthogonal 7,9

Figure 2.7: Comparison of costfunctions for fig. 2.4

The performance of the costfunction does not vary by the used filter, i.e. the same costfunctions that are best for an image for one filter also achieve the best results for other tested filters. However, as the used filters influence the cost for decomposition, the wavelet packet structures produced for different filters by the same costfunction also differ to a certain degree. Fig. 2.8 shows this for the four different filters applied to test-image *D105.pgm* using ℓ-Norm as costfunction.



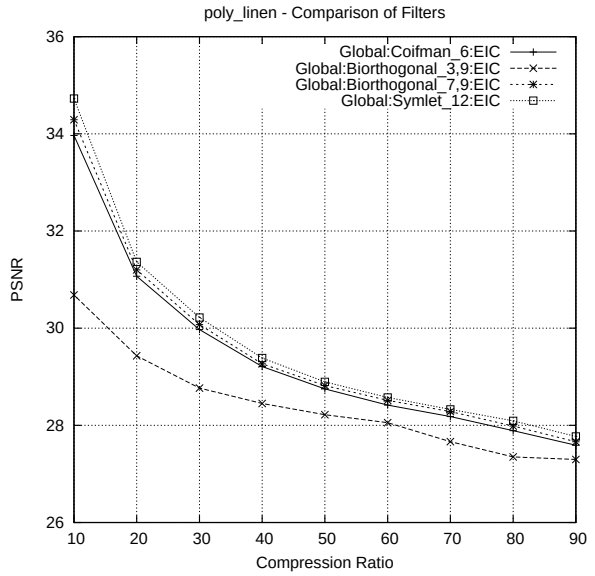
(a) Coifman 6 (b) Symlet 12 (c) Biorthogonal 3,9 (d) Biorthogonal 7,9

Figure 2.8: Decomposition structures for fig. 2.4.a and different filters using costfunction ℓ-Norm

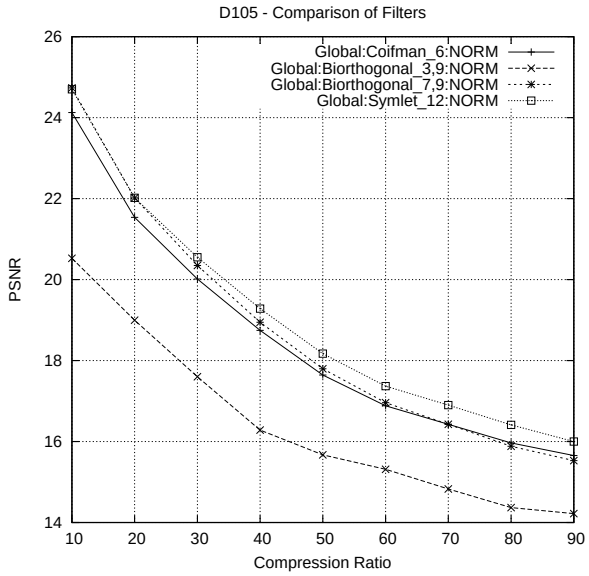
2.6.3 Comparison to JPEG-2000

Fig. 2.10 shows results comparing the performance of WPT using the *Ganesh++*-codec to JASPER, which is a reference implementation of the codec specified in the JPEG-2000 Part-1 standard (ISO/IEC 15444-1). We used the standard *biorthogonal 7,9* filter for both codecs to achieve better comparability. In the result-plots, only the most successful costfunction is shown for WPT in *Ganesh++*.

As can be seen, JPEG-2000 is outperformed over the whole bitrange. This is mainly due to the oscillatory characteristics in frequency space of the tested images, to which JPEG-2000 is unable to adapt. However, JASPER does outperform the pyramidal transform of *Ganesh++*.

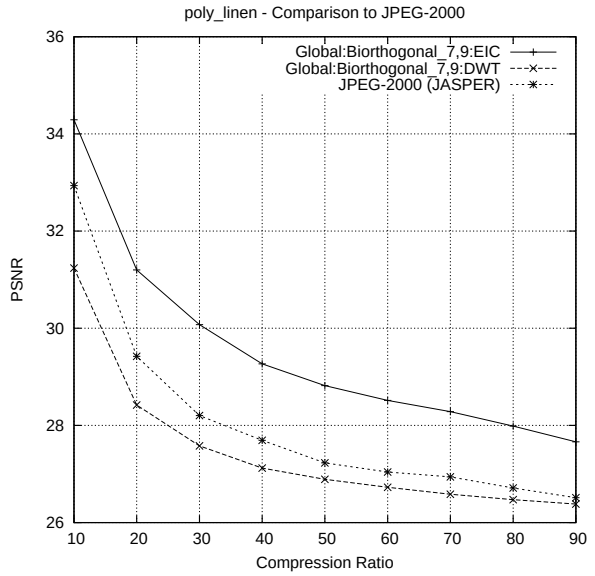


(a) Results for fig. 2.3.b

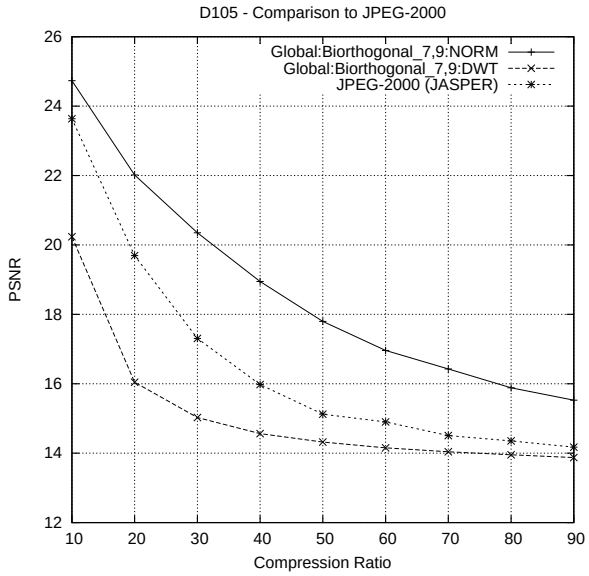


(b) Results for fig. 2.4.a

Figure 2.9: Comparison of filters



(a) Results for fig. 2.3.b



(b) Results for fig. 2.4.a

Figure 2.10: Comparison to JPEG-2000

Chapter 3

Object Shape Representation

3.1 Introduction

The emergence of the objectbased video coding standard MPEG-4 is only one example that uses the advantages of objectbased coding (OBC). Especially, with the object-based wavelet transformation [19, 20] it is possible to get an efficient representation of objects in the frequency domain (see chapter 4). The gains in functionality with OBC are manifold. Region of interest (ROI) [21] coding makes it possible to differentiate between important and unimportant objects, for example, foreground and background objects. Further, objectbased codecs [22, 23, 24] exist that are tailored to optimize coding performance for objects.

However, in order for objects to be transformed in an efficient way, a good representation for object shapes has to be available. This chapter deals with the question how to represent arbitrarily shaped image objects in an efficient way. A good and efficient representation is crucial for compression efficiency and functionality in an objectbased environment. For an overview of the topic see [25].

Many different methods for object shape coding exist. As with coding of data in general, we distinguish between *lossless* and *lossy* shape coding. With lossy shape coding, perfect reconstruction of the original shape is not possible, but the representation is more compact. We will only discuss some lossless representations of object shapes, with a bias towards bitmap-based methods.

We use the following definition for the shape **S** of an object

$$S(x,y) = \begin{cases} 1 & \text{if position } (x,y) \text{ is part of the object} \\ 0 & \text{else.} \end{cases} \tag{3.1}$$

3.2 Contour-based Representation

In contour-based representation, the contours of the object are used to represent the object shape. Intuitively, a contour is nothing else than a digital line around the object. For each pixel in a digital line, there is only a limited number of possible directions in which the line can continue. We distinguish between 4-connectivity and 8-connectivity.

- **4-connectivity** allows the line to continue in only four directions: *N, E, S, W.*
- **8-connectivity** allows the line to continue in eight directions: *N, NE, E, SE, S, SW, W, NW.*

Formally, a contour can be defined by its mask as follows [25]. Let $N(x, y)$ define the neighborhood for position (x, y) with

$$N(x,y) = \begin{cases} \{(u,v) \mid (|u-x| + |v-y|) = 1\} & \text{for 4-connectivity} \\ \{(u,v) \mid max(|u-x|, |v-y|) = 1\} & \text{for 8-connectivity.} \end{cases} \tag{3.2}$$

The contour mask $C(x, y)$ is then defined as

$$C(x,y) = \begin{cases} 1 & \text{if } S(x,y) = 1 \text{ and } \exists (u,v) \in N(x,y) \text{ with } S(u,v) = 0 \\ 0 & \text{else.} \end{cases} \tag{3.3}$$

3.2.1 Freeman-chains

Freeman introduced a method for contour coding in [26] that is still in widespread use. A transition from one pixel to the next on a digital line is called a *link*. In the case of 4-connectivity, a link can be represented by 2 bit, in the case of 8-connectivity it can be represented by 3 bit. A contour chain can be encoded by the sequence of the links that constitute it as follows.

(1) Select an arbitrary starting point that belongs to the contour mask and encode its coordinates

(2) For each point p

- If p is the initially selected point, then stop.
- Else, represent the consecutive point s on the line by a link from p to s and repeat (2) for s.

The above algorithm only works for closed contours. Open contours require minor modifications.

This basic approach has been enhanced over time with several improvements. Freeman himself proposed differential coding of these chains in [27] for efficient subsequent entropy coding, other approaches include probability models or assumptions about the characteristics of the contour (see [25]).

3.2.2 Polygonal Representation

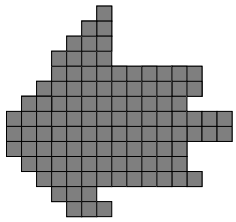
A contour may also be represented by coding all its vertices and connecting them with straight lines. In the worst case, if there are no straight lines, all the positions in the contour line have to be encoded explicitly. This scheme is more useful for lossy coding where the contour is approximated by its key vertices.

3.3 Quadtree Representation

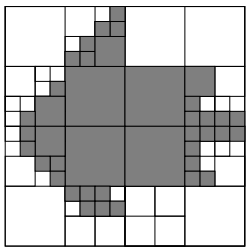
Quadtree structures can be used to represent object shapes. Usually macroblocks of a power of two are used and a quadtree is built on each of these macroblocks. In a quadtree, each node that is not a leaf node has four children. The root of the tree represents the macroblock as a whole. The four children $C_i,\ i = 0, \ldots, 3$ of a node N each represent a quarter of the area that N represents. In order to get a quadtree representation of an object shape, the following algorithm is applied recursively to a node N, starting at the root:

- If the area represented by N is contained in the object shape as a whole, mark it as *"inside"*
- If the area represented by N lies outside the object shape as a whole, mark it as *"outside"*
- If only part of the are represented by N lies inside the object shape, then split N into four child-nodes $C_i,\ i = 0, \ldots, 3$ and apply the algorithm to each of them.

The nodes marked as *"inside"* give a unique representation of the object shape, as illustrated in fig. 3.1. There are many algorithms for efficient coding of quadtrees. The method as such is lossless, if the above algorithm is applied until pixel granularity is reached. By introducing a different break condition, like reaching a certain depth in the tree, the method can be made lossy.



(a) Original shape



(b) Quadtree representation

Figure 3.1: Quadtrees

3.4 Bitmap-based Representations

3.4.1 Context-based Arithmetic Encoding (CAE)

In [28], Langdon and Rissanen first proposed a method to encode binary images using a state machine and arithmetic coding. We will discuss this approach and some enhancements that were proposed later without going into details of arithmetic coding. See e.g. [29] for an introduction to arithmetic coding.

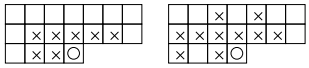


Figure 3.2: Templates for CAE

In CAE a template is given that defines which pixels are counted to the neighborhood of a position. Fig. 3.2 shows 7-pixel and 10-pixel templates defined in the original proposal. The object shape is scanned line by line, and the neighborhood is defined for each pixel. The values of the positions

in the neighborhood are used to determine the state of the finite state machine. A probability distribution is associated with each such state and is used to drive an arithmetic encoder. The coding can be done adaptively, i.e. the probability distributions are updated with each pass. Although the idea is simple, the approach is very efficient and has been adopted for the JBIG standard.

In [30] a scalable variation of the algorithm is presented. A multiresolution decomposition of an object shape $\mathbf{S}$ is created by defining layers $\{\mathbf{L}_0, \ldots, \mathbf{L}_m\}$ on $\mathbf{S}$, where $\mathbf{L}_0$ is equal to $\mathbf{S}$ and the other layers are defined as

$$\begin{aligned} L_{l+1}(i,j) &= L_l(2i,2j) \vee L_l(2i+1,2j) \vee \\ &\quad L_l(2i,2j+1) \vee L_l(2i+1,2j+1). \end{aligned} \tag{3.4}$$

$\mathbf{L}_m$ is called the *base* layer and the other layers are called *enhancement* layers. The enhancement layers are encoded using a template that contains positions of previous layers.

3.4.2 One-Two-Four (OTF)

OTF [31] in its basic idea is similar to the scalable version of the previously presented method. However, the downscaling process is done differently and is fitted to the shape adaptive wavelet transform. OTF uses a blockbased preprocessing with respect to the original shape, and stores information whether all positions in a block are inside the mask, all positions in a block are outside the mask or whether a block is a border block containing both masked and non-masked positions.

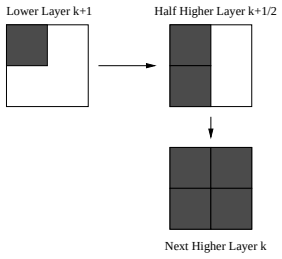


Figure 3.3: Layers in OTF

The object shape is decomposed with the same wavelet-filter that is used for SA-DWT on the actual object into $m+1$ layers $\{\mathbf{L}_0, \ldots, \mathbf{L}_m\}$, where again $\mathbf{L}_0$ corresponds to the object shape in full resolution and $\mathbf{L}_m$ is called the *base* layer. In each layer, only border blocks need to be encoded. The base layer is encoded on its own, using a non-adaptive CAE scheme.

Each of the following layers are encoded using the respective underlying layer. As shown in fig. 3.3 an intermediate layer is used for encoding. The intermediate layer $\mathbf{L}_{l+1/2}$ between $\mathbf{L}_l$ and $\mathbf{L}_{l+1}$ is created by applying the wavelet transform only in horizontal direction to $\mathbf{L}_l$.

The coding of $\mathbf{L}_l$ is thus done in two steps.

(1) Encode $\mathbf{L}_{l+1/2}$ based on $\mathbf{L}_{l+1}$

(2) Encode $\mathbf{L}_l$ based on $\mathbf{L}_{l+1/2}$

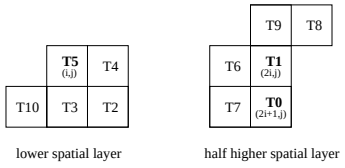


Figure 3.4: OTF templates

Fig. 3.4 shows the template for the first step. The templates for the second step are created by transposing the templates of the first step. The context numbers for T_0 and T_1 are given by

$$S_{T_1} = 2^7 T_9 + 2^6 T_8 + 2^5 T_7 + 2^4 T_6 + 2^3 T_5 + 2^2 T_4 + 2^1 T_3 + T_2 \tag{3.5}$$

$$S_{T_0} = 2^7 T_1 + 2^6 T_{10} + 2^5 T_7 + 2^4 T_6 + 2^3 T_5 + 2^2 T_4 + 2^1 T_3 + T_2 \tag{3.6}$$

The scalability of this method provides spatial layers that correspond to the object shapes needed for the subsequent passes of the shape adaptive wavelet transform.

Chapter 4

Shape Adaptive Wavelet Packet Transform

4.1 Advantages

There are several propositions on how to perform the transformation of arbitrarily shaped objects. Since the wavelet transformation of rectangular regions has been very well researched, the use of padding techniques would seem rewarding. By finding the bounding box of an object and padding values into the region that is not contained in the object-mask, the problem of arbitrary shapes is reduced to a rectangular transformation. However, because in this approach more pixels than contained in the original object have to be coded, it is not efficient. For better efficiency a way has to be found to deal with arbitrary shapes directly and to transform them without producing overhead pixels.

Egger et. al. [32] proposed to adapt a scheme similar to the shape-adaptive discrete cosine transformation (SA-DCT) to Wavelets. The idea is to flush all pixels to the left and upper corner for horizontal and vertical transformation respectively, and then perform the transformation. Because of the localization attribute of the wavelet transformation, coefficients then have to be redistributed to their previous positions. This approach produces the same amount of coefficients as in the original object. However, when the transformation is applied in the second direction, by flushing the coefficients the phases are mangled and high energy is induced, affecting the quality of the wavelet transform.

The shape-adaptive DWT (SA-DWT) proposed by Li [20] does not have this drawback. SA-DWT also produces the exact required amount of coefficients, but at the same time preserves coefficient positions. It has been adopted for the MPEG-4 Visual Texture Coding (VTC) module for coding of still textures. Below, we will give an overview of the principle mechanism of the SA-DWT, and how it deals with problems that arise from using arbitrarily shaped objects. We then propose a method to incorporate the SA-DWT into a wavelet-packet based framework.

4.2 1-D SA-DWT

When dealing with objects of arbitrary shapes, not all the image-segments for the horizontal and vertical transformation are of even length. Odd-length segments have to receive special treatment, depending on the used filters.

4.2.1 Borderextension

To ensure perfect reconstruction, the positions outside the boundaries of the segment have to be filled with values related to the boundary regions. If these regions were only padded with zeros then the transform would produce more coefficients than contained in the original signal. This detrimental effect is known as *expansiveness*. If, however, the signal is extended by taking the known boundary values and reusing them for extension in a way that reflects the properties of the used filters, it is possible to produce a *non-expansive* transformation.

The available strategies for borderextension depend on the used filter. We will discuss two strategies in this section, periodic and symmetric extension (Fig. 4.1). For short length signals, the border extension has to be applied repeatedly.

4.2.1.1 Periodic Extension

This type of extension works with all filters. However, if the segments are long and the difference between start and end regions is big, artificial high-frequency components which affect the coding efficiency are produced. Periodic extension is easy to implement:

$$c_p(i) = s(i \ mod \ n) \tag{4.1}$$

where s is the original interval, i is the requested index and n is the length of the signal.

4.2.1.2 Symmetric Extension

Symmetric Extension can only be used with biorthogonal filters. For a detailed account of this, refer to section 4.2.3. MPEG-4 defines 3 Types of symmetric border extension (Fig. 4.1). In the following, possible implementations of symmetric border extension are presented. $s(j)$, $j = 0, \ldots, n-1$ refers to the original segment, n is the number of values in the original segment and i denotes the requested index for values outside the index-set of s, i.e. $i \notin \{0, \ldots, n-1\}$.

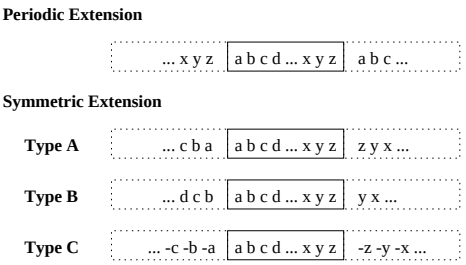


Figure 4.1: Periodic and symmetric border extension

Type A

$$c_{SA}(i) = \begin{cases} s(n-1-(i \ mod \ n)) & \text{if } (t \ mod \ 2) = 1 \\ s(i \ mod \ n) & \text{else} \end{cases} \tag{4.2}$$

where

$$t = \begin{cases} \lfloor i/n \rfloor & \text{if } i >= 0 \\ \lfloor (i-(n-1))/n \rfloor & \text{else} \end{cases}$$

Type B

$$c_{SB}(i) = \begin{cases} s(n-1-((sgn(i)*(t-1)*n-1)\ mod\ n) + sgn(i)*(-1)) & \text{if } (t\ mod\ 2) = 1 \\ s((i-sgn(i)*(t-1)*n-1)\ mod\ n + sgn(i)*(-1)) & \text{else} \end{cases} \tag{4.3}$$

where

$$t = \begin{cases} \mid \lfloor (i-1)/(n-1) \rfloor \mid & \text{if } i >= 0 \\ \mid \lfloor (i-(n-1)+1)/(n-1) \rfloor \mid & \text{else} \end{cases}$$

and

$$sgn(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x = 0 \\ -1 & \text{if } x < 0 \end{cases}$$

Type C Type-C border extension can be deduced from Type A by negation.

4.2.2 1-D Transformation

The transformation varies for each filter-type used, namely for orthogonal, oddsized biorthogonal and evensized biorthogonal filters.

4.2.2.1 Orthogonal Filters

Let m be an even number and

$$\{h(i), i = 0, \ldots, m-1\},$$

$$\{g(i), i = 0, \ldots, m-1\}$$

be the lowpass and highpass analysis filter, respectively. Further, let

$$\{s(i), i = 0 \ldots, n-1\}$$

define the values of the input segment with length n.

The lowpass and highpass synthesis filters $f(i), e(i)$ can be derived from $h(i), g(i)$ as follows.

$$f(i) = h(m-1-i),\ i = 0, \ldots, m-1 \tag{4.4}$$

$$e(i) = g(m-1-i),\ i = 0, \ldots, m-1 \tag{4.5}$$

The following equations are used for analysis for producing lowpass and highpass coefficients, respectively.

$$T(i) = \sum_{j=0}^{m-1} s(i+j-m/2+1)g(m-1-j) \tag{4.6}$$

$$S(i) = \sum_{j=0}^{m-1} s(i+j-m/2+1)h(m-1-j) \tag{4.7}$$

In the following synthesis process, $T^*(i)$ and $S^*(i)$ refer to the upsampled signals (see sec. 4.2.2.4) and $r(i)$ is the reconstructed signal.

$$u(i) = \sum_{j=0}^{m-1} T^*(i - m/2 + j) f(m - 1 - j) \tag{4.8}$$

$$v(i) = \sum_{j=0}^{m-1} S^*(i - m/2 + j) e(m - 1 - j) \tag{4.9}$$

$$r(i) = u(i) + v(i) \tag{4.10}$$

For orthogonal filters, the only possible choice of border extension is periodic (see sec. 4.2.3). In order to use orthogonal filters for transforming oddlength signals, [20] proposed to split oddlength signals into $n-1$ and 1 samples. The $n-1$ samples can be filtered using periodic extension (4.1). (4.6) and (4.7) are used for analysis, (4.8) and (4.9) for synthesis. The remaining one sample is scaled by $\sqrt{2}$ and put into the lowpass subband. Scaling by $\sqrt{2}$ is the same as extending the one sample repeatedly and applying the lowpass-filter, i.e. multiplying by factor $\sum_{j=0}^{L-1} g(i)$, which is $\sqrt{2}$ for orthogonal filters.

4.2.2.2 Oddsized Biorthogonal Filters

Let

$$\{h(i), i = 0, \ldots, m_h - 1\},\ m_h \ mod\ 2 = 1$$

be the lowpass analysis filter and

$$\{g(i), i = 0, \ldots, m_g - 1\},\ m_g \ mod\ 2 = 1$$

be the highpass analysis filter. Further, let

$$\{s(i), i = 0 \ldots, n - 1\}$$

define the values of the input segment.

We can derive the lowpass and highpass synthesis filters, $f(i)$ and $e(i)$, from the respective analysis filters:

$$f(i) = (-1)^{i+1} g(i),\ i = 0, \ldots, m_g - 1 \tag{4.11}$$

$$e(i) = (-1)^{i} h(i),\ i = 0, \ldots, m_h - 1 \tag{4.12}$$

The following equations are used for analysis, where (4.13) produces lowpass and (4.14) produces highpass coefficients.

$$T(i) = \sum_{j=0}^{m_h-1} s(i + j - (m_h - 1)/2) h(m_h - 1 - j) \tag{4.13}$$

$$S(i) = \sum_{j=0}^{m_g-1} s(i + j - (m_g - 1)/2) g(m_g - 1 - j) \tag{4.14}$$

For synthesis, the following Equations are used. (4.15) is used for lowpass and (4.16) is used for highpass. $T^*(i)$ and $S^*(i)$ refer to the upsampled signals (see sec. 4.2.2.4). $r(i)$ is the reconstructed signal.

$$u(i) = \sum_{j=0}^{m_g-1} T^*(i-(m_g-1)/2+j) f(m_g-1-j) \quad (4.15)$$

$$v(i) = \sum_{j=0}^{m_h-1} S^*(i-(m_h-1)/2+j) e(m_h-1-j) \quad (4.16)$$

$$r(i) = u(i) + v(i) \quad (4.17)$$

If $n = 1$, the analysis and the synthesis process both work in the same way as if an input signal with values of all s(0) had been received. For analysis (4.13) is applied for lowpass coefficients and the result is written into the lowpass subband (regardless of the subsampling strategy).

If $n > 1$, s(i) is extended using Type B (4.3) extension. The analysis process produces n lowpass by applying (4.13) and n highpass coefficients by applying (4.14) to the input signal. Both vectors of coefficients are downsampled by two (see sec. 4.2.2.4), producing a total of n coefficients.

The synthesis process receives a total of n coefficients. Depending on the previously used downsampling strategy, the lowpass and highpass coefficients are upsampled in different ways. However, the upsampled signals always add up to $2n$ coefficients.

Both, the lowpass and the highpass signal, are extended using Type B (4.3) extension. By applying (4.15) to the upsampled lowpass coefficients and (4.16) to the upsampled highpass coefficients, the original signal is restored.

4.2.2.3 Evensized Biorthogonal Filters

Let

$$\{s(i), i = 0 \ldots, n-1\}$$

again denote the values of the inputs signal. Let

$$\{h(i), i = 0, \ldots, m_h - 1\},\ m_h \ mod\ 2 = 0$$

be the lowpass analysis filter and

$$\{g(i), i = 0, \ldots, m_g - 1\},\ m_g \ mod\ 2 = 0$$

be the highpass analysis filter.

The lowpass filters for analysis and synthesis are both symmetric, whereas the highpass filters for analysis and synthesis are both antisymmetric. The analysis and synthesis filterpairs have the following relationship.

$$f(i) = (-1)^{i+1} g(i),\ i = 0, \ldots, m_g - 1 \quad (4.18)$$

$$e(i) = (-1)^{i} h(i),\ i = 0, \ldots, m_h - 1 \quad (4.19)$$

The equations for analysis and synthesis are the same as for oddsized biorthogonal filters (4.13, (4.14) for analysis and (4.15), (4.16) for synthesis). However, for evensized biorthogonal filters, symmetric borderextensions are applied in a different way: for analysis Type A (4.2) is used, for synthesis Type C is used. (Intuitively this makes sense, as

these borderextensions reflect the symmetric/antisymmetric properties of the used filter-pairs).

For $n = 1$ the same procedure as for oddsized biorthogonal filters is applied. $s(0)$ is extended repeatedly and then the lowpass wavelet filter is used to produce a signal coefficient, which is put into the lowpass subband (regardless of subsampling strategy).

For $n > 1$ and n even, the analysis step produces $2n$, coefficients.

For $n > 1$ and n odd, $n+1$ coefficients are produced for highpass and lowpass subband. However, as the antisymmetric Type C extension is used with an antisymmetric filter in the highpass analysis step, the last and the first coefficient in the highpass subband are always zero. Thus the number of coefficients is reduced to $2n$, with $n + 1$ coefficients for the lowpass subband and $n - 1$ coefficients for the highpass subband (Fig. 4.7.b on page 41 illustrates this).

4.2.2.4 1-D Subsampling

The wavelet packet analysis described above produces $2n$ coefficients for a signal of length n. As only n coefficients are needed for perfect reconstruction, downsampling is performed. Subsampling can be done at even or odd position with the signal being indexed starting from zero. Following [20], we call subsampling at even positions "even subsampling" and subsampling add odd positions "odd subsampling".

The following equations describe the downsampling process. D(i) is the downsampled signal and $s = 0$, if subsampling is even and $s = 1$, if subsampling is odd. S(i) is the lowpass or highpass filtered original signal.

$$D(i) = (2i + s), \ i = 0, \ldots, n - 1 \tag{4.20}$$

Because odd-sized biorthogonal filters introduce a phaseshift during the analysis step, different subsampling strategies have to be used for lowpass and highpass subbands, i.e. if the lowpass subband is subsampled at even positions, the highpass subband has to be subsampled at odd positions and vice versa. We denote the subsampling strategy for this kind of filters as "even-odd subsampling" and "odd-even subsampling", where the first part refers to the strategy used for the lowpass subband and the second part refers to the strategy for the highpass subband. For other filter types, the same subsampling strategy is applied for both subbands. We will denote these strategies as "even-even subsampling" and "odd-odd subsampling".

For an even-length signal the downsampled signal contains $n/2$ coefficients in the lowpass part and $n/2$ coefficients in the highpass part, regardless of filter and subsampling strategy used. For an odd-length signal, the number of coefficients in the lowpass and highpass part respectively depend on the used subsampling strategy and the type of the used filter. For orthogonal and evensized biorthogonal filters, $n/2 + 1$ coefficients are put into the lowpass subband and $n/2$ coefficients are put into the highpass subband, regardless of subsampling strategy. For oddsized biorthogonal filters and even-odd subsampling, the lowpass subband contains $n/2+1$ coefficients and the highpass subband contains $n/2$. For oddsized biorthogonal filters used with odd:even subsampling, the lowpass subband contains $n/2$ coefficients and the highpass subband contains $n/2 + 1$ coefficients.

4.2.2.5 Remark on Implementation

For implementing the transformation described above, it is not necessary to produce n lowpass and n highpass coefficients. The subsampling can be done on the fly, by applying (4.13) and (4.14) for lowpass and highpass filtering respectively only for the subsampling positions in the input segment. The same holds for the synthesis process. Upsampling can be done implicitly by applying (4.15) and (4.16) to the subsampled vectors and write the results in correct order.

4.2.3 Filters and Border Extensions

As implied above, not all types of borderextensions can be used for all types of filters or all length of input segments. This section aims at explaining this fact in an intuitive way, i.e. without demanding to be formally precise.

For perfect reconstruction it is necessary that the same borderextension that was used for transformation is also applicable for the result coefficients.

Example: Take a segment A of length 4: (a, b, c, d). We now repeatedly extend A periodically, and call this segment of infinite length B. When a wavelet transformation is performed on B using periodic extension, the sequence of result coefficients we expect is also a segment of 4 coefficients $(\alpha, \beta, \gamma, \delta)$ extended periodically to infinity.

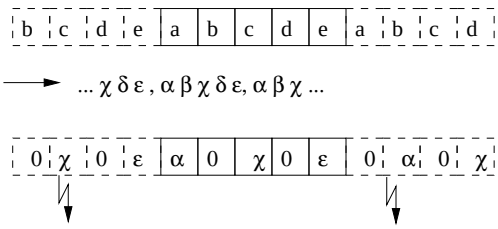


Figure 4.2: Periodic extension with odd length segments

Furthermore, periodic extension cannot be used for transforming segments of odd length. This is because the upsampling cannot be done properly in this case. Fig. 4.2 illustrates that either the periodicity constraint is broken (right hand side) or the constraint that coefficients and zeros have to occur alternately (left hand side) is broken.

With symmetric extension it is possible to filter uneven length segments. This is shown in fig. 4.3 for Type-B symmetric extension. The signal can be extended without breaking the symmetry constraint or the constraint that coefficients and zeros have to occur alternately. The same is true for the type A and type C extensions that are used in conjunction with evenlength biorthogonal filters. Neither constraint is broken, because the border extension is suited for the filter properties. In the following different filter types and possible border extensions are discussed.

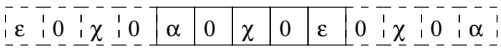


Figure 4.3: Symmetric extension with odd length segments (Type B)

4.2.3.1 Orthogonal Filters

Orthogonal filters are always even length. Fig. 4.4 shows the lowpass and highpass filter for analysis. For orthogonal filters, only periodic extension is possible. Fig. 4.5.b shows that for periodic extension of the original signal the result sequence has the same periodic properties.

In Fig. 4.5.b it is shown that the same is not true for symmetric extension. Without loss of generality, we show the case for Type B extension, but the same is true for Type A. The result sequence should have the same symmetric properties as the original segment, i.e. the coefficient denoted as ϵ should be the same as γ. However, as the filter coefficients are not symmetric themselves, the coefficients that contribute to ϵ are not the same as γ.

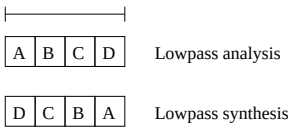
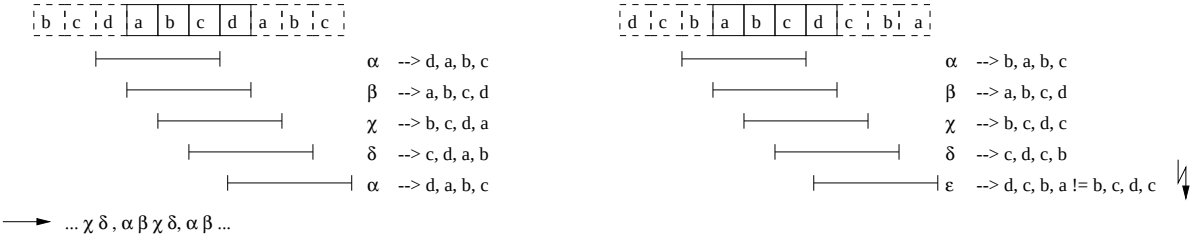


Figure 4.4: Orthogonal filter



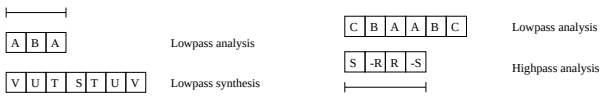
(a) Periodic extension

(b) Symmetric extension

Figure 4.5: Border extensions and orthogonal filters

4.2.3.2 Biorthogonal Filters

For oddsized biorthogonal filters, periodic and symmetric extension is possible. Fig. 4.6.a shows the schema of biorthogonal lowpass and highpass analysis filters. In Fig. 4.7.a the analysis step is shown for the lowpass filter on an even length segments. Note that the resulting coefficients have the same symmetric properties as the original signal. This is possible because the filter likewise is symmetric.



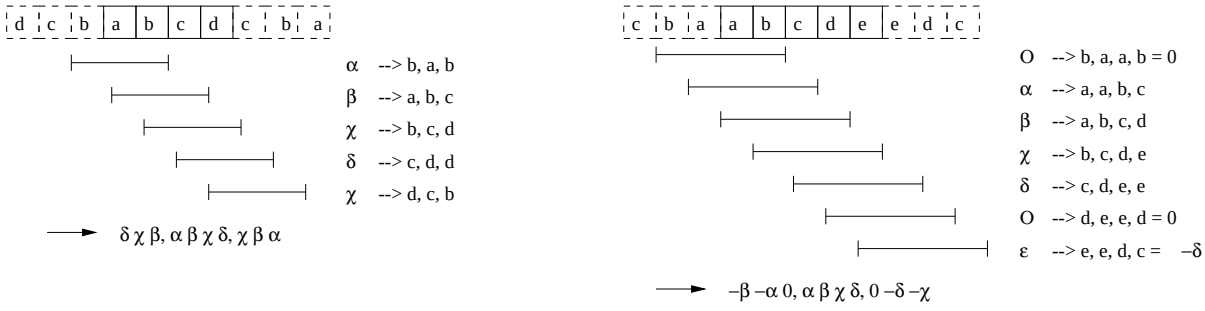
(a) Oddsized biorthogonal filter

(b) Evensized biorthogonal filter

Figure 4.6: Biorthogonal filters

For evensized biorthogonal filters the principle is slightly different. Fig. 4.6.b shows the lowpass and highpass analysis filters in this case. For an input signal of odd length n, $n+1$ lowpass coefficients and $n+1$ highpass coefficients are produced. In Fig. 4.7.b it is shown that of the highpass coefficients the first and the last are always zero (leaving only

$n-1$ coefficients of interest). Furthermore, it is shown that the sequence of coefficients produced, does have the necessary symmetric properties for perfect reconstruction.



(a) Oddsized filters

(b) Evensized filters

Figure 4.7: Border extensions and biorthogonal filters

4.3 2-D SA-DWT

4.3.1 2-D Transformation

The wavelet transform in 2-D is separable, i.e. it is possible to apply 1-D horizontal and 1-D vertical wavelet transform sequentially and produce the same result as a 2-D transform.

For arbitrary shapes, some modifications have to be made:

(1) for each line/row in a direction

- while (more segments in line)
  - find continuous segment of values
  - apply 1-D transformation to segment
- end while

end for

(2) in the second direction, start with (1) applying the transformation to the coefficients of the previous run

As the 1-D transformation produces just as many coefficients as contained in the original segment, it is obvious that the above procedure produces exactly the same amount of coefficients as there are pixels in the original object.

4.3.2 2-D Downsampling

As the wavelet transform is separable, the downsampling process is straightforward in 2-D. However, as a 1-D signal now has a global context, two more modes for subsampling can be used in the 2-D case. We refer to them as local and global (cf. [20]) mode . Local subsampling is done without taking the global position of the segment into account.

For global subsampling the local subsampling strategy depends on the start of the signal relative to the left and upper border (for horizontal and vertical transformation, respectively) of the bounding box of the 2-D object. In this case, subsampling is always done at even or odd offsets, respectively, from the border. From a local perspective, this may result in local even or local odd subsampling. As shown in fig. 4.8, all combinations of subsampling mode and strategies are possible. Positions with dark grey backgrounds represent the values that are dropped in subsampling. Depending on the used subsampling technique, the objects in different subbands differ in appearance (Fig. 4.9).

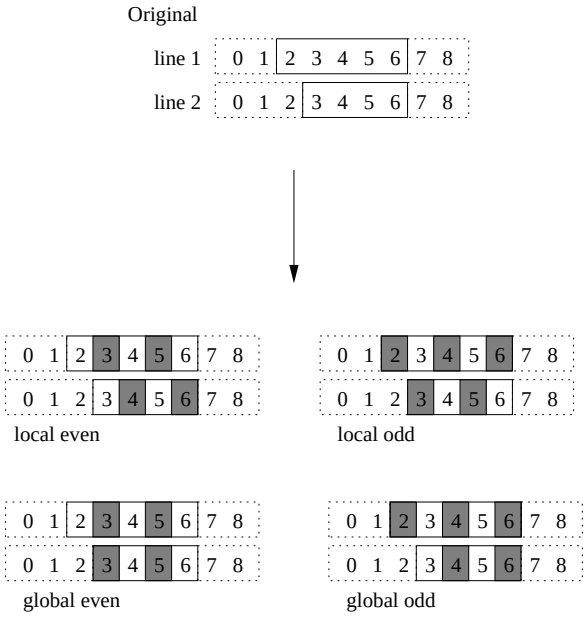


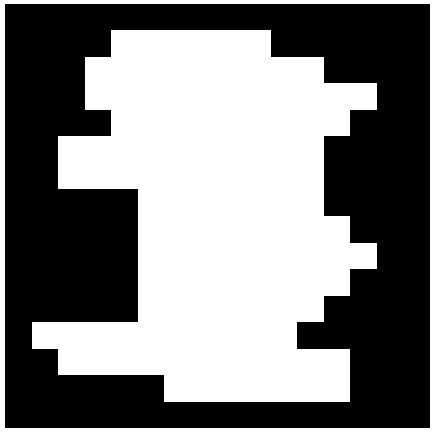
Figure 4.8: Subsampling modes and strategies

4.3.3 Remark on Implementation

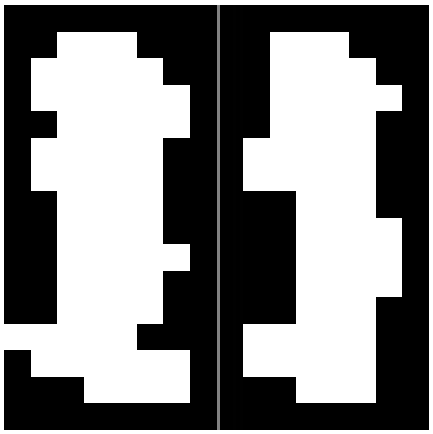
The computational complexity of the objectbased transform is the same as for the non-objectbased approach, apart from the overhead for finding continuous segments for 1D-transformation. The memory requirements exceed the requirements of the non-object-based wavelet transform, due to the necessary handling of object shapes.

Compared to the non-objectbased approach the size of the resulting bitstream is only affected by the need of inclusion of the object shape. As shown in chapter 3 there are efficient methods to encode object shapes. Progressive approaches like OTF are especially well suited, as they encode exactly the resolutions, i.e. downscaled versions, of the object shape needed for reconstruction.

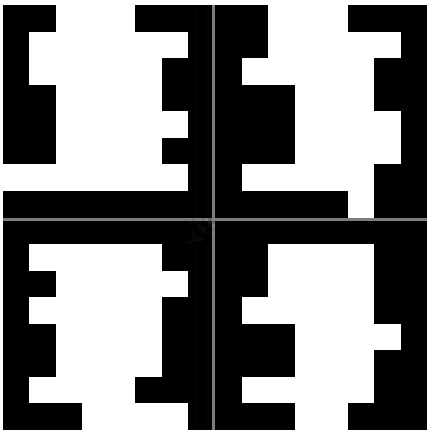
During decomposition the choice can be made whether to generate the downscaled versions of the full resolution bitmask in advance and keep them in memory or whether to downscale on the fly during the decomposition of each subband, which means less memory requirements. For the reconstruction process, if the shape was encoded using a non-progressive method, the shapes have to be downscaled in advance and kept in memory, as the result of an upscaling process is not unique. If a progressive method was used, the object shapes can be decoded from the bitstream when needed.



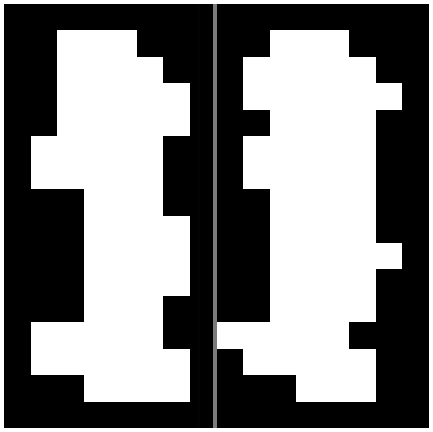
(a) Original object



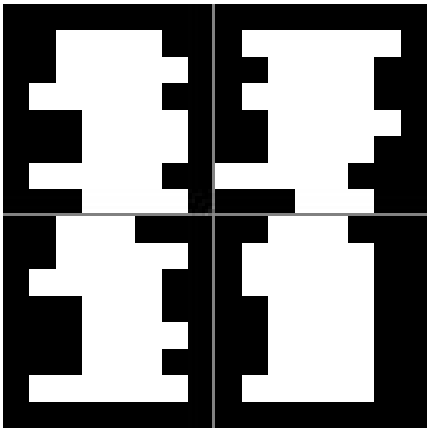
(b) Local Even-Odd - Horizontal



(c) Local Even-Odd - Level 1



(d) Global Even-Odd - Horizontal



(e) Global Even-Odd - Level 1

Figure 4.9: 2-D downsampling

4.3.4 Choice of Subsampling Mode and Subsampling Strategy

Figure 4.10: High energy patterns in local subsampling, ratio 90

Merging of two distinct segments occurs frequently in SA-DWT as the decomposition level progresses. However, the merging does not affect PSNR-performance in a high degree and is usually neglectible. The more important question is, how to arrange coefficients to prevent an artificial phase shift between subsequent lines and columns respectively. The choice of a certain combination of subsampling mode and strategy affects the resulting coefficients. If the resulting coefficients are to be encoded with a zerotree-scheme, it makes more sense to have as many coefficients in the lower subbands as possible. This can be achieved by using local mode, with even subsampling for the lowpass subband and odd subsampling for highpass subbands. The effect is that the number of coefficients in the lowpass subband is always equal (in the case of an even segment) or higher (in the case of an odd segment) than the number of coefficients in the highpass subband.

On the other hand, local subsampling has the disadvantage of mangling phases to a certain degree. Constellations exist, where this affects coding quality to a high degree. Take a pair of subsequent horizontal segments with start indices i_1, i_2 and lengths l_1, l_2. Assume that $|i_1 - i_2| < \max(l_1, l_2)$. If $i_1 \bmod 2 = 1$ and $i_2 \bmod 2 = 0$, the phases of the two lines are not compatible, high energy patterns are artificially introduced and PSNR quality is severely affected (see fig. 4.10 for a reconstructed image). In order to preserve images quality and favor a gain in PSNR, the global approach is to be preferred in such a case, because all pairs of lines are of the same phase here. Note that for oddlength segments there is more than one possibility where to put the smaller of the two result segments, as shown in fig. 4.11. For optimal PSNR performance, the positions have to be chosen so that the transformation in the other direction has coefficients of the same phase in each of its segments. Whether globally even-odd or odd-even subsampling (referring to the strategy of lowpass and highpass subband) produces better results, depends much on the shape used. Having more coefficients in the lowpass subband is of advantage. This issue will be discussed in more detail in chapter 5, where we will also present relevant results.

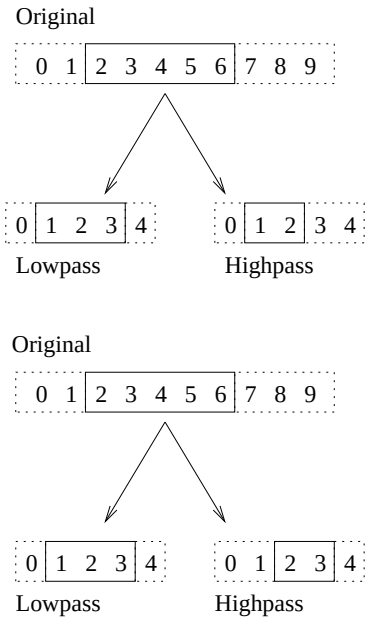
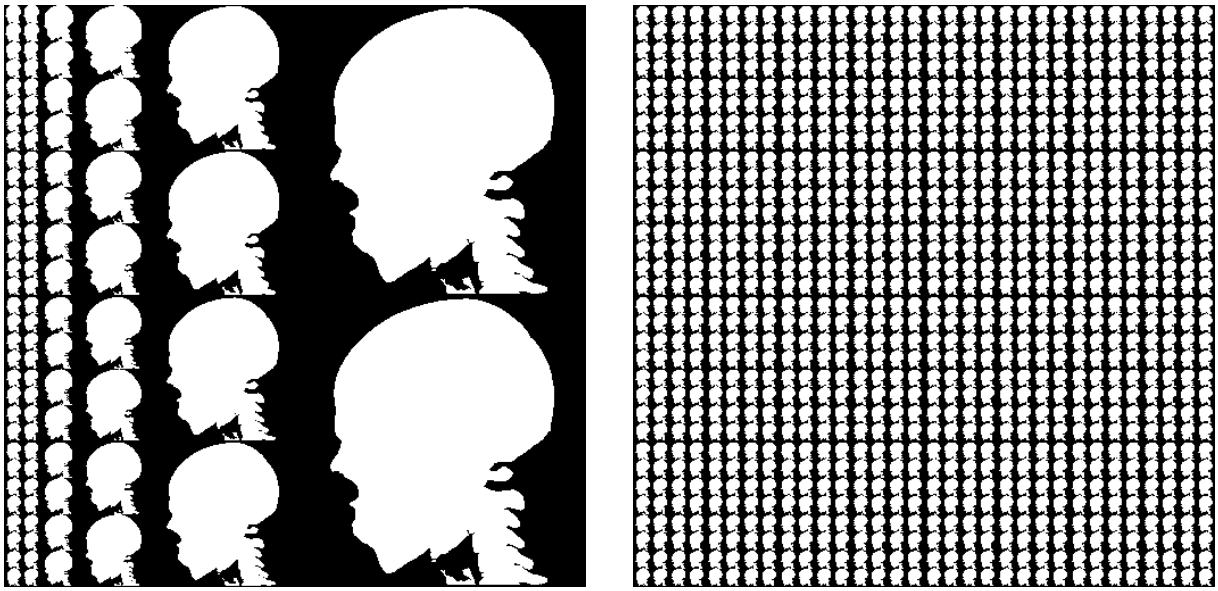


Figure 4.11: Positioning for oddlength segments

4.4 SA-Wavelet Packet Transform (SA-WPT)



(a) Logarithm of Energy

(b) Full Decomposition

Figure 4.12: Subband masks for wavelet packet decomposition

The extensions to SA-DWT necessary to perform a wavelet packet transformation are not limited to the transformation as such, but also have a considerable impact on other areas of the coding process. Especially for zerotree coding, an altered paradigm has to be used. We will discuss the relevant alterations in section 5.5.

After determining the decomposition structure for a given object, e.g. by using the best basis algorithm (see sec. 2.4) the actual transformation process can be used as presented in section 4.3. However, attention has to be paid to the different shapes in the various subbands of the decomposition. The general wavelet packet decomposition can contain a larger number of subbands at various scales than the pyramidal wavelet decomposition, and due to the chosen downsampling strategies, each subband usually contains a different bitmask (see fig. 4.12).

4.5 Results

We limit our testruns to biorthogonal filters for the objectbased transform, where a direct objectbased transformation is possible. We still vary the used filters to investigate how the choice of filter affects performance. Also, we are interested in the coding performance for different shapes and, of course, for different textures. The parameter space thus tends to get fairly large and we will only present the most important results.

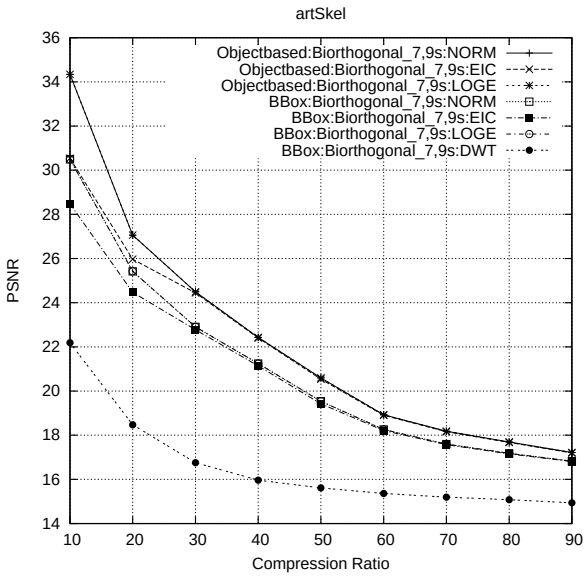
In the following, we compare the objectbased approach using SA-WPT to a bounding box strategy to transform the object. In the latter, first the bounding box is found, then the transformation is performed. For both approaches, compression ratio and PSNR are measured only for pixels actually contained in the object.

4.5.1 Costfunctions

The comparison of different costfunction is mainly done implicitly in the other test categories. We will only discuss the results for fig. 4.13.a, *artSkel*. As can be seen in fig. 4.13.b, the objectbased approach outperforms the bounding box approach over the whole bitrange considered.

It is interesting to observe that the best results for the non-objectbased image *artificial* (fig. 2.3.a) and the bounding box approach are produced using the ℓ-Norm costfunction, whereas in the objectbased case, the best results are achieved using the LogE-costfunction. The decomposition structure produced by LogE and ℓ-Norm in the objectbased approach is the same that is produced by LogE for *artificial*.

(a) *artSkel*

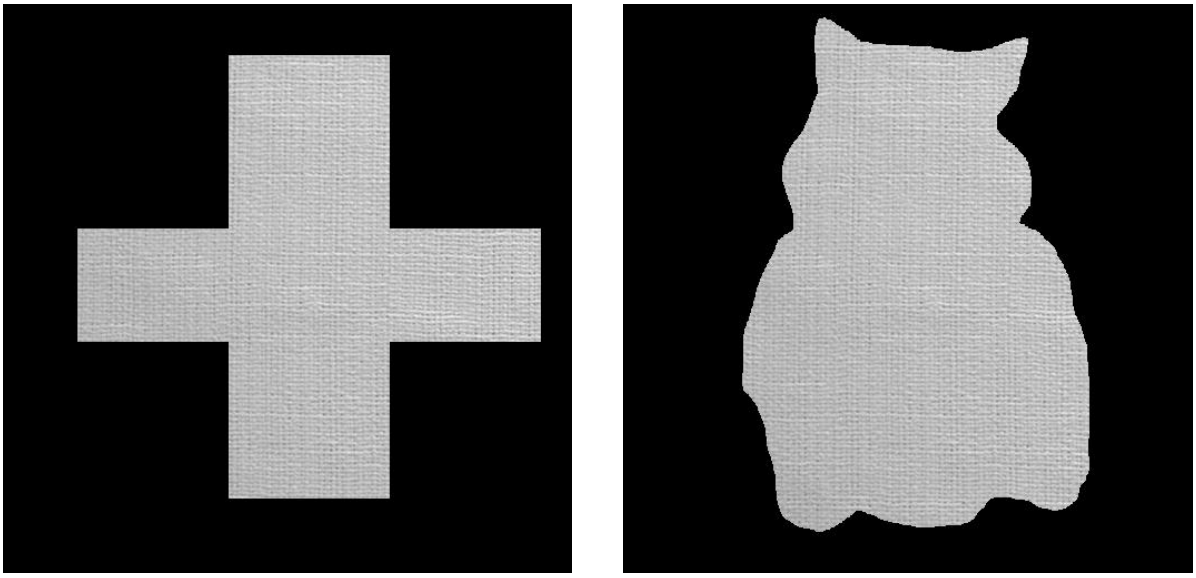


(b) Results for fig. 4.13.a

Figure 4.13: Shapes: Costfunctions

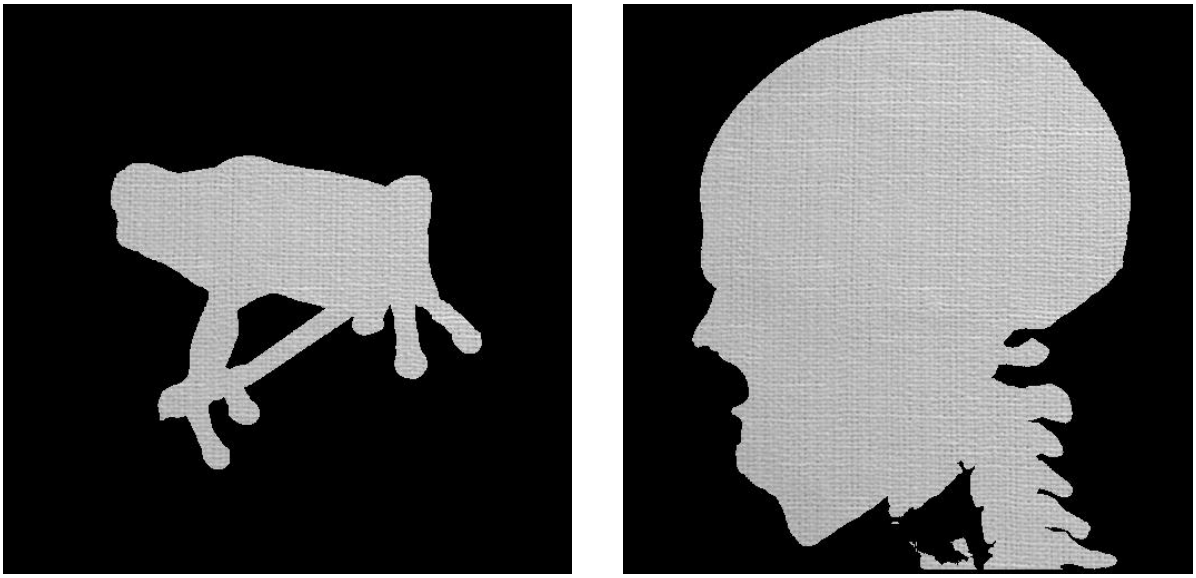
4.5.2 Shapes

We compare four different shapes shown in fig. 4.14 and fig. 4.15. The shapes feature rising levels of complexity – whereas *cross* does not have any rough edges and is only a little more complex than a simple rectangle, *skel* features a lot of uneven edges as well as smaller regions only partly or not at all connected to the rest of the shape. We compare



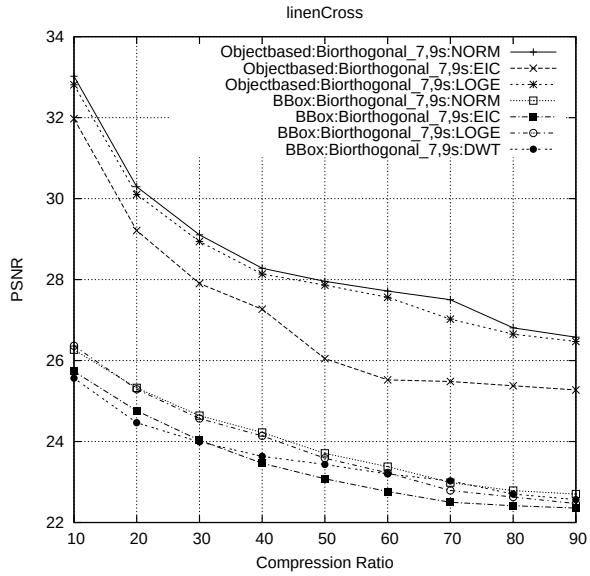
(a) *linenCross* (b) *linenPig*

Figure 4.14: Shapes: Test Images I

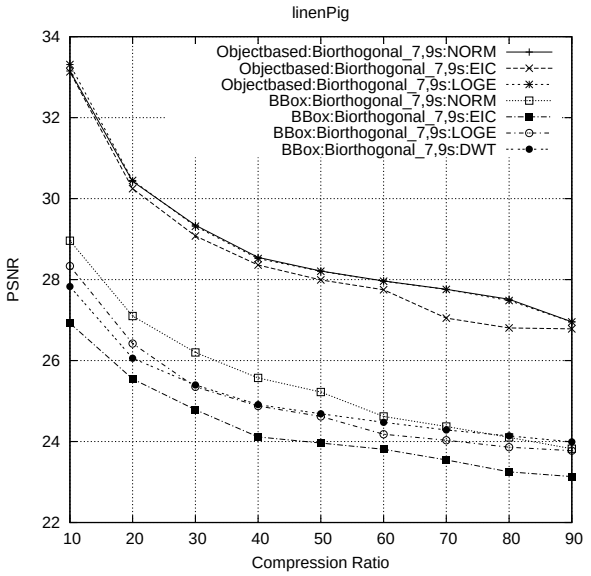


(a) *linenFrog* (b) *linenSkel*

Figure 4.15: Shapes: Test Images II

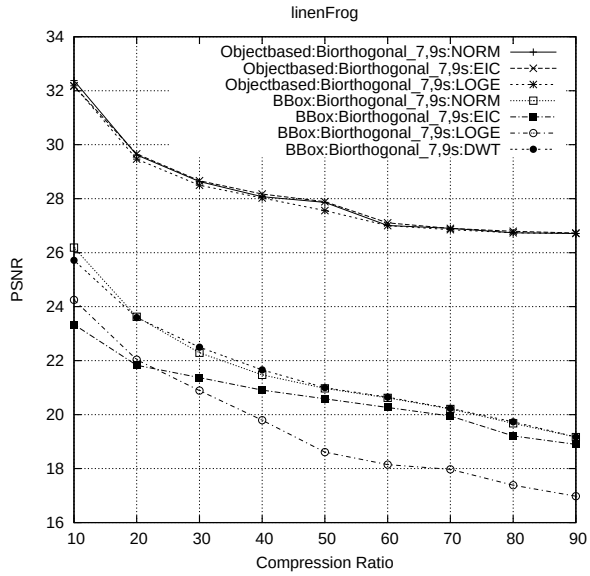


(a) Results for fig. 4.14.a

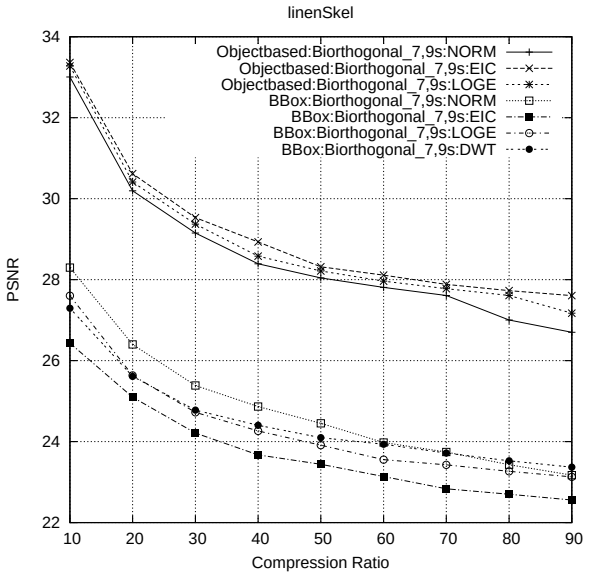


(b) Results for fig. 4.14.b

Figure 4.16: Results for fig. 4.14



(a) Results for fig. 4.15.a



(b) Results for fig. 4.15.b

Figure 4.17: Results for fig. 4.15

the performance of the SA-WPT for these shapes with the test-images shown in fig. 4.14, which all use the *poly_linen*-texture.

Obviously, it is not possible to compare the PSNR-results for the different shapes directly as they are all of different sizes. But, as can be seen in fig. 4.16 and fig. 4.17, the distance between the objectbased approach and the bounding box approach is considerable, regardless of the shape. It has to be taken into account, of course, that the cost for coding the bitmask or the WPT-structure were not included in order to illustrate how the gain through adaptive objectbased approach can compensate the introduced overhead, even exceed it. As regards the performance of costfunctions in the objectbased approach, it can be seen that the same costfunction, in this case the ℓ-Norm, produces the best results for all the shapes.

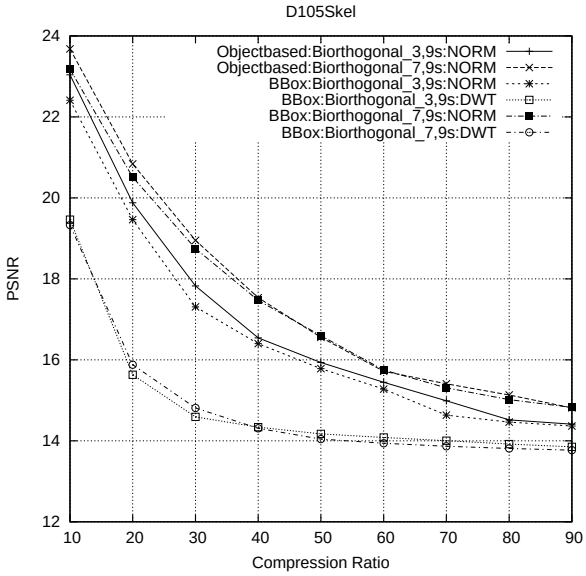
The better performance of the objectbased approach as opposed to the bounding box approach has a variety of reasons. Obviously, the objectbased approach only codes pixels that are actually contained in the object and does not waste precious bits on pixels outside the object shape. Further, in the approach using bounding boxes, the border regions of positions in the object and positions outside the object containing padded values introduce artificial changes in the frequential patterns and thus is problematic not only for coding efficiency but also affects the performance of the BBA. Fig. 4.20 shows the difference in visual quality of reconstructed versions of *D51Skel* (fig. 4.19.a) for the approaches, obtained at compression ratio 80.

4.5.3 Filters and Textures

The gains in compression efficiency are not that high for all textures. The results for fig. 4.18.a, for example, are not as striking as other results. Fig. 4.18.b shows that in this case the bounding box approach produces more competitive results, although the general level of PSNR is very low.

As regards the used filters, we only compare the standard filter of JPEG-2000, *biorthogonal 7,9*, to the standard filter of MPEG-4 VTC, *biorthogonal 3,9*. As shown in fig. 4.18.b and fig. 4.19.b, *biorthogonal 7,9* outperforms *biorthogonal 3,9* over the whole bitrange considered. This is the case for most of the tested object textures, which all exhibit strong oscillatory patterns.

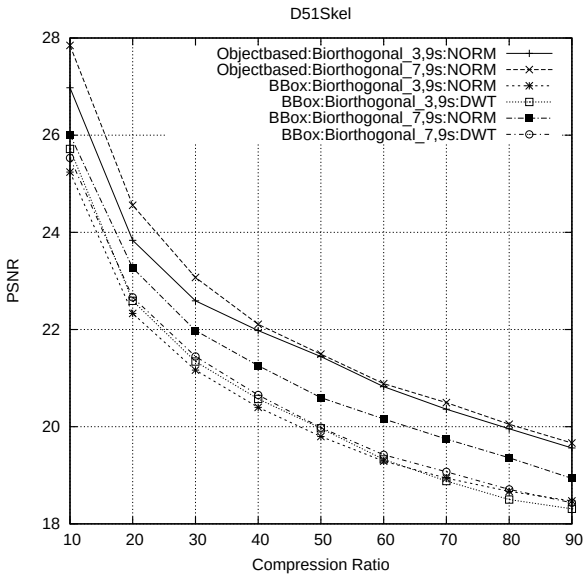
(a) *D105Skel*



(b) Results

Figure 4.18: Comparison of filters for fig. 4.18.a

(a) *D51Skel*



(b) Results

Figure 4.19: Comparison of filters for fig. 4.19.a

(a) Pyramidal, Bounding box, *biorthogonal 7,9*

(b) WPT ℓ-Norm, Bounding box, *biorthogonal 7,9*

(c) WPT ℓ-Norm, Objectbased, *biorthogonal 3,9*

(d) WPT ℓ-Norm, Objectbased, *biorthogonal 7,9*

Figure 4.20: Comparison of reconstructed images for fig. 4.19.a, ratio 80

Chapter 5

Zerotree Coding (ZTC)

5.1 Introduction

Zerotree coding uses the hierarchical and self-similar structure of the wavelet decomposition to increase coding efficiency for image wavelet coefficients. The advantage of this technique is that not only does it produce competitive results, but it is also low in computational complexity.

The concept was first proposed by Shapiro in [33]. Since the original paper, zerotree coding has received a lot of attention in the scientific community and now exists in several variations. In the following we will discuss some basic concepts and then present some of these variations. First, we will discuss techniques of zerotree coding for the classical wavelet decomposition that form the basis for the approach we use in our testruns. We will then go on to present approaches that adapt the zerotree-paradigm to wavelet packets. Concludingly, we will present an approach to combine wavelet packet zerotree coding with the functionality for arbitrary shapes.

A wavelet coefficient c is called insignificant with regard to a threshold T, if $|c| < T$. For a motivation that coefficients of greater magnitude are indeed more important than coefficients of smaller magnitude see 5.3. The crux of successful image coding is to encode the positions of significant coefficients efficiently. One way to do this is by so-called *significance maps*. Significance maps represent the binary decision for each coefficient whether it is significant (with respect to a threshold T) or not. However, especially for low-bitrate coding, the portion of the entire bitbudget needed to encode the significance map is considerable. This is where zerotree coding tries to achieve better compression.

5.2 Embedded Zerotree Wavelet Algorithm (EZW)

5.2.1 Significance Map Coding

The basic idea of EZW and zerotree coding as such, is to make use of the hierarchical structure of the wavelet decomposition and introduces zerotrees to achieve better compression of significance maps. Subbands at higher levels, which usually contain more of the total energy of the image in fewer coefficients, are related to subbands at lower levels

in a way that can be expressed by a parent-child relationship. EZW is based on the assumption that if a coefficients in the parent subband is insignificant, then all the coefficients of the same orientation and in the same spatial location in the child subbands are insignificant as well. A coefficient c is part of a zerotree Z (with respect to a threshold T), if it is insignificant (with respect to T), and all its children are insignificant (with respect to T) as well. If such a coefficient does not have an insignificant parent, it is called a *zerotree root*.

Shapiro proposes to use 4 symbols for encoding significance maps with zerotrees:

- zerotree root (ZTR)
- isolated zero (IZ)
- positive significant (PS)
- negative significant (NS)

With the hierarchical pyramidal structure of the wavelet decomposition, a zerotree root at a high decomposition level, can express insignificance for a large set of child coefficients in lower levels with only one symbol (see Fig. 5.1). For example, in a five level wavelet decomposition, a zerotree root in the approximation subband, i.e. in the highest level of decomposition, can denote $3\sum_{j=1}^{5} 4^j = 4029$ insignificant coefficients. Of course, the approximation subband is a special case, as each of its coefficients has three times four child coefficients. Other subbands (excluding the lowest level of decomposition) have four child coefficient per parent.

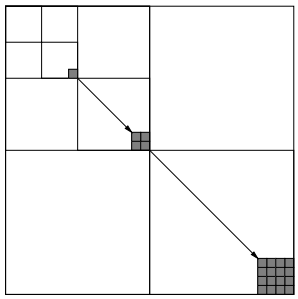


Figure 5.1: Parent/child relationship between wavelet coefficients

An *isolated zero* is an insignificant coefficients that has significant child coefficients. *Positive significant* and *negative significant* denote significant coefficient greater than zero and less than zero respectively.

5.2.2 Successive Approximation Quantization (SAQ)

SAQ uses a logarithmic sequence of thresholds of powers of two, and encodes the significance map following the above described procedure for each of these thresholds. In principle, this is related to bitplane coding.

The initial threshold T_0 is chosen that

$$|c_i| < T_0, \ \forall \text{ coefficients } c_i. \tag{5.1}$$

Further thresholds T_{i+1} are calculated by

$$T_{i+1} = T_i/2. \tag{5.2}$$

The actual encoding is done in a *dominant* and a *subordinate* pass, both with an associated list of coefficients. The *dominant list* contains coefficients that have not yet

been found to be significant. During the dominate pass, each coefficient in this list is compared to the current threshold T_i. If a coefficient c_i is found to be insignificant, it is encoded as ZTR or IZ. Otherwise, its sign is encoded using PS or NS and $|c_i|$ is appended to the subordinate list.

Thus, the *subordinate list* contains the magnitude of coefficients that have been found to be significant before. In the following subordinate pass, these coefficients are refined by the encoding of one further bit per coefficient.

Encoding is stopped, when a target criteria is met. This could be a distortion criteria or a certain bitrate being met. This can happen at an arbitrary point in time. The bitstream produced is still embedded, i.e. it contains lower rates in the same order and form that would have been created had the lower rate been the target rate in the beginning.

The resulting bitstream is finally processed by an adaptive arithmetic encoder. The encoder adapts the alphabet to the output of each pass. For a subordinate pass an alphabet containing 2 symbols is used. For a dominate pass with no zerotree root symbol an alphabet with 3 symbols is used and for a dominate pass with zerotree root symbol an alphabet containing the full 4 symbols is used.

5.3 Set Partitioning in Hierarchical Trees (SPIHT)

Said and Pearlman give an alternative explanation of the principles of operation of EZW in [34], along with a proposition for an enhanced implementation as regards coding efficiency. They name

- partial ordering by magnitude with a set partitioning sorting algorithm
- ordered bit plane transmission
- exploitation of self-similarity across different scales of a wavelet transform

as the principles of EZW.

Further, they explain why coefficients should be ordered by magnitude for transmission. We will present this explanation here in a compressed version. The MSE measurement (see (1.1)) is invariant to the unitary wavelet transformation, i.e.

$$\mathrm{MSE}(\mathbf{I}, \hat{\mathbf{I}}) = \mathrm{MSE}(\mathbf{C}, \hat{\mathbf{C}}) = \frac{1}{n^2} \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} (C(i,j) - \hat{C}(i,j))^2 \tag{5.3}$$

where n^2 is the number of pixels in the original image $\mathbf{I}$. Transmitting coefficient $\hat{C}(i,j)$ with the exact value of $C(i,j)$ lowers the MSE by $|C(i,j)|/n^2$. Thus, the higher the magnitude of a coefficient, the more is its impact on the MSE. So for fast gain in PSNR, the largest coefficients should be transmitted first. Analogous to (5.3), we can state that the most significant bits should be transmitted first for achieving the fastest reduction of distortion even for low bitrates. (Note that this reasoning is relative to MSE as a measure. One could argue that MSE is not the best measure to use as regards reduction of distortion.)

5.3.1 Set Partitioning And Wavelets

Said and Pearlman propose not to transmit the ordering of coefficients explicitly, but to use a *set partitioning sorting algorithm* which is known to encoder and decoder alike, to transmit the ordering implicitly. The set of coefficients $\{c_{i,j}\}$ is partitioned according to significance relative to a series of thresholds $\{T_n = 2^n\}$. A subset $\mathcal{P}_n$ contains coefficients $c_{i,j}$ with $T_n \leq |c_{i,j}| < T_{n+1}$, i.e. all coefficient that are significant for the first time for threshold T_n. Obviously,

$$\mathcal{P}_n \cap \mathcal{P}_m = \emptyset, \ \forall n, m. \tag{5.4}$$

Further, if the largest threshold $T_m = \max_n(\{T_n\})$ is chosen to be

$$T_m = 2^m \text{ with } m = \lfloor \log_2(\max_{i,j}(\{|c_{i,j}|\})) \rfloor, \tag{5.5}$$

then also

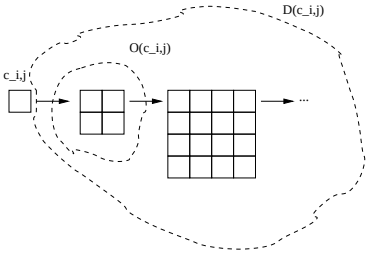
$$\bigcup_{n=0}^{m} \mathcal{P}_n = \{c_{i,j}\}, \tag{5.6}$$

the initial set of coefficients.

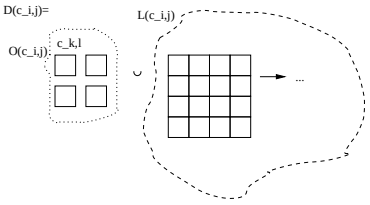
In order to achieve this partitioning, the significance test is performed on each subset. If a subset is found to contain significant coefficients, it is further divided.

$$\max_{c_{i,j} \in \mathcal{P}_n} < 2^n \ \rightarrow \ \mathcal{P}_n \text{ is insignificant} \tag{5.7}$$

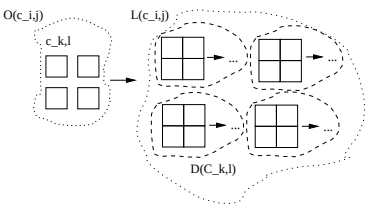
$$\max_{c_{i,j} \in \mathcal{P}_n} \geq 2^n \ \rightarrow \ \mathcal{P}_n \text{ needs to be divided} \tag{5.8}$$



(a) $\mathcal{O}(c_{i,j})$ and $\mathcal{D}(c_{i,j})$



(b) Partitioning for $\mathcal{D}(c_{i,j})$ significant



(c) Partitioning for $\mathcal{L}(c_{i,j})$ significant

Figure 5.2: Set partitioning and wavelet coefficients

The hierarchical self-similar structure of the wavelet transformation is used for the division in (5.8). The same basic principle that underlies EZW is used for the set partitioning algorithm. We will outline the method without discussing the actual algorithm in detail (for a thorough description see [34]).

Let $\mathcal{H}$ be the set of coefficients of the approximation subband, let $\mathcal{O}(c_{i,j})$ be the set of direct offsprings of coefficient $c_{i,j}$ and let $\mathcal{D}(c_{i,j})$ be the set of descendants of coefficient $c_{i,j}$, i.e. all direct and indirect children of $c_{i,j}$ (see Fig. 5.2.a). Further let $\mathcal{L}(c_{i,j}) = \mathcal{D}(c_{i,j}) \backslash \mathcal{O}(c_{i,j})$. The initial partition is formed by $\{c_{i,j}\}$ and $\mathcal{D}(c_{i,j})$ with $c_{i,j} \in \mathcal{H}$. The partitioning rules for each threshold T_n are as follows:

(1) If $\mathcal{D}(c_{i,j})$ is significant with respect to T_n then it is partitioned into $\mathcal{L}(c_{i,j})$ and $c_{k,l} \in \mathcal{O}(c_{i,j})$ (Fig. 5.2.b)

(2) If $\mathcal{L}(c_{i,j})$ is significant with respect to T_n then $\mathcal{L}(c_{i,j})$ is partitioned into four sets $\mathcal{D}(c_{k,l})$ with $c_{(k,l)} \in \mathcal{O}(c_{i,j})$ (Fig. 5.2.c). For these sets, apply (1).

These rules are subsequently applied for each threshold. Like in EZW, for coefficients that have been found to be significant previously, only refinement bits are transmitted.

5.4 Zerotree Wavelet Entropy (ZTE) and Multiscale ZTE (MZTE) Codecs

MZTE is part of the visual texture coding tool (VTC) of the MPEG-4 standard. It is based on ZTE, which in turn is based on EZW, and enhances the former with multiscale functionality. We will give a very brief overview of both techniques. For a more detailed discussion of ZTE and MZTE see [4].

5.4.1 Zerotree Wavelet Entropy Codec (ZTE)

Sodagar et al. [4] name four major areas where ZTE differs from EZW:

- explicit quantization
- coefficient scanning, tree growing, and coding are done bit-plane by bit-plane
- coefficient scanning can be done subband by subband or depth-first
- different alphabet of symbols

ZTE summarizes the coefficient of a single tree of wavelet coefficients (i.e. a coefficient in the approximation subband with all its descendants in the subbands of finer scale) in so-called wavelet-blocks. A wavelet block thus contains all coefficients of all orientations and scales corresponding to a specific spatial location. Obviously, a wavelet decomposition contains just as many wavelet blocks as there are coefficients in the approximation subband.

The approximation subband is encoded separately from the other subbands using a differential pulse code modulation (DPCM) technique. For scanning the coefficients in the detail subbands, ZTE gives the choice of *tree-depth-first*, i.e. each wavelet block is completely scanned before the next, and *band-by-band*, which corresponds to the method used in EZW in which each subband is completely scanned before the next, going from coarse to fine scale, i.e. from low to high frequency. Explicit quantization can be performed before or after the scanning process or done adaptively during the scanning process.

ZTE use four symbols for entropy coding:

- zerotree root (ZTR)
- valued zerotree root (VZTR)
- value (VAL)

- isolated zero (IZ)

A valued zerotree root is a significant coefficient whose set of descendants only contains insignificant coefficients. An isolated zero is an insignificant coefficient whose set of descendants contains at least one significant coefficient. Note that, other than EZW, ZTE only uses one symbol VAL for a significant coefficients that are not valued zerotree roots, regardless of its sign. The sign of a valued coefficient is encoded during the subsequent arithmetic encoding step.

5.4.2 Multiscale Zerotree Wavelet Entropy Codec (MZTE)

MZTE enhances ZTE with scalability while preserving the spatial localization properties of wavelet blocks. For this purpose, it uses a sequence of quantizers $(Q_i,\ i = 1\ldots, n)$, with Q_1 being the coarsest quantizer.

Let $\mathbf{C}$ be the initial set of coefficients. Further let $\mathbf{O_i}$ be the output of quantizer Q_i and let $\hat{\mathbf{C}}_\mathbf{i}$ be the coefficients that are produced by reconstructing $\mathbf{O_i}$. We define $\mathbf{O_0}$ as $\mathbf{C}$.

$\mathbf{O_{i+1}}$ is produced by applying $Q_i + 1$ to the residual of $\mathbf{O_i} - \mathbf{1}$ and $\mathbf{O_i}$.

Zerotree coding is done for each O_i with $i > 1$, resulting in a sequence $S_i,\ i = 1, \ldots, n$, which is finally arithmetically encoded. MZTE makes use of the dependencies between $\mathbf{O_i}$ and $\mathbf{O_{i+1}}$ for zerotree and arithmetic coding. Zerotree symbols used in $\mathbf{O_i}$ determine the range of possibilities in $\mathbf{O_{i+1}}$ (see Fig. 5.3).

(a) Not coded (X) or ZTR in $\mathbf{O_i}$

(b) IZ, VZTR or VAL in $\mathbf{O_i}$

Figure 5.3: Dependencies between different layers in MZTE

5.5 Extending Zerotrees to Wavelet Packets

The extension of zerotrees to wavelet packets is not a trivial task. Some of the constraints that are given in the pyramidal decomposition are missing in a wavelet packet decomposition. The hypothesis that the magnitude of coefficients decays with frequency is to be questioned [33]. Furthermore, a more sophisticated definition of the parent-child

relationship between subbands has to be used, as the relationship is not as clear as in the pyramidal case. With wavelet packets it is possible that a child subband is at a coarser scale than its parent, i.e. a coefficient in the child subband has multiple possible parents, resulting in a *parenting conflict* [35].

Xiong et al. [15] propose to avoid this parenting conflict in the first place by considering it in the selection of the best basis and merging children that have parents at a coarser scale. However, this approach severely limits the flexibility of the best basis algorithm to adapt to a given signal.

5.5.1 Compatible Zerotrees

Rajpoot et al. [35] propose a zerotree structure called *compatible zerotree* to resolve the parenting conflict. For the initial creation of a compatible zerotree, the parent-child relationship is defined between whole subbands (unlike the spatial orientation trees of [34], where the relationship is defined between single coefficients, see 5.3). Accordingly, a node in a compatible zerotree refers to a whole subband. After the initial tree has been created, its nodes are reordered to resolve parenting conflicts.

The prerequisite for the use of compatible zerotrees is that the approximation subband is at the coarsest scale. The initial three trees are created from the three children of the approximation subband, by applying the following rules recursively ([35, 36]) :

(1) If a node P is followed by four nodes $C_i,\ i = 0, \ldots 3$ at the same scale, P is set the parent of all four of them

(2) If four nodes $P_i, ,\ i = 0, \ldots 3$ are followed by four subbands $C_i,\ i = 0, \ldots 3$, then P_i is set the parent of C_i for $i = 0, \ldots, 3$

(3) If a node P at a coarser scale is followed by only one node C at a finer scale, then P is set the parent of C

Note that rule 3. represents the pyramidal case. Each of the three resulting trees represents subbands of a different orientation, i.e. horizontal, vertical and diagonal. The parental conflict, however, is still there. Coefficients in different nodes may still have multiple candidates as parents. Only the final reordering step solves the conflict by moving nodes of coarser scale up in the tree, adhering to the following rule:

(4) If a node P is at a finer scale than four of its children $C_i,\ i = 0, \ldots 3$, then the parent-child relationship between P and $C_i, i = 0, \ldots, 3$ is dissolved and the child nodes C_i of coarser scale are moved up in the tree

For moving a set of coefficients $C_i, i = 0, \ldots, 3$ toward the root of the tree, an ancestor with the nearest (but not finer) scale and nearest orientation is scanned for. The resulting compatible zerotree has parent-child relationships defined in a way that no child has a parent at finer scale.

Rajpoot et al. also rephrase the zerotree hypothesis for wavelet packets: "If a wavelet coefficient of a node from the compatible zerotree is insignificant, it is more likely that the

wavelet packet coefficients at the same similar spatial locations in all the descendent nodes of the same compatible zerotree will be insignificant as well." Obviously, this hypothesis is not as strict as the original zerotree hypothesis ([33]), but as shown in [35], it still succeeds in creating a topdown hierarchy of wavelet packet subbands that can be used for prediction.

5.5.2 Significance Map Based Adaptive Wavelet Zerotree Codec (SMAWZ)

Our own tests are based on SMAWZ [24]. SMAWZ uses a zerotree like structure called *similarity trees*. The similarity tree resembles the compatible zerotree in that is defines a parent-child relationship between whole subbands. However, the parental conflict is not resolved by moving coarse scale subbands up the tree. If a node C has multiple possible parents $P_i,\ i = 0, \ldots 3$, only P_0 (i.e. the subband of lowest frequency range) is chosen as parent, while the others are marked as childless. Note that, other than with compatible zerotrees, no restriction as to the scale of the approximation subband needs to be taken into account.

By following a set of rules, a hierarchical representations of the subbands in the decomposition tree is produced. Even though the resulting similarity tree only connects leaf subbands of the decomposition tree (see sec. 2.4), the rules also use nodes that are not leaves in intermediate stages. It is important to point out that even though the whole structure of the decomposition tree is used, there is never the need to access the data contained in the coefficient tree. The algorithm for creating similarity trees is purely structural.

The initial relationship is defined by the approximation subband of first level decomposition with the root of the coefficient tree, i.e. the original image as a whole. Kutil defines two types of similarity trees: *"across"* and *"separated"*. The rules for each differ slightly and whereas *"across"* produces a similarity tree with a single root, *"separated"* features multiple roots. We will only outline the basic rules for *"across"* here. For a more detailed description see [24]. The rules are worked through in order. Only if a rule with a lower number fails, the next higher number is tested for applicability. Each rule consists of two parts. The first part describes the situation in the *decomposition tree*. The second part defines how to resolve a parent-child relationship in the *similarity tree*. As the structure of the decomposition tree is never changed, the second part of each rule always refers to changes of relationships in the similarity tree.

(1) Let N be a non-leaf node with parent P and children $C_i,\ i = 0, \ldots, 3$. Further let $D_i \neq N,\ i = 1, \ldots, 3$ be the other children of P, i.e. "neighboring" nodes to N.

A parent-child relationship between N and P in the similarity tree is resolved by setting N the parent of C_0 and D_i the parent of C_i for $i = 1, \ldots, 3$

(2) Let N be a leaf node with parent P. Let $D_i \neq N,\ i = 1, \ldots, 3$ be the other children of P.

A parent-child relationship between N and P in the similarity tree is resolved by setting N the parent of D_i for $i = 0, \ldots, 3$

(3) Let N and M be non-leaf nodes of same scale with children C_i and D_i, respectively, with $i = 0, \ldots, 3$.

A parent-child relationship between N and M in the similarity tree is resolved by setting C_i the parent of D_i for $i = 0, \ldots, 3$

(4) Let N be a leaf node and M be a a non-leaf node of the same scale as N with children $D_i,\ i = 0, \ldots, 3$.

A parent-child relationship between N and M in the similarity tree is resolved by setting N the parent of $D_i,\ i = 0, \ldots, 3$.

(5) Let N be a non-leaf node with children $C_i,\ i = 0, \ldots, 3$. Further, let M be a leaf node of the same scale as N.

A parent-child relationship between N and M in the similarity tree is resolved by setting C_0 the parent of M. (This corresponds to the case mentioned above, in which there are multiple possible parents for a single node).

The resulting similarity tree, which connects the leaf subbands of the decomposition tree, is used to encode the actual wavelet coefficients with a significance map based approach and finally arithmetic coding. We chose the SMAWZ codec for our test runs, because it produces competitive results while at the same time providing rich functionality.

5.6 Extending zerotrees to Arbitrary Shapes

5.6.1 Extensions for Bitmasks

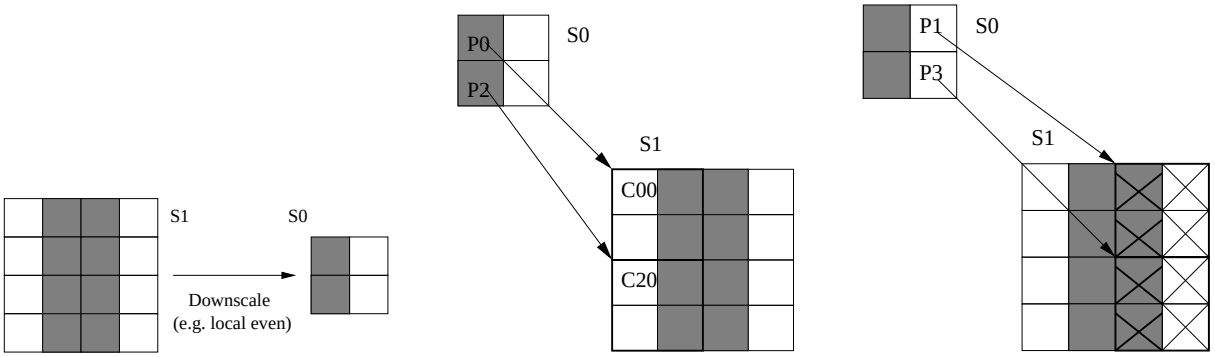
Let S_0, S_1 be subbands, where S_1 is one scale finer than S_0, $|S_0| = 4$ and $|S_1| = 16$. Let B_1 be the bitmask of an arbitrary shape in S_1. Let B_0 be the bitmask of the same arbitrary shape in S_0 that is constructed according to the chosen downscaling strategy and mode (e.g. *local-even*, Fig. 5.4.a).

Further let $P_i : i = 0, \ldots, 3, P_i \in S_0$ be the coefficients contained in S_0 and $C_{i,j} : i = 0, \ldots, 3,\ \ j = 0, \ldots, 3$ be coefficients contained in S_1 with $parent(C_{i,j}) = P_i$. Assume that P_i significant $\forall\ P_i \in B_0$ and $C_{i,j}$ significant $\forall\ C_{i,j} \in B_1$.

In Fig. 5.4.b the zerotree relationship is shown for P_0 and P_2. Both, P_0 and P_2 are contained in B_0. Even though not all child-coefficients at a finer scale are contained in B_1 no problem occurs for zerotree coding. However, in Fig. 5.4.c a case is shown where special treatment is necessary. Neither P_1 or P_3 are contained in B_0, but for $i = 1, 3$ the following equation holds

$$\exists C_{i,j} :\ parent(C_{i,j}) = P_i,\ C_{i,j} \in B_1,\ C_{i,j} \text{ significant.} \tag{5.9}$$

This is a situation that cannot be deduced by the decoder, if the parent coefficients that are not contained in the bitmask are not encoded at all. A possible solution is to encode the parent coefficients with a special symbol in this case, denoting a position not contained



(a) Downscale (b) OK (c) Problem

Figure 5.4: Bitmasks and zerotrees

in the bitmask but with significant child-coefficients. However, this approach lowers the efficiency of entropy coding.

With regard to PSNR performance, an approach is preferable that allows the decoder to deduce the positions in such a case without the need for additional symbols. This requires some preprocessing and thus has an impact on coding time. In our test setup, which is aimed at gains in coding efficiency and not at gains in coding time, we chose the following approach that is performed by encoder and decoder alike prior to the actual encoding process:

- Positions of coefficients with children fulfilling (5.9) are marked recursively. This is done by applying the following recursively, starting in the approximation subband.

  For a coefficient P_i:

  - If P_i is not a leaf node, process all its children $C_{i,j}$ first.
  - If the marking process returned true for any of the children, return *true* otherwise return *false*. Further, if P_i is not contained in the bitmask associated with its subband, mark P_i
  - If P_i is a leaf node, return *true* if P_i is significant, *false* otherwise.

- In the encoding process these marks are used to determine the children of which positions have to be processed even though the positions themselves are not contained in the bitmask. Note that no information regarding the marked coefficients is encoded at all.

- In the decoding process the marks are used to determine positions that have significant children, but are not contained in the bitmask (and thus not in the bitstream) themselves.

- In both, encoding and decoding the marks are not needed for the actual transformation.

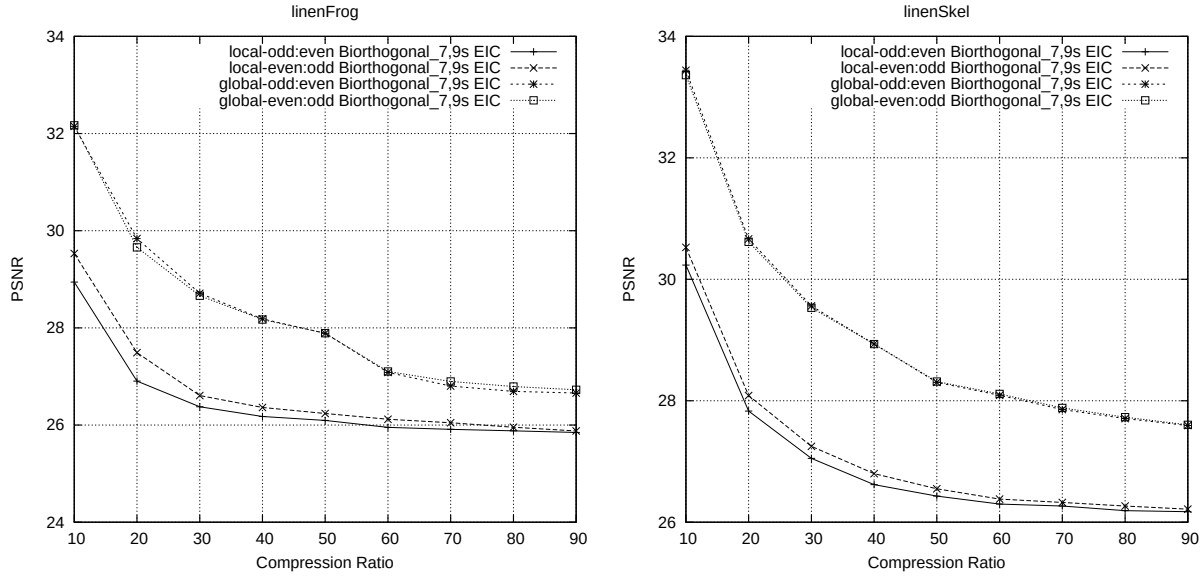
5.6.2 Impact on Downscaling Strategy on Efficiency of ZTC

According to the zerotree hypothesis [33], the magnitude and with it the importance of wavelet coefficients is indirectly proportional to the frequency, i.e. coefficients in subbands of low frequency are important with a higher probability than coefficients in subbands of high frequency. So a higher amount of coefficients in the lowpass subbands has a good chance of producing a gain in zerotree coding (cf. [20]).

With the choice of subsampling mode and strategy the number of coefficients in lowpass and highpass portions of a decomposition can be influenced to a certain degree (see sec. 4.3.2). Using *local* subsampling mode in conjunction with *even-odd* subsampling strategy guarantees that the number of coefficient in the lowpass subband are equal or (one) more than in the highpass subband for one step of 1D-decomposition.

While this is an intriguing approach on a theoretical level, in practice *local* subsampling does have a problem with artificial high frequency components as described in sec. 4.3.2. In our testruns, a gain for zerotree processing through *local* subsampling could only be achieved for a limited set of shapes, where lines as well as columns had a pairwise offset of $2i$, $i \in \mathbb{N}$.

5.6.3 Results



(a) Results for fig. 4.15.a

(b) Results for fig. 4.15.b

Figure 5.5: Comparison of subsampling modes and strategies

For a more general class of shapes, *global* subsampling always outperforms *local* subsampling. As can be seen in fig. 5.5, for the *global* subsampling mode, both subsampling

strategies, *even-odd* and *odd-even*, perform about equally well. It is interesting to observe that in the case of *linenFrog*, *odd-even* performs slightly better for the lower compression ratios, whereas *even-odd* performs better for higher compression ratios. A possible explanation for this phenomenon is that a favor for zerotree coding is achieved in the higher compression ratios, because the majority of the few significant coefficients happen to be positioned in the lowpass subbands with *even-odd*. In turn, for lower compression ratios, more significant coefficients are located in the lowpass subbands with *odd-even*.

In the case of *local* subsampling, *even-odd* outperforms *odd-even* in all testruns. This is due to the fact that in *local* subsampling mode using subsampling strategy *even-odd*, there are never less coefficients in the lowpass subband than there are in the highpass subband and this circumstance favours zerotree coding (cf. sec. 4.3.4).

Chapter 6

Adaptive Objectbased Image Compression

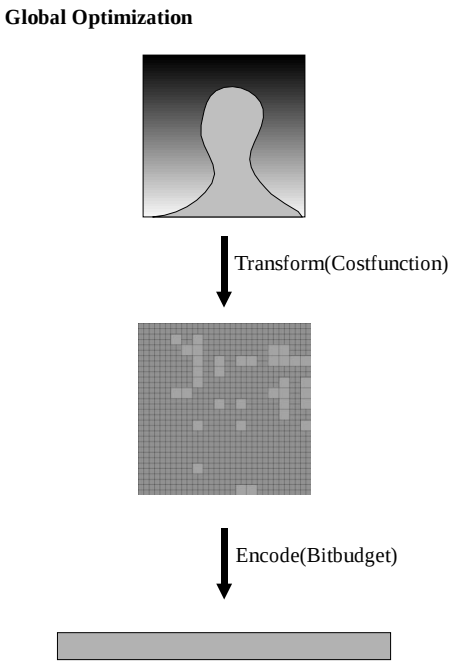
6.1 Introduction

Taking the superior performance of the adaptive WPT for oscillatory patterns, transferring objectbased adaptive coding to a non-objectbased framework seems a rewarding approach. The basic idea is segmenting an image into objects on distinction of frequential patterns and then transforming each object on its own. Thus, the wavelet bases are adapted to each object separately instead of the image as a whole.

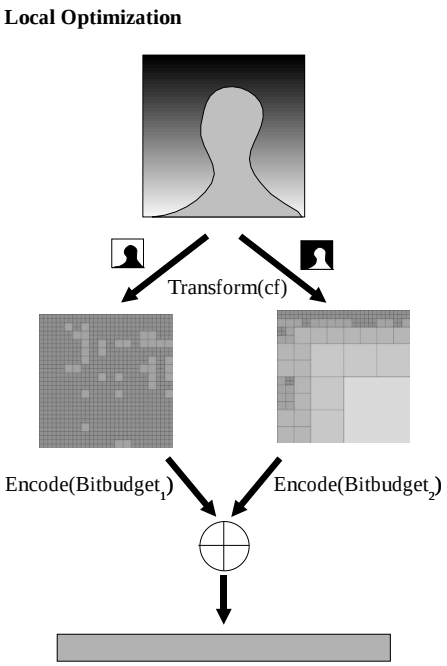
In the following, we describe the test setup that implements this idea. The starting point will be a simplified preliminary test setup which will serve to illustrate the objectbased adaptive approach. We will then present several improvements and alterations, the results for which we will discuss concertedly in sec. 6.4 to draw direct comparisons between the approaches.

We will not deal with the segmentation, but assume an image to be already segmented into objects of different frequential patterns. In fact, we construct test images fitting the above requirements to test whether the adaptive objectbased approach is able to produce better compression results and whether segmentation on the criteria of frequential properties would make sense in the first place. The used testimages are constructed from the previously tested textures for which the adaptive WP-decomposition produced superior results compared to the pyramidal decomposition.

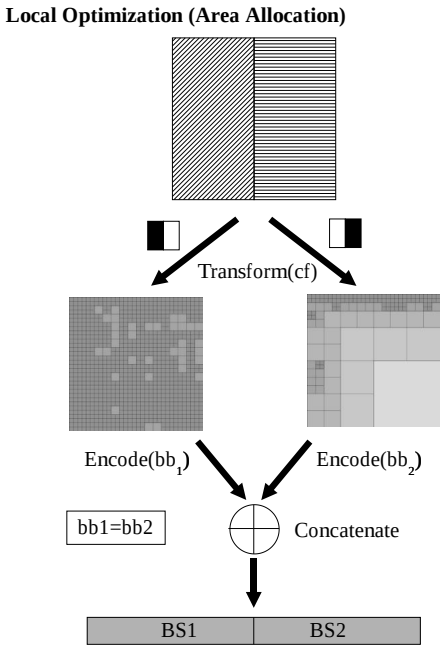
A schematic comparison between global, i.e. non-objectbased, adaptivity, which we will call "global optimization", and objectbased adaptivity, which we will call "local optimization", is shown in figures 6.1.a and 6.1.b. In the former case, the costfunction is used on the image as a whole, which leads to a single decomposition. The transform coefficients are then encoded and written to the target bitstream. Opposed to this, local optimization does not work on the image as a whole, but optimizes the information cost for each object individually. It is not even necessary to use the same costfunction for all objects. Thus, a decomposition structure is produced for each object in the image. A challenging question is how to distribute the target bitbudget between the objects. The choice of distribution has considerable impact on the coding performance, as we will see.



(a) Global Optimization



(b) Local Optimization



(c) Simplified Setup

Figure 6.1: Adaptive objectbased image coding

Based on the allocated bitbudgets, individual bitstreams are produced for the objects, which then have to be merged in some way, producing the final bitstream. There is a range of possibilities how to perform the merging operation, ranging from simple concatenation to more sophisticated methods for producing an embedded bitstream or including ROI coding.

6.2 Simplified setup

Obviously, the parameter space is vast, due to the different possibilities, even in the basic steps of the approach. Fig. 6.1.c shows a simplified test setup that we use to restrain the parameter space in an initial test environment (cf. [37]). There are only two objects in the image, of same size and rectangular shape. Transformation is done using the same costfunction for both of the objects and the bitbudget is distributed proportionally to the size of each object, i.e. as both objects have the same size, they are both granted the same bitbudget. We refer to this method as *area allocation*. The resulting bitstream contains both object-bitstreams sequentially, i.e. the merging operation is merely concatenation.

6.2.1 Results

The adaptive approach using the simple method of area allocation for the distribution of the bitbudget produces satisfactory results for images with objects of similar energy levels and high frequential components in different (ideally orthogonal) directions. Fig. 6.2.a shows a testimage produced from the artificial texture used previously. For this image, for most of the bitrates, a decomposition is produced by the objectbased BBA that outperforms the globally adaptive BBA as shown in fig. 6.3.a. Only for very low bitrates, both methods are tied. It is also noteworthy that the pyramidal decomposition is outperformed by the adaptive approaches for the whole bitrange considered.

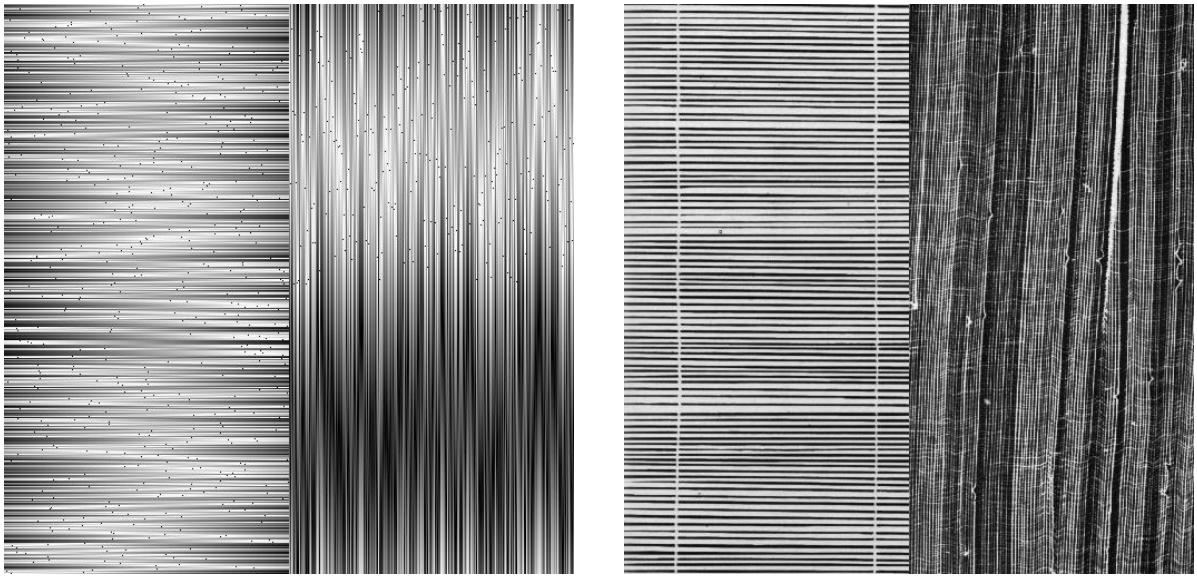
This is also the case for the testimage shown in fig. 6.2.b. *D49XD105* has similar frequential features as *artXart90*. However, the results for the globally and locally adaptive method are much closer here and the objectbased methods are outperformed for very low bitrates as shown in fig. 6.3.b. This is due to the different levels of energy in the two parts of the image – area allocation is suboptimal in this case.

However, the results of the tests with rectangular objects are promising enough to move on to arbitrary shapes. We will not use area allocation for the testruns with arbitrary shapes anymore, but present superior methods of bitbudget allocation.

6.3 Improvements

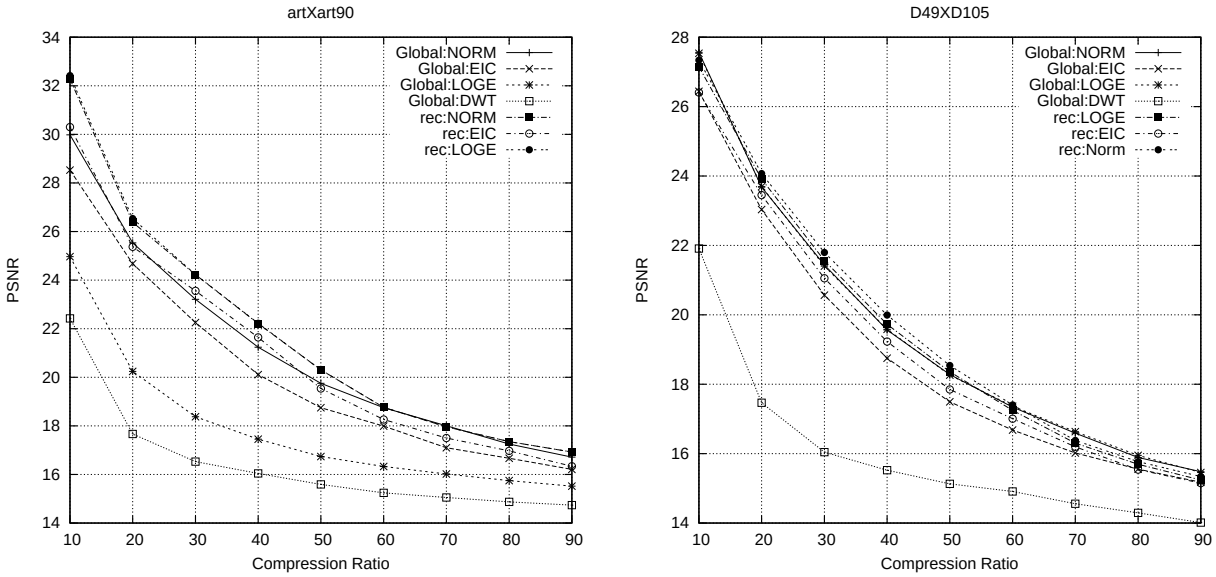
6.3.1 Bitbudget allocation

As far as PSNR performance is concerned, a good distribution of the available bitbudget should take the rate-distortion characteristics of each object into account and perform a “cost-benefit analysis”. For some objects it is possible to concentrate the energy in



(a) Artificial test texture: *artXart90* (b) *D49XD105*

Figure 6.2: Testimages in simplified setup



(a) Results for fig. 6.2.a (b) Results for fig. 6.2.b

Figure 6.3: Results for fig. 6.2 using *area allocation*

just a few transform coefficients, i.e. the object is well suited for compression. Each of these coefficients adds greatly to the quality of the object when transmitted, but further coefficients do not raise the PSNR in a considerable degree anymore. For other objects, the set of coefficients that carry the bulk of the energy is much bigger, and only after a lot of these coefficients have been transmitted the same gain in PSNR is achieved as in the previous case. Therefore, a good method of bitbudget allocation will pay attention to what part an object actually needs of the entire bitbudget and how much is better spent on other objects.

As "area allocation" clearly is not able to address the need of balancing the bitbudget based on the level of energy of each object, we introduce other methods that are more suited for this task. On the one hand, we will investigate the suitability of costfunctions for bitbudget allocation and on the other hand, we will present an approach that takes the decomposition and coefficient trees of all objects in order to build a unified similarity tree that implicitly optimizes bitbudget allocation.

6.3.1.1 Bitbudget costfunctions (BBCF)

The use of costfunctions for bitbudget allocation is fairly straightforward – basically the bitbudget is assigned to each object proportionally to the value a certain costfunction produces for this particular object.

Let $\mathbf{O}_i,\ i = 0,\ldots,n-1$ denote the image objects with their associated object shapes $\mathbf{S}_i,\ i = 0,\ldots,n-1$. By applying a costfunction to each object $\mathbf{O}_i$, cost-values $c_i,\ i = 0,\ldots,n-1$ are produced. Depending on the measure that should be addressed, different costfunctions can be used. We will test the following costfunctions for bitbudget allocation:

- Variance

$$cost_{Variance}(\mathbf{O}_i) = \sum_{(j,k)\in\mathbf{S}_i} (O_i(j,k) - \mu(\mathbf{O}_i))^2 P(O_i(j,k)) \tag{6.1}$$

- Energy

$$cost_{Energy}(\mathbf{O}_i) = \sum_{(j,k)\in\mathbf{S}_i} \log^*(O_i(j,k)^2) \tag{6.2}$$

- Entropy

$$cost_{Entropy}(\mathbf{O}_i) = -\sum_{(j,k)\in\mathbf{S}_i} P(O_i(j,k)) \log_2^*(P(O_i(j,k))) \tag{6.3}$$

$\log^*(t)$ is defined as

$$\log^*(t) = \begin{cases} \log(t) & \text{for } t \neq 0 \\ 0 & \text{else,} \end{cases} \tag{6.4}$$

$\mu(\mathbf{O}_i)$ is defined as

$$\mu(\mathbf{O}_i) = \sum_{(j,k)\in\mathbf{S}_i} \frac{O(j,k)}{|\mathbf{O}_i|} \tag{6.5}$$

and $P(O_i(j,k))$ is defined as

$$P(O_i(j,k)) = \frac{H(O_i(j,k))}{|\mathbf{O}_i|} \tag{6.6}$$

where $H(O_i(j,k))$ is the histogram-function of $\mathbf{O}_i$ evaluated at position (j,k) and $|\mathbf{O}_i|$ is the size of $\mathbf{O}_i$, i.e. the number of positions contained in $\mathbf{S}_i$.

Let B be the total bitbudget for coding the entire image. B is split into bitbudgets $b_i,\ i = 0,\ldots,n-1$ to be assigned to the objects proportional to their respective costs

$$b_i = B \cdot c_i / \sum_{j=0}^{n-1} c_j. \tag{6.7}$$

In our testruns for BBCF, like in the simplified setup, the bitstreams are just concatenated, i.e. the final bitstream is not an embedded bitstream.

6.3.1.2 Multipackettree (MPT)

Let $\mathbf{O}_i,\ i = 0,\ldots,n-1$ again denote the image objects with their associated object shapes $\mathbf{S}_i,\ i = 0,\ldots,n-1$. Further, let D_i be the wavelet packet decomposition structure that is obtained by applying a BBA to $\mathbf{O}_i$. As discussed previously, trees are a common way of representing decomposition structures. We will thus also refer to D_i as "decomposition tree". In parallel to each structural tree D_i, a "coefficient tree" exists for each object (see sec. 2.4).

For the MPT-approach, a similarity tree V_i is created for each object O_i from D_i, $i = 0,\ldots,n-1$, using the algorithm discussed in section 5.5.2. The similarity trees are grouped to form a "unified similarity tree" U with multiple roots,

$$U = \bigcup_{i=0}^{n-1} V_i. \tag{6.8}$$

As discussed in sec. 5.5.2, SMAWZ is able to deal with multiple roots in the case of *separated* similarity trees (see 5.5.2). Hence, it can be used for coding the unified similarity tree as well. The only difference is that the coefficients are located in different coefficient trees as well. The ordering of the coefficients is done for all objects together. Thus, MPT implicitly produces a rate-distortion oriented allocation of the bitbudget to the various objects, while still maintaining an embedded bitstream.

6.3.2 Common Structure Coding (CSC)

The intersection of the decomposition trees of the various objects in an image is never empty. Especially objects whose characteristics in frequency space do not differ in a high degree have considerable overlap in their decompositions. The idea of CSC is to make use of this fact and to code as much of the objects together as possible in a first step. The individual differences of the objects from the common structure are coded as "refinements" in a second step. Fig. 6.4 illustrates the analysis process, which is divided into three major steps.

(1) The best basis algorithm is applied to each of the objects $\mathbf{O}_0, \ldots, \mathbf{O}_{n-1}$ in an image $\mathbf{I}$, producing decomposition structures $D_0, \ldots, D_{n-1}$. The intersection C of $D_0, \ldots, D_{n-1}$ is defined by the decomposition structure that contains all subbands that are part of each D_i, $i = 0, \ldots, n-1$.

(2) The original image is decomposed to the common structure C, producing a set of coefficients $\mathbf{T}_{\mathbf{I}}^{C}$.

(3) Working on the common set of coefficients $\mathbf{T}_{\mathbf{I}}^{C}$, each object $\mathbf{O}_i$ is decomposed from decomposition structure C to decomposition structure D_i, producing $\mathbf{T}_{\mathbf{O}_i}^{D_i}$.

The union of the object shapes used in $\mathbf{T}_{\mathbf{O}_i}^{D_i}$ does not always cover the whole area of $\mathbf{I}$, depending on the used subsampling mode and strategy. In order to remedy this, the object shapes in $\mathbf{T}_{\mathbf{O}_i}^{D_i}$ are enlarged[1] to cover the whole of $\mathbf{I}$.

Finally, the object coefficients $\mathbf{T}_{\mathbf{O}_i}^{D_i}$ are written to the resulting bitstream, along with the object structures $D_0, \ldots, D_{n-1}$. It is not necessary to write the common structure C to the final bitstream, as the decoder can calculate the intersection from $D_0, \ldots, D_{n-1}$.

The synthesis process is illustrated in fig. 6.5 and contains the following steps.

(1) Each object $\mathbf{O}_i$, $i = 0, \ldots, n-1$ is reconstructed up to the common structure C, using the enlarged object shape, producing a set of coefficients $\mathbf{T}_{\mathbf{O}_i}^{C}$.

(2) Creating the union $\bigcup_{i=0}^{n-1} \mathbf{T}_{\mathbf{O}_i}^{C}$ yields $\mathbf{T}_{\mathbf{I}}^{C}$.

(3) The reconstructed image $\hat{\mathbf{I}}$ is produced by reconstructing $\mathbf{T}_{\mathbf{I}}^{C}$.

6.4 Results

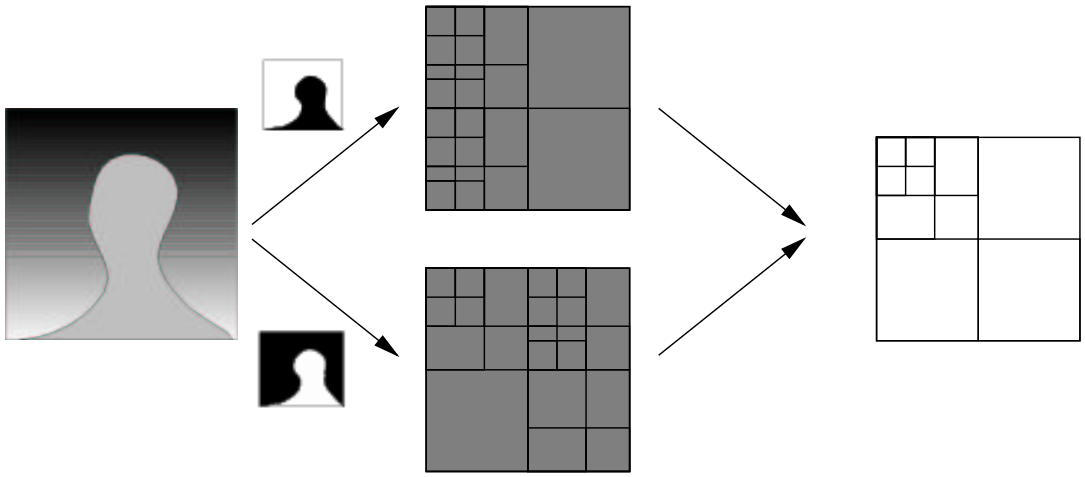
For bitbudget costfunctions, as with the additive costfunctions used in BBA, there is no causal link between the measure the BBCF addresses and actual PSNR performance. Thus, as can be seen in fig. 6.11 and fig. 6.12, there is not just one costfunction that always guarantees to distribute the bitbudget in an optimal way as regards PSNR performance, but the performance depends on the features of the testimages and how well they can be described using a certain measure.

For most of the tested images, MPT is about tied with the best-performing bit-budget costfunction (see fig. 6.11.a, 6.11.b, 6.12.a). But not only does MPT avoid the evaluation of a costfunction, for some images it even outperforms the most successful BBCF-approach, as can be seen in fig. 6.12.b.

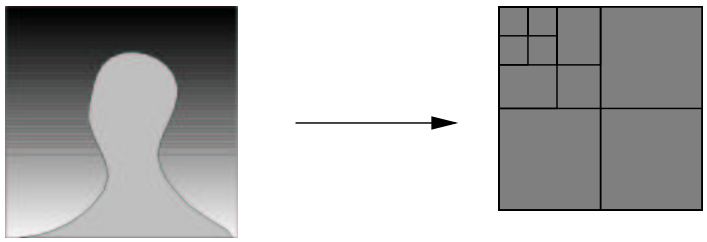
As regards the CSC-approach, as expected and shown in fig. 6.13 and fig. 6.14, the method that enlarges the object-shapes in the common structure to fill the respective

[1] In the case of two objects this can be easily done by inverting one of the bitmasks in all subbands. Thus, in our testresults this method is labelled *Invert*, as compared to *NoInvert*.

1. Find common structure



2. Decompose image to common structure



3. Using the common coefficients, transform each object

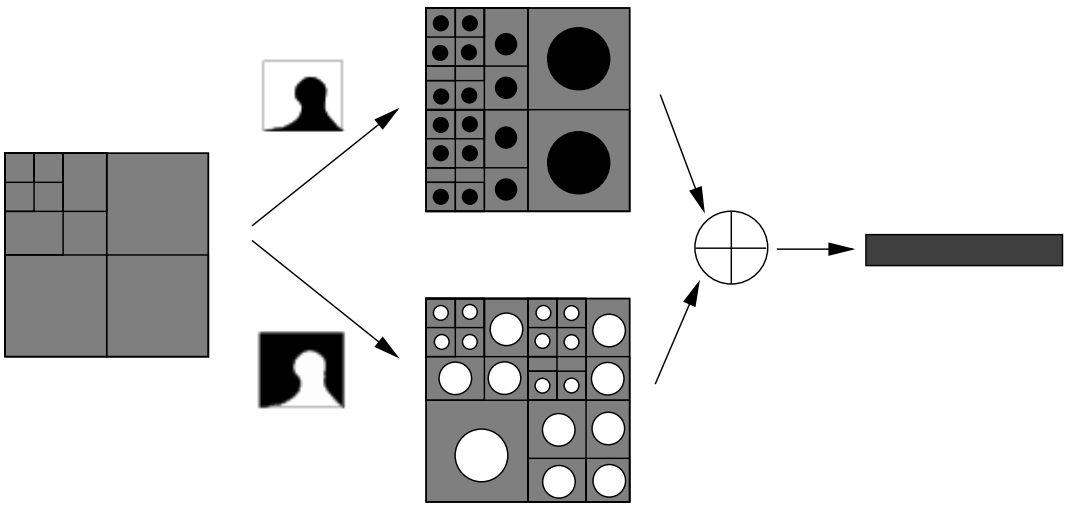


Figure 6.4: Common Structure Coding - Analysis

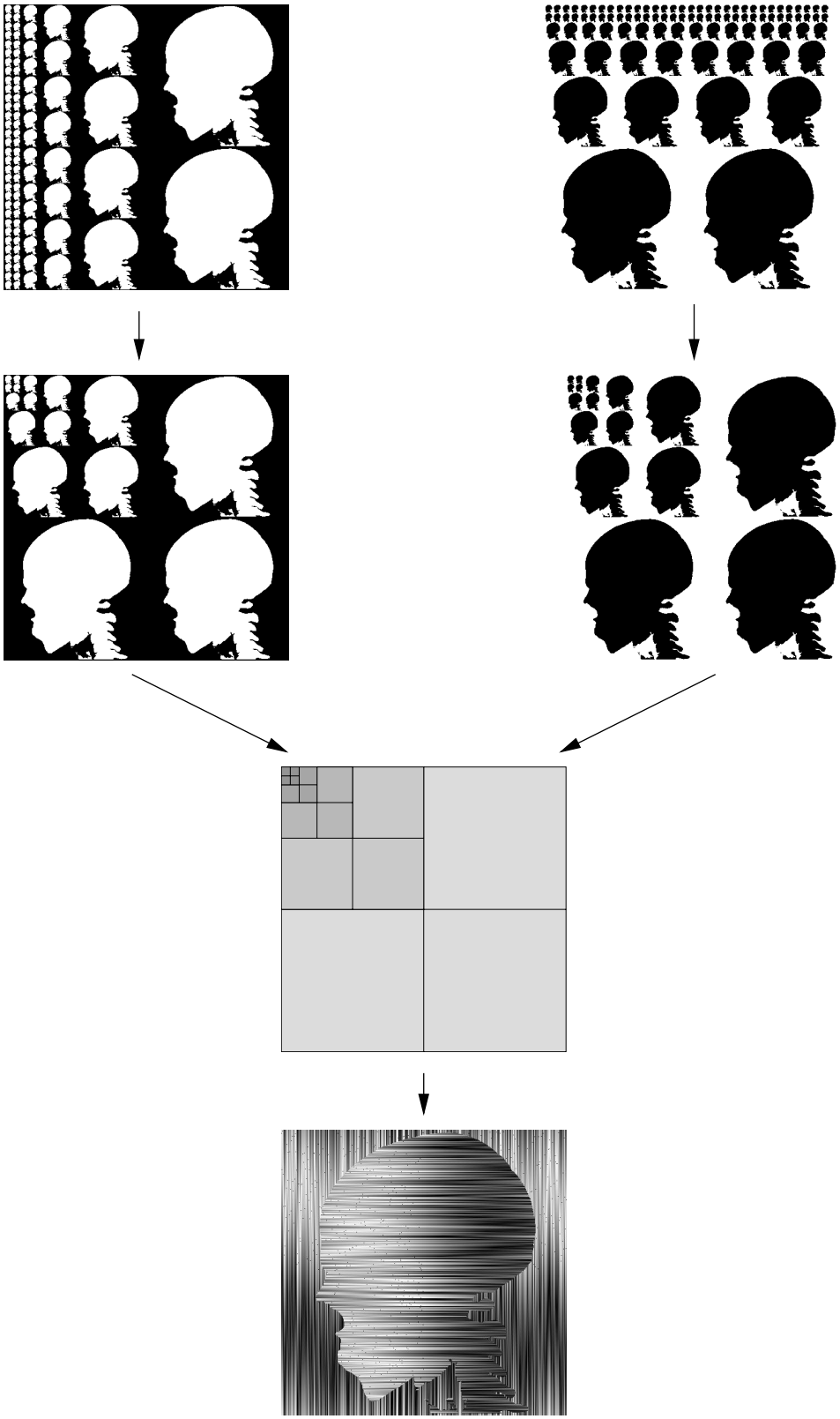
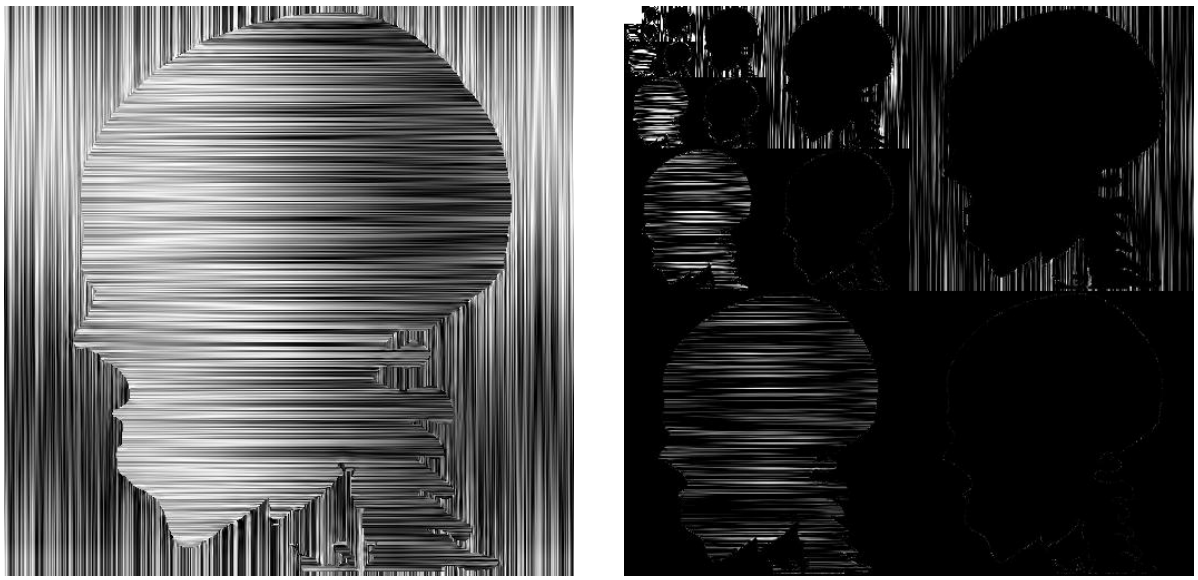


Figure 6.5: Common Structure Coding - Synthesis

subbands completely (*Invert*) performs better than the approach in which the coefficients are just left out (*NoInvert*). However, CSC does not give any gain in PSNR-performance. On the contrary, the PSNR values are generally lower than without CSC. This may be due to the (necessary) enlarging of the object-shapes in the transform domain. But even though this problem could be solved by arranging the coefficients during the subsampling process, the possible gain in coding performance would hardly be worth the considerable overhead that is necessary to find the common structure.

In fig. 6.15 and fig. 6.16 we compare the performance of MPT to the approach using *Ganesh++* with global optimization and to JPEG-2000. As expected, for all of the tested images, *Ganesh++* using pyramidal decomposition and JPEG-2000 are both outperformed by the adaptive methods. However, the difference between the globally adaptive and approach and the objectbased adaptive approach are quite insubstantial for most of the images (see fig. 6.15.a, 6.16.a).

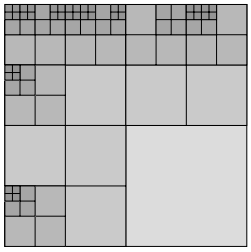


(a) *Artificial test image* (b) *Distribution of coefficients for (a)*

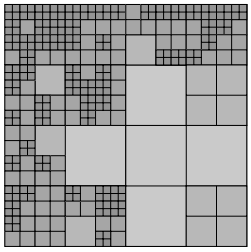
Figure 6.6: Distribution of coefficients in global optimization

A possible explanation is the fact that for these images the globally adaptive approach, due to the local preservation property of the wavelet transform, implicitly performs a segmentation in image space based on characteristics in frequency space. For our testimages, which are constructed from objects which exhibit these different characteristics, the coefficients for each object are almost disjointly divided into different frequency bands. This can best be illustrated by applying the algorithm to the artificial testimage shown in fig. 6.6.a, whose parts have high frequential patterns in orthogonal directions. Fig. 6.6.b shows the coefficient distribution over the subbands for the pyramidal decomposition[2].

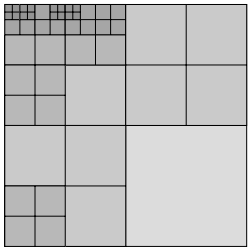
[2]The pyramidal decomposition obviously does not achieve high energy compaction, but is well suited



(a) Global Optimization



(b) Local Optimization - O_1



(c) Local Optimization - O_2

Figure 6.7: Decomposition structures for fig. 6.10.b

As can be seen, the two objects are completely separated into different areas, and it is obvious that a segmentation based on frequency characteristics prior to transformation will not yield much of an improvement.

However, global adaptivity does not outperform objectbased adaptivity for all testimages. In fig. 6.15.b, for example, the global approach is outperformed by more than 1db for most compression ratios. The global optimization in this case is not able to compact the energy as much as the objectbased approach. (The same is true for fig. 6.16, though to a lesser extent.) Looking at the decomposition structure of *poly_ linen X D50* in fig. 6.7, we can observe that the local optimization produced a much more complex decomposition for the foreground object than is contained in the most successful decomposition structure in global optimization.

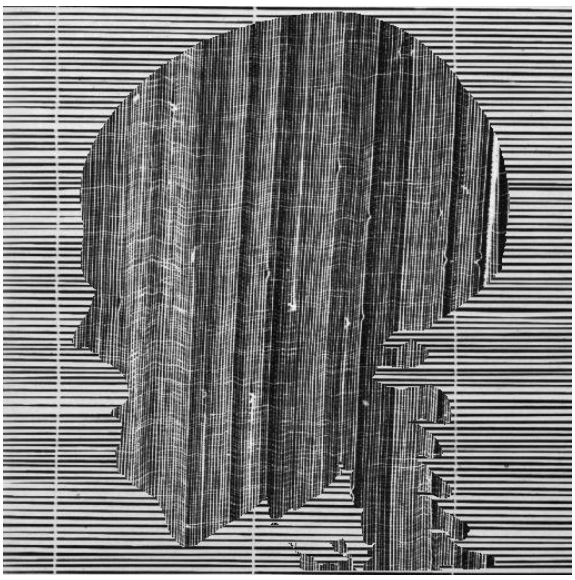
	Classical Wavelet	WPT Global Opt.	SA-WPT Objectbased
Steps	1. Decompose $O(N)$ 2. ZTC	1. Full Packet Tree $O(N \log N)$ 2. Best Basis 3. SMAWZ	1. For each object: (a) Full Packet Tree (b) Best Basis 3. SMAWZ
Complexity	low	medium	high
Area	most natural images	textures with high frequency components in a singular direction where the pyramidal decomposition fails to concentrate the energy in a few coefficients	only useful in special cases considerable overhead

Figure 6.8: Applicability of adaptivity in non-objectbased frameworks

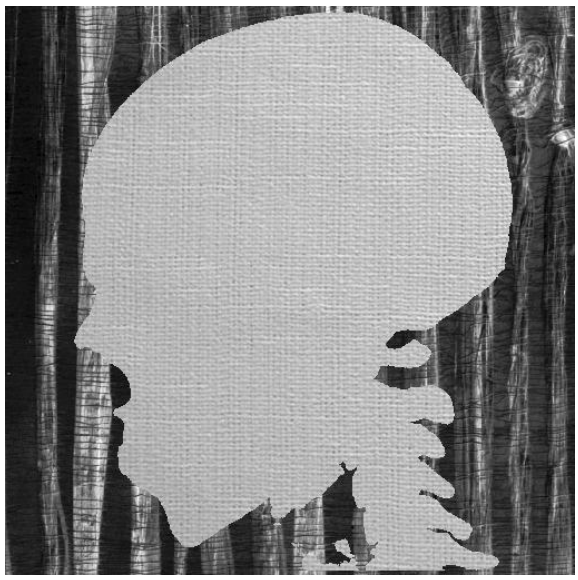
Concludingly, we can state that even though there are some special cases where the

to illustrate the point.

gain in PSNR-performance achieved by introducing objectbased adaptivity is substantial, the little (or no) gain that is produced in most cases will usually not justify the additional overhead. Fig. 6.8 sums up the applicability of adaptivity in non-objectbased frameworks, comparing non-adaptive, pyramidal wavelet transformation, WPT with global adaptivity and SA-WPT with objectbased adaptivity, i.e. *Motivation 2* (in the sense of sec. 1.2).

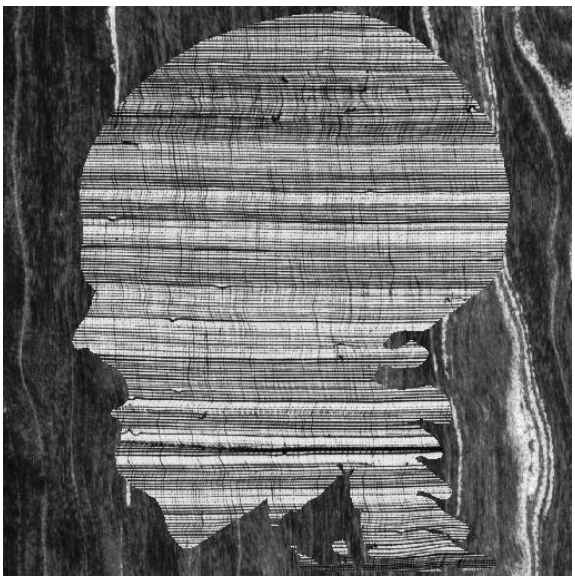


(a) *D105 X D49*



(b) *poly_ linen X D50*

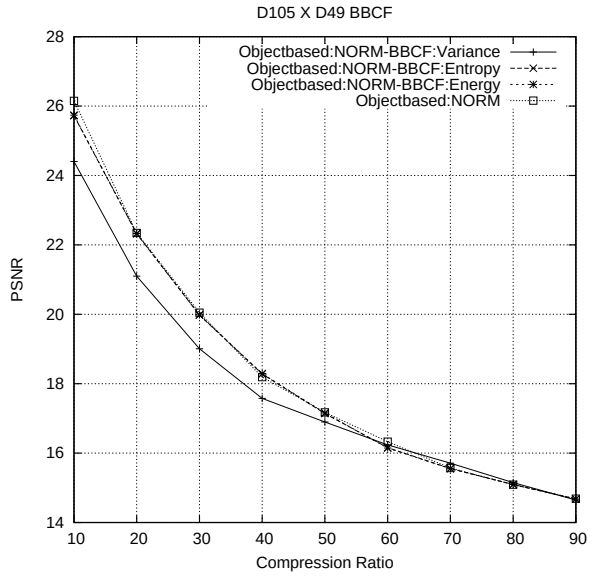
Figure 6.9: Test Images I



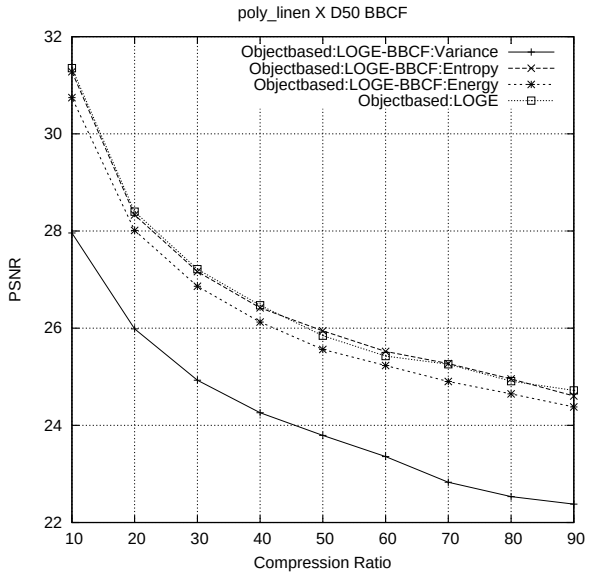
(a) *D106 X D70*

(b) *D51 X poly_ linen*

Figure 6.10: Test Images II

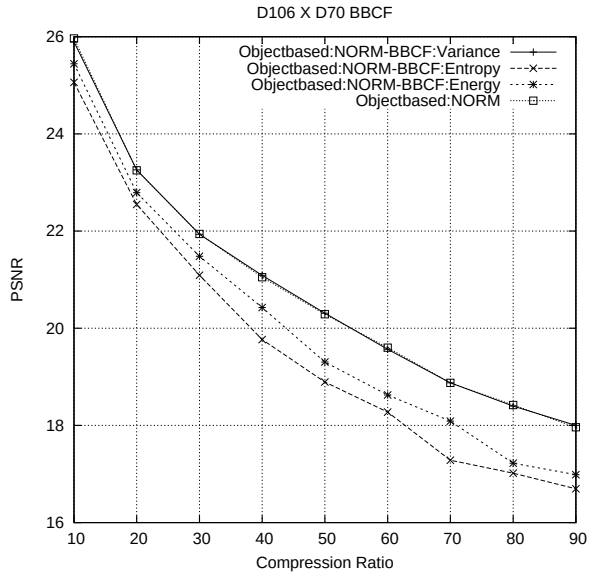


(a) BBCF: Results for fig. 6.9.a

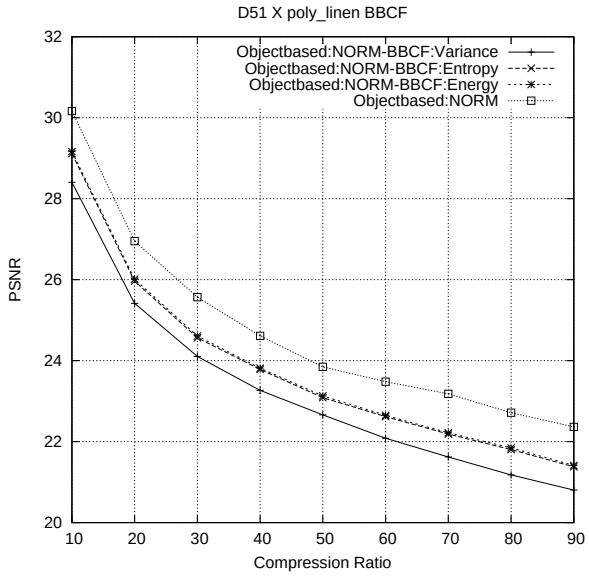


(b) BBCF: Results for fig. 6.9.b

Figure 6.11: BBCF: Results for fig. 4.14

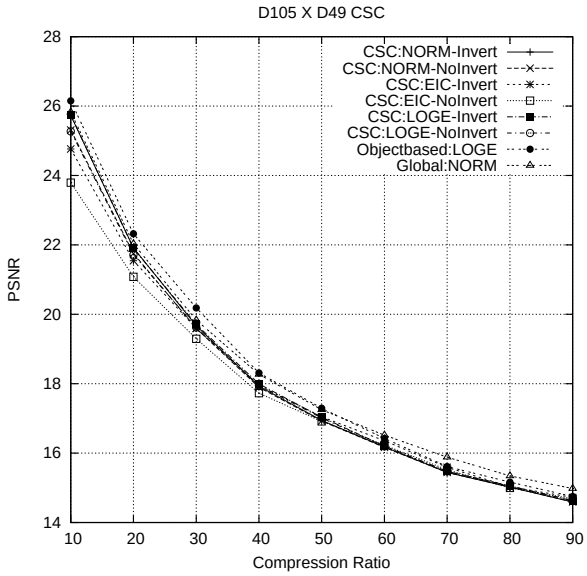


(a) BBCF: Results for fig. 6.10.a

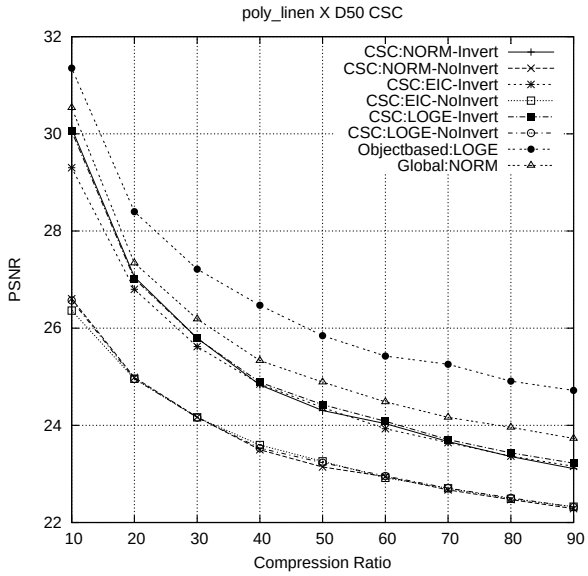


(b) BBCF: Results for fig. 6.10.b

Figure 6.12: BBCF: Results for fig. 4.15

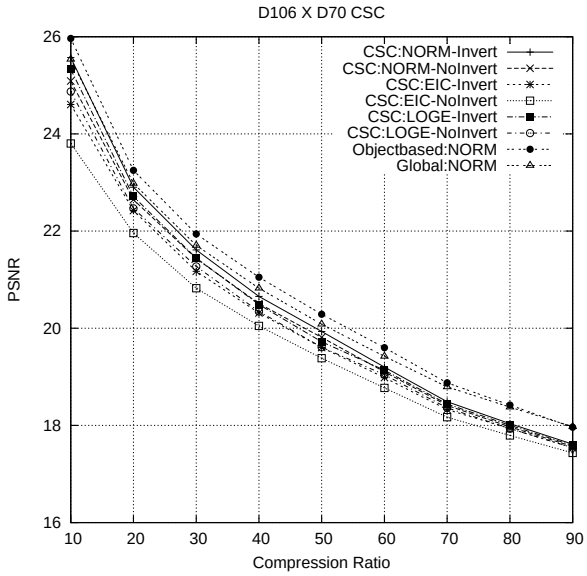


(a) CSC: Results for fig. 6.9.a

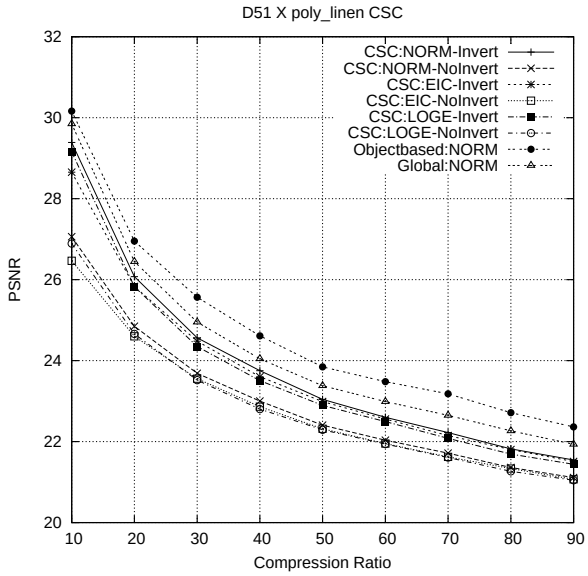


(b) CSC: Results for fig. 6.9.b

Figure 6.13: CSC: Results for fig. 4.14

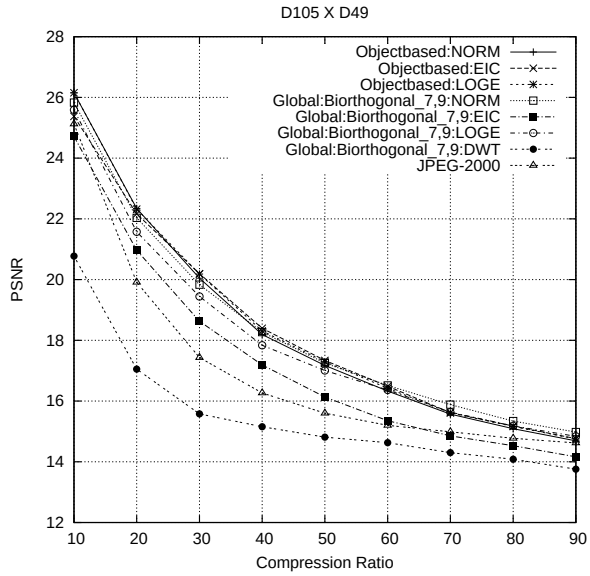


(a) CSC: Results for fig. 6.10.a

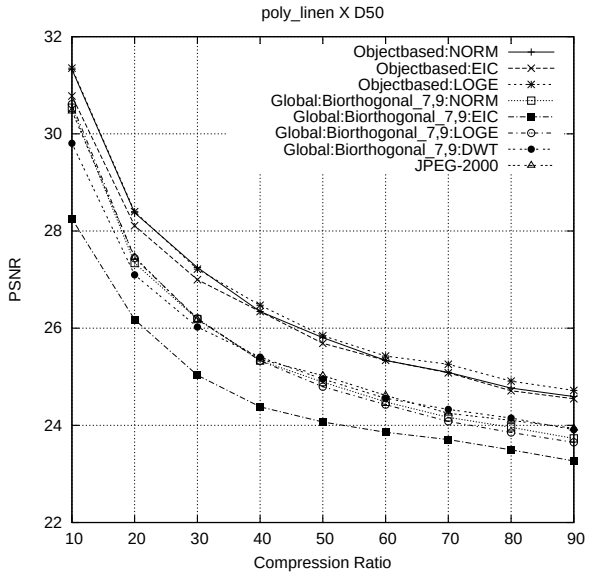


(b) CSC: Results for fig. 6.10.b

Figure 6.14: CSC: Results for fig. 4.15

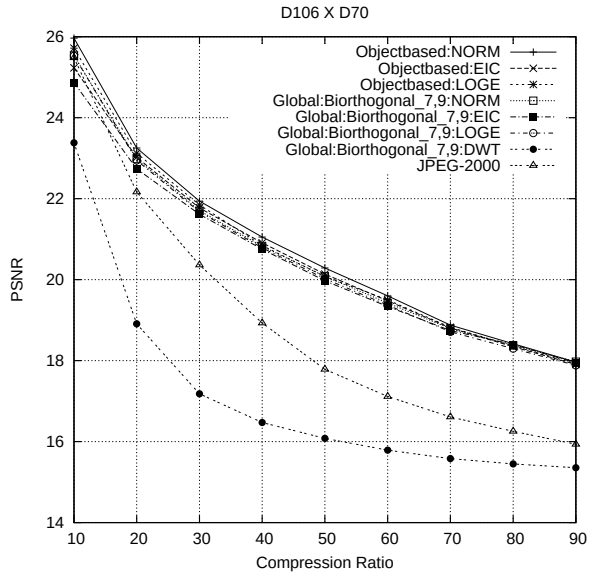


(a) MPT: Results for fig. 6.9.a

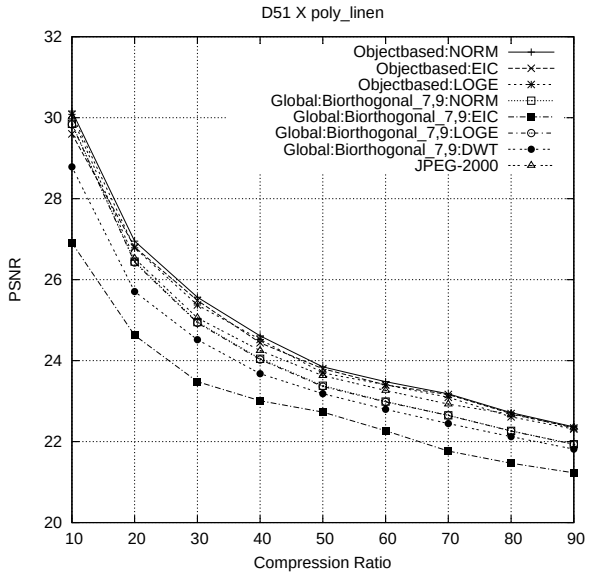


(b) MPT: Results for fig. 6.9.b

Figure 6.15: MPT: Results for fig. 4.14



(a) MPT: Results for fig. 6.10.a



(b) MPT: Results for fig. 6.10.b

Figure 6.16: MPT: Results for fig. 4.15

Chapter 7

Conclusion

The goal of this thesis was to investigate the usefulness of two motivations for objectbased adaptive image compression, i.e. *Motivation 1* and *Motivation 2* that we formulated in the introduction, for efficient image coding and increased functionality.

We presented the necessary foundations for this purpose and gradually developed a framework for our testruns. In our initial testresults the ability of the WPT in conjunction with BBAs to outperform the pyramidal wavelet transformation was shown. For a certain class of images, exhibiting strong oscillatory patterns, the WPT outperforms non-adaptive approaches, including JASPER, a reference implementation of the JPEG-2000 standard.

In the context of *Motivation 1*, it was shown that the objectbased adaptive approach using SA-WPT does not only provide an exact representation of the object and better functionality, but also increases PSNR-performance. By adapting the wavelet basis to the exact object, we have shown that not only the overhead for coding object shape and wavelet packet structure can be regained, but the non-adaptive approaches can be outperformed in compression efficiency. For the coding of object shapes, a method was shown that is especially well suited to work with the SA-WPT as it uses the same basic principles to represent the object shapes. This method produces a scalable bitstream containing exactly the object shapes needed by the SA-WPT.

It was shown that from the perspective of PSNR-performance, *Motivation 2*, i.e. artificially introducing objects in a non-objectbased environment, is not that as successful as *Motivation 1*. Even though gains in PSNR can be produced, they are generally too small to justify the overhead that is necessary for objectbased adaptivity. Global, i.e. non-objectbased, adaptivity has lower computational complexity and implicitly performs an object segmentation. Thus, it is preferable in most situations. For the other cases in which the computational overhead is of little or no importance, satisfactory results can be produced using the proposed improvements. For further improvement in the area, future work should address the issue of bitbudget allocation between the image objects.

In all of our discussion, our focus was rather on the feasibility of the used methods than on performance issues like execution time or memory requirements. In the areas where competitive results (with regard to PSNR) have been produced, future work should provide this focus and find methods to lower computational demand and execution time.

List of Figures

Bibliography

[1] M. Charrier, D. S. Cruz, and M. Larsson, "JPEG2000, the next millenium compression standard for still images," in *Proceedings of the IEEE International Conference on Multimedia & Computing Systems, ICMCS '99*, vol. 1, (Florence, Italy), pp. 131–132, June 1999.

[2] C. Christopoulos, A. N. Skodras, and T. Ebrahimi, "The JPEG2000 still image coding system: an overwiew," *IEEE Transactions on Consumer Electronics*, vol. 46, pp. 1103–1127, Nov. 2000.

[3] R. Koenen, "Overview of the MPEG-4 standard." ISO/IEC JTC1/SC29/WG11, July 1996.

[4] I. Sodagar, H. J. Lee, P. Hatrack, and Y.-Q. Zhang, "Scalable wavelet coding for synthetic/natural hybrid coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 9, no. 2, pp. 244–254, 1999.

[5] N. Ahmed, T. Natarajan, and K. Rao, "Discrete cosine transform," *IEEE Transactions on Computers*, vol. C-23, pp. 90–93, Jan. 1974.

[6] Z. Xiong, K. Ramchandran, M. T. Orchard, and Y.-Q. Zhang, "A comparative study of DCT- and wavelet-based image coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 9, pp. 692–695, Aug. 1999.

[7] T. Hopper, "Compression of gray-scale fingerprint images," in *Wavelet Applications* (H. Szu, ed.), vol. 2242 of *SPIE Proceedings*, pp. 180–187, 1994.

[8] I. Daubechies, *Ten Lectures on Wavelets*. No. 61 in CBMS-NSF Series in Applied Mathematics, Philadelphia, PA, USA: SIAM Press, 1992.

[9] C. Chui, *An Introduction to Wavelets*. San Diego: Academic Press, 1992.

[10] M. Wickerhauser, *Adapted wavelet analysis from theory to software*. Wellesley, Mass.: A.K. Peters, 1994.

[11] J. Benedetto and M. Frazier, eds., *Wavelets: Mathematics and Applications*. Boca Raton, Florida: CRC Press, 1994.

[12] S. Mallat, *A wavelet tour of signal processing*. Academic Press, 1997.

[13] P. Topiwala, ed., *Wavelet Image and Video Compression*. Boston: Kluwer Academic Publishers Group, 1998.

[14] D. Marpe, H. Cycon, and W. Li, "Complexity constrained best-basis wavelet packet algorithm for image compression," *IEE Proceedings Vision, Image, and Signal Processing*, vol. 145, no. 6, pp. 391–398, 1998.

[15] Z. Xiong, K. Ramchandran, and M. T. Orchard, "Wavelet packet image coding using space-frequency quantization," *IEEE Transactions on Image Processing*, vol. 7, pp. 892–898, June 1998.

[16] F. G. Meyer, A. Z. Averbuch, and J.-O. Strömberg, "Fast adaptive wavelet packet image compression," *IEEE Trans. on Image Process.*, vol. 9, pp. 792–800, May 2000.

[17] N. M. Rajpoot, R. G. Wilson, F. G. Meyer, and R. R. Coifman, "A new basis selection paradigm for wavelet packet image coding," in *Proceedings of the IEEE International Conference on Image Processing (ICIP'01)*, (Thessaloniki, Greece), pp. 816–819, Oct. 2001.

[18] P. Brodatz, *Textures: A Photographic Album for Artists and Designers*. New York: Dover Publications, 1966. Pictures downloaded from *http://www.ux.his.no/~tranden/brodatz.html* (Trygve Randen).

[19] H. Katata, N. Ito, T. Aono, and H. Kusao, "Object wavelet transform for coding of arbitrarily shaped image segments," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 7, no. 1, pp. 234–237, 1997.

[20] S. Li and W. Li, "Shape-adaptive discrete wavelet transform for arbitrarily shaped visual object coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 10, pp. 725–743, Aug. 2000.

[21] A. Signoroni and R. Leonardi, "Progressive ROI coding and diagnostic quality for medical image compression," in *Visual Communications and Image Processing '98* (S. Rajala and M. Rabbani, eds.), vol. 3309 of *SPIE Proceedings*, pp. 674–685, Jan. 1998.

[22] A. Kassim and L. Zhao, "Rate-scalable object-based wavelet codec with implicit shape coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 10, no. 7, pp. 1068–1079, 2000.

[23] H. Flatscher and A. Uhl, "oSPIHT - embedded object-based SPIHT image coding," in *Proceedings of the 2nd IEEE Region 8 - EURASIP Symposium on Image and Signal Processing and Analysis, ISPA '01* (S. Loncaric and H. Babic, eds.), (Pula, Croatia), pp. 593–598, June 2001.

[24] R. Kutil, "A significance map based adaptive wavelet zerotree codec (SMAWZ)," in *Media Processors 2002* (S. Panchanathan, V. Bove, and S. Sudharsanan, eds.), vol. 4674 of *SPIE Proceedings*, pp. 61–71, Jan. 2002.

[25] C. L. B. Jordan, S. K. Bhattacharjee, F. Bossen, F. Jordan, and T. Ebrahimi, "Shape representation and coding of visual objects in multimedia applications - an overview," *Annales des Telecommunications*, vol. 53, pp. 164–178, May 1998.

[26] H. Freeman, "On the encoding of arbitrary geometric configurations," *IRE Trans. Electronic Computers*, pp. 260–268, June 1961.

[27] H. Freeman, "Computer processing of line drawing images," *Computing Surveys*, vol. 6, pp. 57–97, March 1974.

[28] G. Langdon and J. Rissanen, "Compression of black-white images with arithmetic coding," *IEEE Transactions on Communications*, vol. COM-29, pp. 858–867, June 1981.

[29] I. Witten, R. Neal, and J. Cleary, "Arithmetic coding for data compression," *Communications of the ACM*, vol. 30, no. 6, pp. 520–540, 1987.

[30] F. Bossen and T. Ebrahimi, "A simple and efficient binary shape coding technique based on bitmap representation," *Proceedings of ICASSP 97*, 1997.

[31] S. Li and I. Sodagar, "Generic, scalable and efficient shape coding for visual texture objects in MPEG-4," in *Proceedings of the 2000 IEEE International Symposium on Circuits & Systems, ISCAS '00*, (Geneva, Switzerland), May 2000.

[32] O. Egger, P. Fleury, and T. Ebrahimi, "Shape-adaptive wavelet transform for zerotree-coding," in *Proceedings of the European Workshop on Image Analysis and Coding for TV, HDTV and Multimedia Application*, (Rennes, France), pp. 201–208, Feb. 1996.

[33] J. M. Shapiro, "Embedded image coding using zerotrees of wavelet coefficients," *IEEE Trans. on Signal Process.*, vol. 41, pp. 3445–3462, Dec. 1993.

[34] 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*, vol. 6, pp. 243–249, June 1996.

[35] N. M. Rajpoot, F. G. Meyer, R. G. Wilson, and R. R. Coifman, "On zerotree quantization for embedded wavelet packet image coding," in *Proceedings of the IEEE International Conference on Image Processing (ICIP'99)*, (Kobe, Japan), Oct. 1999.

[36] N. M. Rajpoot, F. G. Meyer, and R. R. Coifman, "Wavelet packet image coding using compatible zerotree quantization," tech. rep., Yale University, New Haven, 1999.

[37] D. Engel and A. Uhl, "Adaptive object-based image compression using wavelet packets," in *Proceedings of the 4th International Symposium on Video/Image Processing and Multimedia Communications (VIPromCom 2002)* (M. Grgic, ed.), pp. 183–187, 2002.