An efficient wavelet representation for large medical image stacks
An efficient wavelet representation for large medical image stacks
An efficient wavelet representation for large medical image stacks

Daniel Forsberg

Like the rest of the society modern health care has to deal with the ever increasing information flow. Imaging modalities such as CT, MRI, US, SPECT and PET just keep producing more and more image data. Especially CT and MRI and their 3D image stacks cause problems in terms of how to efficiently handle these data sets of images. Usually a PACS is used to manage storage and display of medical images. Since a PACS often is implemented with a server-client setup, the management of these large data sets requires an efficient representation of medical image stacks that minimizes the amount of data transmitted between server and client and that efficiently supports the workflow of a practitioner.

In this thesis an efficient wavelet representation for large medical image stacks is proposed for the use in a PACS. The representation supports features such as lossless viewing, random access, ROI-viewing, scalable resolution, thick slab viewing and progressive transmission. All of these features are believed to be essential to form an efficient tool for navigation and reconstruction of an image stack.

The proposed wavelet representation has also been implemented and found to be better in terms of memory allocation and amount of data transmitted between server and client when compared to prior solutions. Performance tests of the implementation has also shown the proposed wavelet representation to have a good computational performance.
Abstract

Like the rest of the society modern health care has to deal with the ever increasing information flow. Imaging modalities such as CT, MRI, US, SPECT and PET just keep producing more and more image data. Especially CT and MRI and their 3D image stacks cause problems in terms of how to efficiently handle these data sets of images. Usually a PACS is used to manage storage and display of medical images. Since a PACS often is implemented with a server-client setup, the management of these large data sets requires an efficient representation of medical image stacks that minimizes the amount of data transmitted between server and client and that efficiently supports the workflow of a practitioner.

In this thesis an efficient wavelet representation for large medical image stacks is proposed for the use in a PACS. The representation supports features such as lossless viewing, random access, ROI-viewing, scalable resolution, thick slab viewing and progressive transmission. All of these features are believed to be essential to form an efficient tool for navigation and reconstruction of an image stack.

The proposed wavelet representation has also been implemented and found to be better in terms of memory allocation and amount of data transmitted between server and client when compared to prior solutions. Performance tests of the implementation has also shown the proposed wavelet representation to have a good computational performance.
Acknowledgments

Since this thesis concludes my studies of Applied Physics and Electrical Engineering - International at Linköping University there are a number of people to whom I would like to express my gratitude but to name them all would be outside the scope of this text, ;-). Though there are some that I would like to acknowledge in a special way.

- Class Yi of year 2002, thanks for the fun time together.
- Göran Salerud, thanks for your involvement in all you do.
- PD RADIT Development at Sectra, thanks for answering my questions.
- Erik Sjöblom, thanks for your guidance throughout my whole thesis work.
- Harald Nautsch, thanks for your theoretical support.
- Robert Forchheimer, thanks for challenging me to believe in my own ideas.
- Emelie Forsberg, thanks for being a loving and supporting wife.
# Contents

1 Introduction  
1.1 Background ............................... 1  
1.2 Purpose .................................. 2  
1.3 Scope .................................... 2  

2 The problem domain  
2.1 The basic setup of a PACS .................. 3  
2.2 PACS handling large image stacks ........... 5  
2.3 Prior solutions ............................. 6  
2.4 Requirements for an efficient solution ....... 7  

3 Wavelet theory  
3.1 Time vs. frequency analysis ................ 9  
3.2 Wavelet analysis and synthesis ............... 10  
3.3 Multiresolution analysis ..................... 12  
3.4 The forward and the inverse wavelet transform .... 15  
3.5 Examples of wavelets ....................... 17  

4 Integer-to-integer wavelet transforms  
4.1 The lifting scheme .......................... 21  
4.2 Perfect reconstruction with finite precision .... 23  

5 Overview of wavelet-based codecs for medical image stacks  
5.1 2D codecs .................................. 27  
5.2 3D codecs ................................. 29  

6 Proposals for an efficient wavelet representation  
6.1 Wavelet packet structure #1 ................ 31  
6.2 Wavelet packet structure #2 ................ 33  

7 Implementation of wavelet packet structure #1  
7.1 Implementing wavelet packet structure #1 .... 41  
7.2 Blocking .................................. 44  
7.3 Reducing block artifacts ..................... 44  
7.4 Achieving fast wavelet transformations ....... 47
7.5 Organizing wavelet coefficients on disc .......................... 49
7.6 Handling the dynamic coefficient interval ........................ 49

8 Results of the proposed wavelet representation 53
  8.1 Memory allocation .............................................. 53
  8.2 Computation time .............................................. 54

9 Discussion, conclusion and future work 57
  9.1 Discussion ...................................................... 57
  9.2 Conclusion ..................................................... 60
  9.3 Future work ..................................................... 61

Bibliography .......................................................... 63
Chapter 1

Introduction

This chapter presents a short description of the background, the purpose and the scope of this thesis.

1.1 Background

The use of digital media as a medium for storage of data related to health care has been used since the latter half of the last century. The needed storage in the beginning was of rather modest size when compared to today’s vast amounts of data produced at a regular hospital. A great proportion of this data comes from imaging modalities. Many of these modalities produce so called image stacks. A single stack consists of a sequence of images, each representing a cross section (slice) of an examined patient. A CT examination can produce an image stack consisting of up to 6000 images with 1024*1024 pixels per image and 16 bits per pixel (bpp). An examination like this would result in a dataset requiring roughly 12 GB of storage. These huge amounts of data are handled by so called picture archiving and communication systems (PACS).

The normal setup for a PACS consists of a server-client system where all the data from the different image modalities is stored on the server. The server is connected to a LAN or/and a WAN to provide access to the data for practitioners working at different clients. Alongside with the rise of telemedicine web-based PACS are becoming more and more common.

The large amount of data from a single examination results in different problems as it is transmitted from the server to the client in order to be viewed by a practitioner. One of these problems is long waiting times for the practitioner in order to receive all data. This is due to poor bandwidth capacities. A second problem is that once all the data has been received the practitioner is faced with the task of navigating the huge amount of data in order to find the relevant images/slices that will assist the practitioner in making a correct diagnosis/decision. A third problem arises when the received images do not display the contained information in a preferable way.
1.2 Purpose

The purpose of this thesis is to explore the possibilities of using a wavelet representation as a way of efficiently representing an image stack in order to decrease the amount of data transmitted between the server and the client and to provide the practitioner with an efficient tool for navigating and reconstructing the data of an image stack.

1.3 Scope

The scope of this thesis covers a presentation of the problem domain, which includes the setup of a PACS and how a modern PACS supports the workflow of a practitioner whilst dealing with large image stacks. The basic theory for wavelets and integer-to-integer wavelet transforms alongside with an overview of some well known codecs from the literature are also presented. The main focus of this thesis is to investigate if there are methods that can accomplish an efficient algorithm for doing 3D wavelet analysis/synthesis of image stacks and to study different ways to organize wavelet coefficients on disc so that a number of access patterns (reconstructions) are possible and efficient. A proposed method should be able to handle large medical image stacks and to support the following features:

- Lossless viewing (perfect reconstruction)
- Arbitrary 2D viewing (random access of original images)
- Progressive transmission (data received by client should be reusable)
- Scalable resolution (ability to change resolution)
- Thick slab viewing (ability to view thicker slices)
- ROI-viewing (ability to cut out and view a certain region from the original image without transmitting the whole image)

These features are believed to be essential to form an efficient navigation and reconstruction tool for a medical image stack. The implementation of a proposed method is also of great relevance in this thesis. The implementation is to be done in C#. Important aspects for a feasible implementation are the amount of data transmitted between server and client, the computation time for analysis/synthesis at the server/client and the need for auxiliary memory allocation during computation.
Chapter 2

The problem domain

This chapter presents the basic setup of a PACS and the workflow of a modern PACS when dealing with large image stacks. They together form the description of the problem domain. Some prior solutions to this problem, which do not utilize wavelets, are discussed together with the requirements for an efficient solution. The setup outlined in section 2.1 is based upon [1].

2.1 The basic setup of a PACS

Today many hospitals possess to their aid in diagnosis different imaging modalities such as computed tomography (CT), ultrasound (US), magnetic resonance imaging (MRI), digital radiography (DR), nuclear medicine (NM) and projectional radiography. Images from these modalities are either digital or analog and can therefore be stored on a digital or an analog medium. The range of storage media is vast and consists of video tapes, films, DVDs, CDs, magnetic discs and optical discs. A simple but inefficient solution for image storage and retrieval would be to store all images on these previously mentioned types of media in an archive and then just collect the proper storage medium whilst retrieving some images. For obvious reasons this is not how archiving and retrieving images are done today, instead a computer-based system, known as a PACS, is used.

The first implementation of a PACS saw its light in the beginning of the eighties. A great challenge for the first systems was the connectivity and interoperability between a PACS and individual imaging modalities. Since the introduction of the Digital Imaging and Communications in Medicine (DICOM) standards these problems of the early PACS have been resolved.

A PACS comprises three major subsystems: acquisition, archive and display. The acquisition system consists of a number of computers that handle the importation of the images from the individual imaging modalities into the PACS. The archive system consists of a host computer with mass storage devices for archiving the images to support future retrieval. The display system comprises different workstations, each consisting of a computer with a display device, often a high-resolution monitor, to support the practitioners whilst displaying and
manipulating the images in order to make a correct diagnosis/decision.

The archive system or server, which is the main system within a PACS, is a multitasking computer supporting multiple processes to be run simultaneously. Thus the archive server is configured with dual or multiple CPUs, high-capacity RAM and a high-speed network interface. The server runs an image management software to handle major functions such as:

- Image receiving
- Image routing
- Image stacking
- Image archiving
- Database updating
- Image retrieving

The archive server uses a database to store necessary information in order to support the server in the above mentioned functions. Closely linked to the server is the storage subsystem that often consists of storage devices on different levels, e.g., short-term storage with high speed discs for caching and slower long-term storage for archiving.

For a PACS to operate successfully it must have an interface to a hospital information system (HIS) and/or a radiology information system (RIS). Information exchanged between PACS and HIS/RIS helps the server to perform various tasks such as image routing, stacking and retrieving.

In order for a PACS to operate satisfactory there are certain aspects that must be met:
2.2 PACS handling large image stacks

In the previous section we stated that the efficiency of the system is one of the most crucial aspects of a modern PACS. Efficiency can be measured by how well and how fast a PACS supports the workflow of a practitioner. The efficiency becomes especially obvious whilst dealing with very large image stacks and considering the required high pace at a radiology department. Due to today’s constant increase of information (read number of images) per patient the practitioner has to view more and more images in the same time in order to keep up the work pace.

The traditional way of viewing images from a CT modality is to view each original image. This method is unacceptable with today’s CT modalities due to the small slice thickness (for example 0.625 or 1.25 mm) and the large number of images that can be produced at a normal examination (500-1000). The thin slices render images that have a large amount of noise and are therefore unpleasant for viewing. The large number of images that is produced presents the practitioner with the problem of dealing with very large image stacks to cine through in order to find the images that will assist him/her whilst making a correct diagnosis. A common method to avoid these problems is to reconstruct images with a larger slice thickness (say 5 mm) at the imaging modality before they are imported to the PACS. Both the noise and the number of images are reduced due to the averaging that takes place. Traditionally all of this is done in the transversal plane.

Viewing slices of the transversal plane is however not the only way to view data from an examination. The image modalities offer the possibility to reconstruct slices of the sagittal, of the coronal or even of an arbitrary plane. Many modalities can also produce 2D snapshots of 3D reconstructions at different angles which can be utilized for viewing. These different ways of viewing an image stack can aid the practitioner in his/her work. All of this is a great resource but we must remember that all of these reconstructions take place at the imaging modality before the images are imported to the PACS.

Based upon this there are two obvious workflows for dealing with large image stacks.

• After the examination the imaging modality reconstructs the normally used transversal slices, which are imported to the PACS and used for viewing by

---

1The transversal plane \((x \text{ and } y)\) divides the body into upper and lower, the sagittal plane \((y \text{ and } z)\) into right and left and the coronal plane \((x \text{ and } z)\) into front and back.
the practitioner. If the practitioner finds the transversal images insufficient or inadequate he/she can request other reconstructions to be performed at the imaging modality and imported to the PACS for viewing.

- After the examination some different but not all reconstructions are performed (e.g. transversal, sagittal and coronal slices) and imported to the PACS. The practitioner can then use the preferred slices in his/her work.

Both of these workflows result in problems.

The first workflow results in time issues since the practitioner must first view the transversal images and if they do not adequately support the practitioner some other reconstructions has to be requested. This will introduce a time delay, since another set of slices has to be reconstructed and imported. Also there is a possibility of not exactly reconstructing the slices that the practitioner has requested, since it is very likely that someone else than the practitioner himself/herself will perform the reconstructions. Important to remember is also the fact that the data from a CT examination is not stored on the imaging modality forever.

The second workflow will result in memory issues. Often it will be sufficient with the transversal reconstructions, thus the other reconstructions only cause storage problems. Equally important is the fact that the already performed reconstructions might not contain the one that the practitioner needs but he/she has to request another reconstruction set and we are back to the problem of the first workflow but with additional data to store.

In order to solve these problems many PACS today offer the practitioner, while working at the workstation, the possibility to do sub sampling, orthogonal projections, multi-planar reconstruction (MPR) and 3D volume rendering (VR) from the transversal image stacks that have been imported to the PACS. The problem is that when dealing with large image stacks the amount of data considerably increases the computation time and the memory allocation. Thus it becomes practically impossible to use these features at a high pace radiology department.

### 2.3 Prior solutions

From the last section we can conclude that a preferable solution would be where the practitioner whilst at the workstation can by himself/herself choose how to view an image stack. Since performing many different reconstructions at the imaging modality causes a manifold increase in the needed storage space and the possibility of still not having the preferred reconstruction available to the practitioner, we can conclude that the best solution would be if the PACS allows the practitioner to manipulate the stored image stack for visualization. PACS vendors have tried to overcome the problems with long computation times and large memory demands by creating multiresolution representations of the original image stack (often consisting of transversal slices) but without causing a manifold increase in needed storage space. There are two popular multiresolution representations that are used for this.

The first representation is achieved by repeatedly sub sampling the original images by a factor 2 and to store the resulting images. This will cause an increase
in the needed storage space by 33%. The second representation is also achieved by sub sampling but this time the whole stack is repeatedly sub sampled instead of just the images. This causes an increase in needed storage space by 12.5%. Both of these two representations are known as Gaussian pyramids.

![Figure 2.2. Repeatedly sub sampling each original image by a factor 2.](image1)

![Figure 2.3. Repeatedly sub sampling the original image stack by a factor 2.](image2)

The advantage of these two representations is that the sub sampled images or stacks can be used if the practitioner needs to view something with a lower resolution than the original data. Often this is the case when the practitioner needs to navigate in the image stack in order to find the right images to examine. Thus the server can provide the workstation with low resolution data to navigate in and then send high resolution data once the practitioner has found the right images to examine. Unfortunately there are some disadvantages. The first disadvantage being that already sent data can not be reused when displaying data with a higher resolution, although the low resolution data contains information that is contained in the high resolution data (note the difference between data and information). This causes a redundancy of sent information between server and client. Another disadvantage is the fact that if the practitioner wishes to view images with another resolution than provided by the multiresolution representation stored on the server, than the server has to send all data at the next higher resolution level to the client which then has to process the data. For instance, if the practitioner wishes to view images with a high planar resolution but with a low axial resolution (thick slab viewing), then all original data must be sent to the client which then averages the data to achieve the preferred resolution.

### 2.4 Requirements for an efficient solution

From all this we can conclude that an important tool for a PACS that adequately supports a practitioner in his/her work would be to find a representation of an image stack that supports the following features and still handle large image stacks gracefully:

- Lossless viewing
- Scalable resolution
- MPR
- Random access
The problem domain

- ROI-/VOI-viewing
- Progressive transmission
- Compression

Each one of these features is believed to be an important tool for supporting the practitioner. The ability of lossless viewing is important to make sure that no data is lost or, more correctly, that no information is lost that could be relevant for the diagnosis/decision. Perhaps this is foremost a legal requirement since few practitioners actually view and carefully examine all of their reconstructed images. Scalable resolution is an important tool to provide the practitioner with thick slab viewing (viewing thicker slices) and with the possibility to choose the desired resolution in all three dimensions. As we already have discussed, not only transversal but also sagittal, coronal and arbitrary reconstructed slices are relevant to the practitioner, thus MPR is important. Random access, ROI-viewing and VOI-viewing and progressive transmission should be understood in the sense that in order to view something of the stack, not all of the stack must be transmitted to the client and processed at the client. These features will decrease the amount of data that must be transmitted between server and client and thus improve the speed of the system. Compression of data is important to further increase the efficiency of the system by decreasing the amount of data that must be transmitted between server and client. It will also help in decreasing needed storage space. Using compression comes though with a substantial computational cost as we will discuss later in this thesis.

The observant reader will have noted that the above list with features is more comprehensive than the list given in section 1.3, which forms the scope of this thesis.
Chapter 3

Wavelet theory

This chapter presents an introduction to wavelet analysis and synthesis for 1D signals. The 1D wavelet theory can easily be extended to n-D wavelet theory in the case of separable wavelets. The introduction in the following sections is to a great extent based upon a very accessible introduction found in [2].

3.1 Time vs. frequency analysis

Since Joseph Fourier proved that all periodic functions can be written as a series of cosine- and sine-functions, the Fourier transform has been a standard tool for analyzing signals. Given a function \( f(t) \) the Fourier transform \( F(\omega) \) can be found as

\[
F(\omega) = \mathcal{F}[f(t)] = \int_{-\infty}^{\infty} f(t)e^{-j\omega t}. 
\]

Although the Fourier transform provides a good tool for analysis, it has some rather disturbing disadvantages. This is due to the infinite support of the basis functions of the Fourier transform in the time domain, the cosine- and the sine-functions. For example, whilst analyzing a signal with the Fourier transform one can tell exactly how large the component of frequency 440 Hz is in the signal, but it is impossible to say where in the signal this component is present. Thus the Fourier analysis provides us with exact localization in frequency but none in time. The opposite is true for the time analysis of the signal where we have perfect localization in time but no frequency information. It is important to note that both \( f(t) \) and \( F(\omega) \) describe the same function, the only difference is the basis functions used to represent the signal.

A possible method to find out where a frequency component is present in a signal is provided by simply chopping the signal \( f(t) \) into pieces of equal length \( T \) and then applying the Fourier transform to each piece of the signal and to analyze the frequency content of each piece. This is also known as short-term Fourier analysis. It is no good just to simply cut the signal into pieces, because this will cause distortion in form of boundary effects. To avoid this problem we can use
windowing. If \( g(t) \) represents the window applied to the signal \( f(t) \), the short-term Fourier transform (STFT) is given as

\[
F(\omega, \tau) = \int_{-\infty}^{\infty} f(t)g(t - \tau)e^{-j\omega t}.
\]

Even though our analysis with the STFT provides us with better information regarding to where a frequency component is present in a signal we still have some obstacles to face. Say that a part of the signal contains a frequency of \( 1/t_0 \) Hz. Thus we would need a window of at least the size \( t_0 \) s to analyze it. Unfortunately this means that we cannot accurately localize components with higher frequencies in the signal. Our uncertainty will be proportional to the size of the window. Of course the opposite is valid again, with a short time window we can better localize high-frequency components but not low-frequency components. This is formalized in the uncertainty principle given as

\[
\sigma_t^2 \cdot \sigma_\omega^2 \geq \frac{1}{\sqrt{2}}
\]

where

\[
\sigma_t^2 = \frac{\int t^2|g(t)|^2dt}{\int |g(t)|^2dt} \quad \sigma_\omega^2 = \frac{\int \omega^2|g(\omega)|^2d\omega}{\int |g(\omega)|^2d\omega}
\]

The problem with the STFT is due to the fixed window length, but if we instead could find a function that scales and translates the window accordingly to the frequency and the part of the signal that we wish to analyze, than we might have a proper tool for signal analysis. Roughly speaking this is what is done in wavelet analysis.

### 3.2 Wavelet analysis and synthesis

In wavelet analysis we start with a function called the mother wavelet, \( \psi(t) \). This function can be scaled by substituting \( t \) with \( t/a \), where \( a \) controls the amount of scaling. In the same manner the function is translated by substituting \( t \) with \( t - b \). Important to note regarding the scaling is that it changes the norm of the function.

\[
\|\psi(t)\| = \int_{-\infty}^{\infty} \psi^2(t)dt
\]

\[
\|\psi(t/a)\| = \int_{-\infty}^{\infty} \psi^2(t/a)dt = a \int_{-\infty}^{\infty} \psi^2(t)dt = a\|\psi(t)\|
\]
3.2 Wavelet analysis and synthesis

By multiplying the scaled function with $1/\sqrt{a}$ we can still have the same norm as the mother wavelet. If the mother wavelet is given as $\psi(t)$ we can find a set of analyzing functions as

$$\psi_{a,b}(t) = \frac{1}{\sqrt{a}} \psi(t - \frac{b}{a}).$$

The decomposition/analysis of $f(t)$ into these wavelet functions can be obtained by computing the inner product

$$w_{a,b} = \langle f(t), \psi_{a,b}(t) \rangle = \int_{-\infty}^{\infty} f(t) \psi_{a,b}(t) dt.$$ (3.1)

The reconstruction/synthesis of the function $f(t)$ can be achieved by computing

$$f(t) = \frac{1}{C_\psi} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \frac{1}{a^2} w_{a,b} \psi_{a,b}(t) dadb$$ (3.2)

where

$$C_\psi = \int_{0}^{\infty} \frac{\Psi(\omega)^2}{\omega} d\omega$$ (3.3)

and

$$\Psi(\omega) = \mathcal{F}[\psi(t)].$$ (3.4)

As we might expect this decomposition is not valid for any arbitrary function $\psi(t)$ but only when certain conditions on $\psi(t)$ are fulfilled. One condition is that $C_\psi$ must be finite, thus we require the mother wavelet to have a zero mean, i.e. the mother wavelet must be oscillatory (hence the name wavelet). This can easily be verified by observing equation 3.4 that if $\Psi(0) \neq 0$ than there is a singularity in the integrand of equation 3.3.

For $a,b \in \mathbb{R}$ and $t \in \mathbb{R}$ equation 3.1 is known as the continuous wavelet transform (CWT). In this thesis though we are more interested in finding a discrete representation of equation 3.1. A popular choice is to choose $a$ and $b$ as

$$a = a_0^{-m}, \quad b = n b_0 a_0^{-m}, \quad m,n \in \mathbb{Z}.$$

Selecting $a_0$ as 2 and $b_0$ as 1 gives us the following set of wavelets.

$$\psi_{m,n}(t) = 2^{m/2} \psi(2^m t - n), \quad m,n \in \mathbb{Z}$$

An important question at this point is whether this set of functions, $\psi_{m,n}(t)$, is a complete set of basis functions for all functions $f(t) \in L^2(\mathbb{R})$? The answer to this question is yes but only under some conditions on $\psi(t)$\footnote{These conditions and the proofs for them are outside the scope of this thesis but can be found in [3], [4], [5].}. If our set $\psi_{m,n}(t)$ is complete then we can decompose $f(t)$ as

$$f(t) = \sum_{m} \sum_{n} w_{m,n} \psi_{m,n}(t)$$ (3.5)
where the wavelet coefficients \( w_{m,n} \) are given by

\[
    w_{m,n} = \langle f(t), \psi_{m,n}(t) \rangle = 2^{m/2} \int f(t)\psi(2^m t - n)dt.
\]  

When \( t \) as well is discrete we have what is called the discrete wavelet transform (DWT). A subject closely linked to the discrete wavelet transform that will be introduced in the next section is multiresolution analysis.

A simple wavelet that is useful for introducing wavelets and that we will use later is the Haar wavelet function.

\[
    \psi(t) = \begin{cases} 
    1 & 0 \leq t < \frac{1}{2} \\
    -1 & \frac{1}{2} \leq t < 1 \\
    0 & \text{otherwise}
    \end{cases}
\]  

Figure 3.1. The Haar wavelet function.

3.3 Multiresolution analysis

The central idea of multiresolution analysis (MRA) is fairly simple, namely to approximate a function \( f(t) \) at different levels of resolution with the help of a scaling function \( \phi(t) \). A simple example of a scaling function is the Haar scaling function.

\[
    \phi(t) = \begin{cases} 
    1 & 0 \leq t < 1 \\
    0 & \text{otherwise}
    \end{cases}
\]  

If we let

\[
    \phi_k(t) = \phi(t - k), \quad k \in \mathbb{Z}
\]
then we can approximate $f(t)$ with

$$f_0(t) = \sum_k a_k \phi(t - k)$$

where $a_k$ can be computed as the mean of $f(t)$ on the interval $[k, k+1]$ or $a_k =< f, \phi(t - k) >$. We let $V_0$ denote the set of functions which can be written on the form of $\sum_k a_k \phi(t - k)$ in $L^2(\mathcal{R})$. We can in the same manner, by defining

$$\phi_{j,k}(t) = 2^{j/2} \phi(2^j t - k), \quad j, k \in \mathbb{Z}$$

and if we choose $j > 0$, approximate $f(t)$ at an arbitrary higher resolution. The functions $\phi_{j,k}(t)$ are known as dilations of the scaling function $\phi(t)$. If we denote $V_j$ as the set of functions that can be written as

$$f_j(t) = \sum_k a_{j,k} \phi_{j,k}$$

where $a_{j,k} =< f, \phi_{j,k} >$, then we can easily verify that

$$V_0 \subset V_1 \subset \ldots \subset V_j \subset V_{j+1}$$

holds. The approximation of a function $f(t) \in L^2(\mathcal{R})$ onto the set $V_j$ is the orthogonal projection of $f(t)$ onto $V_j$.

What $V_0 \subset V_1 \ldots$ basically means is that the scaling function $\phi(t)$ can be written as a sum of its dilations at an arbitrary higher resolution. For $j = 1$ this would give us

$$\phi(t) = \sqrt{2} \sum_k h_k \phi(2t - k), \quad (3.9)$$

Figure 3.2. The Haar scaling function.
which is known as the scaling equation. We have so far concluded that we can approximate a function \( f(t) \) at an arbitrary level of resolution with the help of a
scaling function \( \phi(t) \) and its dilations and that the scaling function can be written as sum of its dilations.

If we assume that we have approximated \( f \) at two consecutive scales, \( f_{j-1} \in V_{j-1} \) and \( f_j \in V_j \), we can then compute the difference between the two as
\[
d_{j-1} = f_j - f_{j-1}.
\]
Since \( f_{j-1} \) is an approximation at a coarser scale than \( f_j \), \( d_{j-1} \) gives us the difference in details between the two. If we assume that our scaling function is the Haar scaling function in equation 3.8, then
\[
a_{j-1,k} = \frac{1}{2}(a_{j,2k} + a_{j,2k+1}).
\]
We can also see that \( d_{j-1} \in V_j \). If we recall the definition of the Haar wavelet from equation 3.7 we see that we can write \( d_{j-1} \) as
\[
d_{j-1}(t) = \sum_k w_{j-1,k}\psi_{j-1,k}(t).
\]
The coefficients \( w_{j-1,k} \) can be shown to be the same as the coefficients in equation 3.6, see for instance [3].

In MRA a function \( \psi \) is termed a wavelet if the set of functions, known as the detail space \( W_{j-1} \), spanned by \( \psi_{j-1,k} \) complements \( V_{j-1} \) in \( V_j \). This can be written as \( V_j = V_{j-1} \oplus W_{j-1} \). Thus an approximation at an arbitrary scale can be expressed as sum of details and an approximation at a coarser scale. Formalized this can be written as
\[
V_j = W_{j-1} \oplus W_{j-2} \oplus \ldots \oplus W_0 \oplus V_0.
\]

In the same manner as we gained the scaling equation it is possible to attain its counterpart, the wavelet equation.
\[
\psi(t) = \sqrt{2} \sum_k g_k \phi(2t - k) \tag{3.10}
\]
Therefore we can finally conclude that a signal represented at an arbitrary scale with a given scaling function can be perfectly represented by its approximation at a coarser scale together with its details at the scales in between expressed in terms of dilations of the scaling function\(^2\).

To illustrate how this works we can assume that our scaling function is the sinc function,
\[
\phi(t) = \text{sinc}(t) = \frac{\sin(\pi t)}{\pi t},
\]
\(^2\)As in the previous section we have kept the discussion about MRA on a rather shallow level and we have not discussed different conditions on the scaling function and the wavelet function and how the two relate to each other. The interested reader is referred to [3] and [4] for a denser introduction to the theory of multiresolution analysis.
thus $V_0$ is the set of all functions belonging to the frequency band $[0, \pi]$ (actually $[-\pi, \pi]$ but we neglect the negative frequencies). Then the detail space $W_0$ is the set of functions belonging to $[\pi, 2\pi]$. Approximating/decomposing a signal at a coarser scale in this case can be seen as a decomposition of the frequency band according to figure 3.3.

In this case there is no overlap between the different frequency bands since the sinc function corresponds to a perfect low pass filter. For other scaling functions there will be some overlap between the frequency bands. This way of constantly decomposing the frequency band of the approximated signal into frequency bands of $[0, 2^j \pi]$ is known as a dyadic decomposition. To decompose the signal into different frequency bands or on different scales is often known as decomposing the signal into different subbands.

### 3.4 The forward and the inverse wavelet transform

According to the previous section we can decompose $f_{j+1}$ as

$$f_{j+1} = f_j + d_j$$
or written more specifically

\[ \sum_l a_{j+1,l} \phi_{j+1,l}(t) = \sum_k a_{j,k} \phi_{j,k}(t) + \sum_k w_{j,k} \psi_{j,k}(t) \]  

(3.11)

where \( a_{j+1,k} = \langle f, \phi_{j+1,k} \rangle \), \( a_{j,k} = \langle f, \phi_{j,k} \rangle \) and \( w_{j,k} = \langle f, \psi_{j,k} \rangle \). Performing a scalar multiplication on this equation with \( \phi_{j,k} \) and under the assumption of orthogonality between the scaling and the wavelet functions will result in

\[ \sum_l a_{j+1,l} \phi_{j+1,l} \phi_{j,k} = a_{j,k} \].

From equation 3.9 we obtain

\[ \langle \phi_{j+1,k}, \phi_{j,k} \rangle = \sum_m h_m \langle \phi_{j+1,l'}, \phi_{j+1,m+2k} \rangle = h_{l-2k} \]

which gives us

\[ a_{j,k} = \sum_l a_{j+1,l} h_{l-2k}. \]  

(3.12)

In a similar manner for the wavelet coefficients \( w_{j,k} \) we can obtain

\[ w_{j,k} = \sum_l a_{j+1,l} g_{l-2k}. \]  

(3.13)

Thus the computation of the scaling and wavelet coefficients at a coarser scale of a signal can be seen as filtering the signal with the filters \( H^* \) and \( G^* \), where \( h^*[n] = h[-n] \) and \( g^*[n] = g[-n] \), followed by downsampling with a factor 2. The analysis of the signal can continue by repeating the procedure on the scaling coefficients \( a_{j,k} \). Thus the analysis/decomposition of the signal into different subbands can be viewed as an iterated filter bank.

![Figure 3.4. The forward wavelet transform as an iterated filter bank.](image)

In the same manner as we derived the equations for computing the scaling and the wavelet coefficients at a coarser scale we can derive an equation for computing the scaling coefficients at a finer scale, namely

\[ a_{j+1,k} = \sum_l (h_{k-2l} a_{j,k} + g_{k-2l} w_{j,k}). \]  

(3.14)
3.5 Examples of wavelets

This far we have assumed that we are dealing with infinite signals and ignored the fact that most, if not all, signals that we will come to analyze are finite. When analyzing finite signals we have to deal with boundary problems. Another obstacle is that convolution, which is used in equations 3.12, 3.13 and 3.14, leads to an expansion of the signal length. If the signal has a length of $k$ and the filter a length of $l$ the filtered signal will have a length of $k + l - 1$. In this thesis we will deal mainly with symmetric wavelet filters, therefore we state, in accordance with [6], that a symmetric extension of the signal will lead to a non expanding length for the filtered signal and that it also decreases boundary effects, see [7] and [8].

3.5 Examples of wavelets

A well known choice of wavelet is the (5,3) wavelet belonging to the biorthogonal Daubechies wavelet family found in [9], where the filter coefficients $h_k$ and $g_k$ are given as

$$h = [-1/8, 1/4, 6/8, 1/4, -1/8]$$

$$g = [-1/2, 1, -1/2].$$

(5,3) stands for the number of taps in the low-pass and the high-pass filters. Another common notation for the same filters is the (2,2) wavelet but this time (2,2) represents the number of vanishing moments of the low-pass/analysis and the high-pass/synthesis filter. The number of vanishing moments corresponds to the number of zeros at pi of the polynomials of the z-transformed wavelet filters. If an analyzing filter has $p$ vanishing moments this means that the coefficients for a $p$th order wavelet-polynomial will be zero which in turn means that any other polynomial up to order $p-1$ can be described completely in the scaling space. This is why wavelet transformations are useful in data compression. Roughly speaking, the more vanishing moments the analyzing filter has the better the scaling space is to represent more complex signals accurately. In this thesis we will let $(X,Y)$ denote the number of taps in the low-pass respectively high-pass filter.

As mentioned at the beginning of this chapter the theory for 1D signals can easily be extended to n-D signals by using separable wavelets. A simple example

---

$^3$Analyzing the filters $h$ and $g$ will reveal a low-pass and a high-pass filter.
Wavelet theory is 2D signals where the earlier described filtering process is first applied to the rows of the signal and then to columns of the signal (or the other way around since the wavelet transform is linear). If we let \( s \) represent our original signal, \( H_x \) respectively \( H_y \) denote the operations for computing the low-pass coefficients along the rows respectively the columns and \( G_x \) respectively \( G_y \) denote the same but for the high-pass coefficients. Then we can describe the result of a wavelet transform as depicted in figure 3.6.

![Figure 3.6. The separable wavelet transform applied to a 2D signal.](image)

In figure 3.6 we can see how the original 2D signal (image) is decomposed into 4 subbands/sub images. One of the subbands \((H_y H_x s)\) contains an approximation of the original image whereas the others contain detail information from the original image, one with horizontal details \((H_y G_x s)\), one with vertical details \((G_y H_x s)\) and one with diagonal details \((G_y G_x s)\). Continuously decomposing the approximation subband will result in what is known as a Mallat-pyramid. Worth noting is that this way of decomposing an image is not the only one. We could as well decompose the detail subbands. This will lead to an arbitrary wavelet packet structure, whereas a Mallat-pyramid is a special case of a wavelet packet structure.

![Figure 3.7. An example of a Mallat-pyramid (left) and an arbitrary wavelet packet structure (right).](image)
3.5 Examples of wavelets

Assume that a 2D signal consists of a grey scale image from a CT examination\(^4\), see figure 3.8, and that we use the wavelet filters from equations 3.15 and 3.16. Applying the filters first to the columns and then to the rows of the image will yield the result in figure 3.9. Looking at figure 3.9 we see what was earlier described, that one of the sub images holds an approximation of the original image (although not so visible due to the strongly adjusted contrast level) whilst the other sub images hold information about the details. It is also easily verified that the detail sub images respectively display horizontal, vertical and diagonal details. The dyadic decomposition can continue if we transform the approximated sub image, which results in figure 3.10.

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{original_image.png}
\caption{The original image.}
\end{figure}

\(^4\)All image data contained in this thesis has been provided by Center for Medical Image Science and Visualization (CMIV) in Linköping.
Figure 3.9. The image after one step of decomposition.

Figure 3.10. The image after two steps of decomposition.
Chapter 4

Integer-to-integer wavelet transforms

This chapter presents the theory behind integer-to-integer wavelet transforms. The founding theory for this is based upon lifting steps and rounding as described in [10], [11] and [12]. Other attempts have been made to create general integer-to-integer wavelet transforms. Two of these attempts can be found in [13] and [14], but both of them present theories that are more or less equal to the lifting scheme with rounding. Others have tried to build special integer-to-integer transforms such as the S+P-transform, which can be found in [15], although this transform as well can be found with the lifting scheme and rounding.

4.1 The lifting scheme

The lifting scheme was foremost developed as a method for constructing wavelets. Prior attempts to find wavelets included finding filters $h$ and $g$ that would solve the following equations:

$$
|H(\omega)|^2 + |H(\omega + \pi)|^2 = 1
$$

$$
|G(\omega)|^2 + |G(\omega + \pi)|^2 = 1
$$

$$
H(\omega)\tilde{G}(\omega) + H(\omega + \pi)\tilde{G}(\omega + \pi) = 0
$$

This was done whilst at the same time trying to fulfil requirements for the number of vanishing moments and the number of taps of the analysis and the synthesis filters. The basic idea of the lifting scheme was to construct wavelets known as second generation wavelets, which are wavelets that are adapted to special cases, such as wavelets on an interval, on curves and surfaces or adapted to irregular spacing. Although this was the original intention of the lifting scheme it is today used because of its speedup of the wavelet transform, its allowance of in-place computations of the wavelet transform and its built-in possibility to perform integer-to-integer wavelet transforms.
We will start the presentation of the lifting scheme by considering a signal \( x_k \), where \( k \in \mathbb{Z} \) and \( x_k \in \mathbb{R} \), that is split up into two parts, the even indexed samples \( x_e = x_{2k} \) and the odd indexed samples \( x_o = x_{2k+1} \). This is known as a Lazy wavelet transform. One can assume that these two disjoint sets are correlated and that it would be possible given one set to build a predictor, \( P \), of the other set. For example, given the even set \( x_e \) we could predict the odd set \( x_o \). Since the predictor probably is not perfect we need to record the difference \( d \).

\[
    d = x_o - P(x_e)
\]

Thus given the detail \( d \) we can reconstruct \( x_o \). This operation is known as a dual lifting step. The better predictor we have the sparser \( d \) will be and thus the entropy of \( d \) will be less than for \( x_o \). A simple predictor of an odd sample would be to use the average of the two neighboring even samples.

\[
    d_k = x_{2k+1} - \frac{(x_{2k} + x_{2k+2})}{2}
\]

This far we have transformed \((x_e, x_o)\) to \((x_e, d)\) by taking advantage of the spatial correlation. Now we will take advantage of the frequency correlation. Since \( x_e \) is obtained by simply sub sampling \( x \), there is a great risk of aliasing in \( x_e \). To deal with this we can replace \( x_e \) with the values \( s \) where we have used an update operator, \( U \), on the details.

\[
    s = x_e + U(d)
\]

This operation is known as a primal lifting step. We can reconstruct \( x_e \) if we are given \((s, d)\) and then we can reconstruct \( x_o \). From this we can conclude that it

\[
    \log(p(d)) = \sum_{d_k \in D} p(d_k) \log p(d_k)
\]

where \( p(d_k) = Pr(d = d_k) \) and \( D \) denotes the set of all possible \( d \). The entropy is often used in data compression as a measurement of how well something can be compressed.

**Figure 4.1.** A block diagram of a dual and a primal lifting step.
4.2 Perfect reconstruction with finite precision

does not matter how we choose $P$ and $U$, the scheme is always reversible. An update operator that reduces the aliasing in our example is given by

$$s_k = x_{2k} + (d_{k-1} + d_k)/4.$$  

If we closely examine this example we see that this wavelet corresponds to the wavelet that was used in the previous chapter, namely the $(5,3)$ wavelet, but instead of being constructed by solving the earlier mentioned equations, we have now constructed it by simply reasoning. The advantage is not only the easier construction but also the fact that this wavelet transform framework requires less computational power and can be done in-place, thus eliminating the need for auxiliary memory allocation during the computation of a wavelet transform.

![Figure 4.2. The (5,3) wavelet implemented with the lifting scheme.](image)

The above explained construction scheme could now be applied to other settings such as irregular spaced samples but we will refrain from going deeper into this. Instead we concentrate on the fact that the lifting scheme provides a framework for wavelet transforms to be performed with less computational power needed and with in-place computations. Naturally the question arises, which wavelet families from the first generation wavelets can be built with the lifting scheme? According to [10], [11] and [12] all finite wavelet filters can be obtained as a Lazy wavelet transform followed by a finite number of dual and primal lifting steps and a scaling step. The factorization of wavelet filters into primal and dual lifting steps is obtained with the help of the Euclidean algorithm.

4.2 Perfect reconstruction with finite precision

This far we have not been concerned with the precision of our computations, but with a little thinking we can realize that in order for our computations to render a perfect reconstruction, after a wavelet transformation followed by an inverse wavelet transformation, we must have infinite precision. Despite the fact that today’s computers offer floating point types with a very high precision, this is

---

$^{2}$To see how this works in detail the interested reader is referred to [3], [10] and [11].
not sufficient. Many computations are also performed on integer data, thus it would be preferable to be able to do integer-to-integer transforms. Integer-to-integer transforms does not mean that the computations themselves are integer operations but that they can be floating point operations where the answers are rounded of to integers. To see how the lifting scheme can help us in this we will look at a simple example with the Haar transform.

The Haar transform can in its unnormalized form be written as

\[
\begin{align*}
    s_{1,k} &= \frac{s_{0,2k} + s_{0,2k+1}}{2} \\
    d_{1,k} &= s_{0,2k+1} - s_{0,2k}
\end{align*}
\]

and with its inverse as

\[
\begin{align*}
    s_{0,2k} &= s_{1,k} + d_{1,k}/2 \\
    s_{0,2k+1} &= s_{1,k} - d_{1,k}/2.
\end{align*}
\]

This is clearly not an integer transform due to its division by two but it can easily be converted into an integer transform by applying the lifting scheme and adding a rounding step.

\[
\begin{align*}
    d_{1,k} &= s_{0,2k+1} - s_{0,2k} \\
    s_{1,k} &= s_{0,2k} + \lfloor d_{1,k}/2 \rfloor
\end{align*}
\]

Its inverse can be found even though the transform now is non-linear.

\[
\begin{align*}
    s_{0,2k} &= s_{1,k} - \lfloor d_{1,k}/2 \rfloor \\
    s_{0,2k+1} &= s_{0,2k} + d_{1,k}
\end{align*}
\]

Aided by this example it is easy to understand how we can build an integer transform from every wavelet transform. Since all wavelet filters can be constructed with the lifting scheme we simply need to, as in the example, add a rounding operation before adding or subtracting in the dual and the primal lifting steps. Thus integer-to-integer transforms and their inverse can be performed as follows:

Split the signal according to the Lazy wavelet transform.

\[
\begin{align*}
    s^0_{1,k} &= s_{1,2k} \\
    d^i_{1,k} &= s_{1,2k+1}
\end{align*}
\]

Perform the dual lifting steps but remember the rounding operation.

\[
d^i_{1,k} = d^{i-1}_{1,k} - \sum_l \left[p_i^l s^{i-1}_{1,k-l}\right]
\]

Do the same thing with the primal lifting steps.

\[
s^i_{1,k} = s^{i-1}_{1,k} + \sum_l \left[u_i^l d^i_{1,k-l}\right]
\]
4.2 Perfect reconstruction with finite precision

After $M$ pairs of dual and primal lifting steps we have our low- and high-pass coefficients up to a scaling factor $K$.

\begin{align*}
s_{1,k} &= s_{1,k}^M/K \quad \text{(4.5)} \\
d_{1,k} &= Kd_{1,k}^M \quad \text{(4.6)}
\end{align*}

It can be shown, see [10] and [11], that this scaling factor can either be omitted or replaced by three extra lifting steps. If omitting the scaling factor $K$ it should preferably be close to 1, which can be achieved by taking advantage of the non-uniqueness of the lifting factorization to gain a scaling factor close to 1.

\textbf{Figure 4.3.} A block diagram of a dual and primal lifting step with rounding.

In section 3.4 we stated that a symmetric extension of finite signals produces a non-expanding wavelet transform. The same is valid for integer-to-integer wavelet transforms when performed according to the described framework. This is proved in [16].

We have already mentioned that the lifting framework decreases the computation time for the wavelet transform and that the computations are performed as floating operations with rounding. The performance can be further improved if the lifting coefficients are chosen as rationals where the denominators can be written as a power-of-2. Thus the wavelet transform can be implemented with only integer operations such as integer addition, subtraction and multiplication and bit shifts.
Integer-to-integer wavelet transforms
Chapter 5

Overview of wavelet-based codecs for medical image stacks

In this chapter some well known 2D image codecs alongside with their extensions in 3D and other 3D codecs are presented\(^1\). We will discuss some of their advantages respectively disadvantages when it comes to representing medical image stacks in an efficient way.

5.1 2D codecs

Wavelet-based codecs have been very popular since the establishment of the connection between wavelets and MRA in [17]. Many codecs arose quickly and were shown to outperform older ones, at least when it came to lossy coding, see for example [17] and [18]. A large obstacle for the wavelet-based codecs were their incapability of performing lossless coding, which was due to their need for infinite precision in their wavelet transforms. All of this changed with the introduction of integer-to-integer transforms. Some of the most well known 2D codecs are the embedded zerotree wavelet algorithm (EZW) [19], set partitioning in hierarchical trees (SPIHT) [20] and embedded block coding with optimized truncation (EBCOT) [21]. All three have gained a wide acceptance in 2D image compression. EBCOT stands out when compared with the others. This is due to some of its characteristics and because it forms the basis for the new image compression standard JPEG2000, [22].

The EZW algorithm is based upon the following steps:

- Wavelet transformations to provide a pyramid-like multiresolution representation of the image.

---

\(^1\) A codec refers to an algorithm for coding and decoding in data compression.
Overview of wavelet-based codecs for medical image stacks

- Zerotree coding to provide binary significance maps indicating the position of significant coefficients.
- Ordering of coefficients that are deemed significant.
- Arithmetic coding.

The coder produces an embedded bit stream, which means that the bits in the stream are ordered in importance and that the stream can be truncated anywhere along the stream but still produce the best possible representation of the image given the fixed number of bits. The EZW takes advantage of the inter-subband dependencies in order to further decrease the correlation of the image data after the wavelet transformations.

The SPIHT algorithm is more or less a refinement of the EZW algorithm. The refinement lies within its organization of the wavelet coefficients across the subbands, which achieves a better coding of the significance maps.

The EBCOT algorithm differs in its implementation in comparison with the EZW and SPIHT algorithms. EBCOT is based upon the following steps:

- Wavelet transformations to produce an arbitrary wavelet packet structure.
- Each subband is divided into smaller blocks.
- The smaller blocks are each coded into separate embedded bit streams. This coding is based upon layered zero coding (LZC).
- Post-compression rate-distortion (PCRD) is used to determine which bit streams to be sent and how much of each bit stream to meet a certain request of resolution and quality.

All three algorithms have properties that make them useful for progressive transmission. They all offer bit streams that are scalable in quality. This means that if a low quality image is requested, only data containing information valid for this representation is sent, but data containing information for a better quality can be transmitted later to improve the image. With the implementation of integer-to-integer wavelet transform they all offer lossless coding as well. What sets EBCOT apart from the others is that it produces a bit stream that is also scalable in resolution and that it supports ROI-viewing\(^2\). Scalability in resolution is achieved due to the fact that no inter-subband dependencies are utilized for coding but only intra-subband dependencies. The ROI-viewing is due to the partitioning of the subbands into smaller blocks.

Because of their properties they have been suggested to be well suited for representation and transmission of medical images in a PACS or in telemedicine, [23], [24], [25] and [26]. Different wavelet filters have been tried in order to find the best for coding of medical images, [27] and [28]. Today JPEG2000 is the only image codec accepted by DICOM that is based upon wavelets. In its lossless mode JPEG2000 uses the already mentioned (5,3) wavelet, [22].

\(^2\)See [21] for an explanation of the difference between scalability in resolution and quality
5.2 3D codecs

The EZW and SPIHT algorithm have both been extended to 3D settings, see [29] and [30] respectively [31] and [32], although their respective approaches slightly differ. Whereas the EZW algorithm is applied to the whole stack of images the SPIHT algorithm is applied to what is called groups of slices (GOS), i.e. the original image stack is split into smaller sets of images. This is done to decrease the number of images that must be held in memory at the same time but also to introduce a sense of random access in reconstructing the images. As in their 2D counterparts these algorithms produce a bit stream suitable for progressive transmission and scalable quality. Both of them have as well been shown to outperform 2D lossless codecs such as JPEG-LS and CALIC when it comes to compression performance of medical image stacks.

The JPEG2000 standard is until today only implemented for 2D datasets although there are plans for a 3D extension of JPEG2000\(^3\). In [33], [34] and [35] the interested reader can find other examples of suggested 3D codecs for medical image stacks.

Common for all these different 3D codecs is that they suggest themselves to be suitable for compression and progressive transmission within a PACS or telemedicine. This is noteworthy since they are not or very little concerned with how a practitioner actually works and the computational complexity of their suggested algorithms. Thus these codecs lack in reconstructing flexibility and speed. Many of them are just focused on achieving the best compression possible of the whole stack and are not interested in finding a representation that will assist the practitioner in his/her work. Only [31], [32] and [33] are concerned with the fact that most of the time the practitioner only views 2D images and is interested in having some sort of random access to the original images. The question of orthogonal projections is mentioned nowhere. [33] and [34] both state that they have not been concerned with computational complexity during the development of their respective algorithms but that they both believe that with a bit of optimization and the use of multiprocessor architecture the complexity of the their algorithms will be no major issue, an attitude that appears nonchalant.

However, there are codecs that are concerned with the workflow of the practitioner and the computational complexity, see for instance [36], [37] and [38]. [36] tries to achieve a hierarchical data structure suitable for progressive transmission by introducing a certain wavelet packet structure instead of the commonly used pyramid-like wavelet structure. [37] develops a scheme for thick slab viewing (low

\(^3\)For more information about the progress of this work the reader is referred to: http://www.jpeg.org/jpeg2000/
axial resolution and high planar resolution) in order to facilitate the practitioner with a tool to navigate the image stack. In this case a Huffman coder is used for coding the wavelet coefficients in order to decrease the computational complexity. [38] focuses on developing a codec with a high throughput, i.e. low computational complexity, and random access of the images in the stack. This is achieved by applying a low complexity Columb-Rice coder to the wavelet coefficients and using short wavelet filters. Although all three contain many promising features, none of them manage to combine all features in order to create an efficient representation of a medical image stack.
Chapter 6

Proposals for an efficient wavelet representation

In this chapter two different wavelet packet structures are presented that are suitable for an efficient wavelet representation. One that is suitable for 2D viewing of the original images or 3D viewing and a second that is especially suitable for 2D viewing of orthogonal projections.

6.1 Wavelet packet structure #1

We stated in section 4.2 that a wavelet transform implemented with the lifting scheme and rounding is a non-linear operation. This is noteworthy since the normal wavelet transformation is linear. The consequence of the non-linear transformation is that when a dataset is wavelet transformed the order of the transform steps is of great relevance. To be able to perfectly reconstruct the transformed dataset, its inverse transform order must match the reversed transform order. For instance, if an image is transformed in the order $xyxy$ than it must be inverse transformed in the order $yxyx$ in order to yield perfect reconstruction. This is certainly a limiting factor when it comes to the flexibility of reconstructing an image stack, but it is also something that can be used in order to decrease the computational complexity$^1$.

The computational complexity in the previously mentioned codecs, see chapter 5, consists of both the coding and the decoding of wavelet coefficients but also of the actual wavelet transformations that must be carried out. The computational burden due to the wavelet transformations is not to be neglected. Since the scope of this thesis, see section 1.3, is to find an efficient wavelet representation without utilizing coding and decoding of the wavelet coefficients, the main concern, in

$^1$The author of this thesis finds is rather surprising that so few pay any attention to the inherited limitation due to the non-linear transformation. The impression is that either no one sees it as a problem or fails to grasp the implications of it. A reason for this could be that many are interested in lossy compression were the introduced errors through a false inverse transform order are of negligible size.
regards to computational complexity, is to improve the speed of the wavelet transformations. An obvious improvement would be achieved by decreasing the number of wavelet transformations that must be performed. This is something that can be done by utilizing the consequences of the non-linearity in integer-to-integer wavelet transforms.

If we let \( s \) denote a 2D dimensional dataset then we can wavelet transform it in the order \( xy \) by first applying \( H_x \) and \( G_x \) to it and then applying \( H_y \) and \( G_y \) to \( H_x s \) and \( G_x s \), see figure 6.1. Remember now that the use of \( H_x \) and \( G_x \) are for notational use only. We are not applying filters as we did in section 3.5 but the wavelet transforms are done according to the lifting scheme with rounding. If we now wish to reconstruct the data we have to inverse transform it in the order \( yx \). This is done by applying \( H_y^{-1} \) to \( H_y H_x s \) and \( H_y G_x s \) and \( G_y^{-1} \) to \( G_y H_x s \) and \( G_y G_x s \). The second step is to apply \( H_x^{-1} \) to \( H_x s \) and \( G_x^{-1} \) to \( G_x s \). If we observe this process in detail we can see that the step of decomposing \( G_x s \) into \( H_y H_x s \) and \( G_y G_x s \) is unnecessary and just forces us to execute extra wavelet transformations and inverse wavelet transformations. Skipping these extra wavelet transformations will decrease the number of wavelet transformations by a factor of 25% in this specific example\(^2\), see figure 6.1.

\[ \begin{array}{ccc}
  s & \rightarrow & H_x s \quad G_x s \\
  \downarrow & & \downarrow \quad \downarrow \\
  H_y H_x s & H_y G_x s \\
  & G_y H_x s & G_y G_x s \\
\end{array} \]

\[ \begin{array}{ccc}
  s & \rightarrow & H_x s \quad G_x s \\
  \downarrow & & \downarrow \quad \downarrow \\
  H_y H_x s \\
  & G_y H_x s \\
\end{array} \]

Figure 6.1. A 2D wavelet transformation is performed to gain the normal Mallat-pyramid. In the example above it is done in the normal way whereas the example below utilizes the non-linearity in integer-to-integer wavelet transformations to dismiss unnecessary wavelet transformations.

This means that, if we can predict how a practitioner navigates and views a stack, we can create a wavelet packet structure that is suited for this particular workflow, but with a decreased computational complexity simply by skipping

---

\(^2\)A similar structure has been suggested, but without explaining why, in [39] to reduce computational complexity in a 2D codec where a DPCM coder is used together with a wavelet coder.
unnecessary wavelet transformations. When it comes to 2D viewing of original images experience states that the practitioner prefers to have full planar resolution (x and y) before increasing the axial resolution (z)\(^3\). Thus an appropriate transform order would for example be \(zzzyxyx\). This would allow the practitioner to use data with low planar and axial resolution to navigate in. The practitioner can then increase the planar resolution for diagnostic purposes and then increase the axial resolution if necessary. This wavelet packet structure is depicted in figure 6.2. In this specific example the number of wavelet transformations along the y-axis is decreased by 87.5% and along the x-axis by 90.375%.

In the same manner this can be utilized for 3D viewing. Increasing the resolution in just one dimension is seldom done, but often the resolution in all three dimensions is increased at the same time. Thus an appropriate transform order would for example be \(zyzzyzyx\). This wavelet packet structure is depicted in figure 6.3. For this case the number of wavelet transformations along the y-axis is decreased by 50% and along the x-axis by 75%.

---

\(^3\)Here assuming that the original images are in the transversal plane.
Proposals for an efficient wavelet representation

rounding operations per coefficient. The number of rounding operations is in turn dependent on the number of wavelet transformations and the number of lifting steps. The maximum error in a fully reconstructed wavelet coefficient but with the wrong inverse transform order is approximately

\[ \pm 0.5 \times \text{number of lifting steps} \times \text{number of wavelet transformations}. \]

However, this error can be decreased by changing our rounding operations in equations 4.3 and 4.3 to instead operate on the whole sums.

\[
\begin{align*}
d_{i_{1,k}} &= d_{i-1_{1,k}} - \left\lfloor \sum_l p_l s_{i_{1,k-1}} \right\rfloor \\
s_{i_{1,k}} &= s_{i-1_{1,k}} + \left\lfloor \sum_l u_l d_{i_{1,k-1}} \right\rfloor
\end{align*}
\]

Thus the maximum error will be

\[ \pm 0.5 \times \text{number of wavelet transformations}. \]

In order to examine how the error depends on the number of inverse transformations in the wrong order an image stack from a CT examination was transformed and inverse transformed in the wrong order. The stack consisted of 256 images with 256*256 pixels per image and 12 bpp. The mean square error (MSE) was used to measure the error. The MSE is defined as

\[
x_{\text{MSE}} = \frac{1}{N} \sum_i (x[i] - \hat{x}[i])^2
\]

where \(N\) is the number of pixels, \(x[i]\) is original pixel number \(i\) and \(\hat{x}[i]\) is reconstructed pixel number \(i\). The peak signal-to-noise ratio (PSNR) was also used, which is defined as

\[
x_{\text{PSNR}} = 10 \log_{10} \left( \frac{x_{\text{max}}^2}{x_{\text{MSE}}} \right)
\]

where \(x_{\text{max}}\) is the maximum pixel value, which in this case was 4095 since the data used 12 bits for each pixel.

In table 6.1 it is obvious that the more inverse transform steps that are done in the right order the smaller the error will be. Looking at the difference between a reconstructed image and an original image from the stack shows that the error appears to be square shaped, see figure 6.4. Since the PSNR is rather high for all examples it is however likely that the introduced error will not heavily affect the visual quality. In figure 6.5 we see one of the reconstructed images with what appears to be a perfect reconstruction. Changing the window level to a very narrow window close to 0 will reveal the hidden errors, see figure 6.6.
### Table 6.1. Error size depending on the number of wavelet transformations in the wrong order

<table>
<thead>
<tr>
<th>Transform order</th>
<th>Inverse transform order</th>
<th>MSE</th>
<th>PSNR [dB]</th>
</tr>
</thead>
<tbody>
<tr>
<td>zyxzyx</td>
<td>zyxzyx</td>
<td>1.74</td>
<td>42.7</td>
</tr>
<tr>
<td>zyxzyx</td>
<td>zzyyxx</td>
<td>2.26</td>
<td>41.6</td>
</tr>
<tr>
<td>zyxzyx</td>
<td>yyxxzz</td>
<td>1.78</td>
<td>42.7</td>
</tr>
<tr>
<td>zyxzyy</td>
<td>xzyyy</td>
<td>1.51</td>
<td>43.4</td>
</tr>
<tr>
<td>zyxz</td>
<td>xzyy</td>
<td>0.947</td>
<td>45.4</td>
</tr>
<tr>
<td>zyx</td>
<td>zyx</td>
<td>0.926</td>
<td>45.5</td>
</tr>
</tbody>
</table>

**Figure 6.4.** Difference between a reconstructed image and an original image.
Figure 6.5. A reconstructed image with a wide window level.

Figure 6.6. A reconstructed image with a narrow window level.
If we neglect the errors, which were introduced through using a false inverse transform order, it will allow us to construct a wavelet packet structure that is much more flexible in its use for different reconstructions. The normal Mallat-pyramid does not allow much flexibility in the inverse transform order. Assume that our transform order is $xyxy$. Then we know since before that to have a perfect reconstruction we must have the inverse transform order $yxxy$ but if we neglect the errors we could use $xyyx$ or $xyxy$. Still this is not as flexible as we could wish, because after the first inverse $x$ transformation we must perform an inverse $y$ transformation. This is due to a mismatch in the dimensions of the subbands, see figure 6.7.

Figure 6.7. With the transform order $xyxy$ the first inverse transforms steps must be $yx$ or $xy$ and can not be $xx$ or $yy$.

Instead of only transforming the subbands that contain low-pass data from all directions, we transform all subbands containing low-pass data for the given transform direction, see figure 6.8. This wavelet packet structure will allow a

Figure 6.8. Wavelet transformations are applied to all subbands containing low-pass data for the given transform direction.
greater flexibility in the inverse transform order, see figure 6.9. What this basically means is that we can arbitrarily choose in which order we wish to reconstruct the data.

![Figure 6.9](image.png)

**Figure 6.9.** With a more flexible wavelet packet structure the inverse transform order is much more flexible and thus allows a greater variety.

At this point it might still seem a bit unclear how this wavelet packet structure can help us in the pursuit of an efficient wavelet representation. In figure 6.10 we can see the 3D counterpart for this wavelet packet structure. If we assume that this dataset corresponds to an image stack we can then transmit the lowest subband to the practitioner who can choose to view it in either the transversal, sagittal or coronal plane. The practitioner can then arbitrarily choose in which direction to increase the resolution, see figure 6.11, by requesting additional subbands. As stated before, the normal workflow would be to increase the planar resolution and have what is known as thick slab viewing in one of the orthogonal planes. This

![Figure 6.10](image.png)

**Figure 6.10.** An efficient wavelet packet structure for viewing of 2D orthogonal projections.
flexibility would not have been possible with either the normal Mallat-pyramid structure or the structure presented in the previous section. Naturally this structure can be used for 3D viewing as well or just 2D viewing of the original images, but with an increased computational burden when compared to wavelet packet structure #1.

Figure 6.11. Thick slab viewing in either the transversal, sagittal or coronal plane.
Proposals for an efficient wavelet representation
Chapter 7

Implementation of wavelet packet structure #1

In this chapter we present the implementation of wavelet packet structure #1 from the previous chapter. This structure was chosen since it has a better computational complexity than structure #2 but it is still flexible enough to meet the required features for an efficient wavelet representation. In the first section of this chapter a short summary of the implementation is given and in the following sections the reader can find some issues related to the implementation.

7.1 Implementing wavelet packet structure #1

Before we start to discuss the implementation it is important to recall the features an efficient wavelet representation should support namely lossless viewing, arbitrary 2D viewing, progressive transmission, scalable resolution, thick slab viewing and ROI-viewing. With the suggested wavelet packet structure some of these demands have already been fulfilled such as lossless viewing (integer-to-integer transforms), progressive transmission (inherent in all wavelet representations), scalable resolution (inherent in all multiresolution representations) and thick slab viewing (which is supported by wavelet packet structure #1 and the transform order zzzzyxyz). Thus what is left to fulfil is arbitrary 2D viewing and ROI-viewing, although we could say that this is already supported in a sense. Since we will most likely use a wavelet filter with a compact support, we can extract just a part of the relevant subbands to reconstruct a certain image or a certain region in an image, see [33] and [38]. Another important aspect to consider for our implementation is of course the computational complexity or, to be more precise, the execution time.

The implementation is divided into two parts:

- Analysis
- Synthesis
The analysis takes place at the server and consists of splitting the image stack into groups of images that are further divided into smaller blocks. The blocks are wavelet transformed according to wavelet packet structure #1 and a given transform order. The wavelet coefficients are then rearranged and stored on disc in a number of files. Metadata is stored in a database to help the server to keep track of which data has been stored in which files.

The synthesis takes place at the server and the client. This part consists of the client requesting data from the server to fulfil a certain reconstruction request from the practitioner, i.e. a certain region with a certain resolution has been requested to be visualized. The client also informs the server of which data that is already available at the client. The server reads in its database to find which files to read from and decides which subbands from each relevant wavelet block to transmit to the client based upon the reconstruction request and information about already received data at the client. The data is then transmitted to the client, which receives the data and inverse transforms it if necessary. Data from the wavelet blocks is then extracted and used to reconstruct the requested image.
The server will always send complete subbands to the client, thus the client holds enough data to visualize several images at the same time.

The analysis and synthesis process can in pseudo code be written as:

```java
Analysis(listOfImages, transformOrder, preferredBlockSize)
{
    waveletBlocks = createNewWaveletBlocks(preferredBlockSize);
    foreach(image in listOfImages)
    {
        openImage();
        insertImageIntoWaveletBlocks(image, waveletBlocks);
        closeImage();
        if (waveletBlocksAreFull(waveletBlocks))
        {
            transformWaveletBlocks(waveletBlocks, transformOrder);
            writeWaveletBlocksOnDisc(waveletBlocks);
            addMetadataFromWaveletBlocksToDatabase();
            waveletBlocks = createNewWaveletBlocks(preferredBlockSize);
        }
    }
}
```

```java
transformWaveletBlocks(waveletBlocks, transformOrder)
{
    foreach(waveletblock in waveletblocks)
    {
        foreach(transform in transformOrder)
        {
```
{ transformWaveletBlock(waveletBlock, transform); 
  rearrangeCoefficients(waveletBlock); 
}

Synthesis(imageToReconstruct)
{
  if (!DataIsAvailableFor(imageToReconstruct))
  {
    waveletData = requestDataFromServer(imageToReconstruct,
      dataAlreadyAtClient);
    insertWaveletDataInAlreadyReceivedData(waveletData);
    inverseTransformWaveletData();
  }
  image = reconstructImageFromDataAtClient(imageToReconstruct);
  return image;
}

7.2 Blocking

It has already been suggested to perform wavelet transformations on groups of images in order to have some sort of random access to the images, see [31] and [32]. Blocking is also supported by the fact that few computers have enough working memory to hold an entire image stack for wavelet transformations in their RAM. Further dividing the groups of images into smaller blocks, i.e. blocking, can also help in reducing execution time since the pixels of an image or the voxels of a volume is stored sequentially on the working memory and thus loaded sequentially to the cache memory. This will lead to time consuming access patterns whilst performing wavelet transformations along the columns and the images of the image stack. Thus blocking will decrease the number of read and write operations on the RAM. Blocking will also help us to achieve a sense of both random access of the images and ROI-viewing. The smaller the blocks are the less overhead of data we will have whilst viewing a certain region. The overhead will not be a major problem since a practitioner rarely just views a single image but is more likely to view a group of adjacent images.

7.3 Reducing block artifacts

Blocking is a popular technique for increasing the speed of computations that must be performed on large datasets. It is used in both JPEG and JPEG2000. However, blocking does not come without a cost. How this affects the computation of wavelet coefficients while using the (5,3) wavelet can be seen in figure 7.3.

Errors will only be introduced if data from a multiresolution representation is not fully reconstructed i.e. viewing intermediate reconstructions. The errors are of course dependent on the wavelet filters that are used (length of analysis filter), number of wavelet transformations and the size of the blocks.
A CT stack consisting of 256 images with 256*256 pixels per image and 12 bpp was used in order to examine how the size of the errors introduced by blocking are affected by the choice of number of transformations, block size and wavelet filter. The two wavelet filters that were used were the (5,3) wavelet and the (9,7)M wavelet. The (5,3) wavelet was implemented as

$$d_i[n] = d_{i-1}[n] - \frac{1}{2}(s_{i-1}[n+1] + s_{i-1}[n])$$
$$s_i[n] = s_{i-1}[n] + \frac{1}{4}(d_i[n] + d_i[n-1])$$

and the (9,7)M wavelet as

$$d_i[n] = d_{i-1}[n] - \frac{1}{16}(s_{i-1}[n + 2] + s_{i-1}[n - 1] - 9(s_{i-1}[n + 1] + s_{i-1}[n]))$$
$$s_i[n] = s_{i-1}[n] + \frac{1}{4}(d_i[n] + d_i[n-1])$$

<table>
<thead>
<tr>
<th>Wavelet filter</th>
<th>Transform order</th>
<th>Block size</th>
<th>MSE</th>
<th>PSNR [dB]</th>
</tr>
</thead>
<tbody>
<tr>
<td>(5,3)</td>
<td>z</td>
<td>64<em>64</em>64</td>
<td>23.2</td>
<td>31.5</td>
</tr>
<tr>
<td>(5,3)</td>
<td>zy</td>
<td>64<em>64</em>64</td>
<td>51.9</td>
<td>28.0</td>
</tr>
<tr>
<td>(5,3)</td>
<td>zyx</td>
<td>64<em>64</em>64</td>
<td>101</td>
<td>25.1</td>
</tr>
<tr>
<td>(5,3)</td>
<td>zyxzgyx</td>
<td>32<em>32</em>32</td>
<td>1200</td>
<td>14.4</td>
</tr>
<tr>
<td>(5,3)</td>
<td>zyxzgyx</td>
<td>64<em>64</em>64</td>
<td>655</td>
<td>17.0</td>
</tr>
<tr>
<td>(5,3)</td>
<td>zyxzgyx</td>
<td>128<em>128</em>128</td>
<td>252</td>
<td>21.1</td>
</tr>
<tr>
<td>(9,7)M</td>
<td>zyxzgyx</td>
<td>64<em>64</em>64</td>
<td>1130</td>
<td>14.6</td>
</tr>
</tbody>
</table>

Table 7.1. Error size depending on the number of wavelet transformations, block size and length of wavelet filter.
The results from table 7.1 are fairly straightforward, namely larger blocks, less number of transformations and shorter filters will lead to smaller errors. Using the (5,3) wavelet appears to be a good choice of wavelet since it is has a very short support.

Since blocking is used in JPEG2000 there are numerous suggestions for how to decrease the block artifacts near the boundaries. Some of them are overlapping blocks [40], pre- and post-filtering [41], using odd block sizes [42] or using point symmetric extensions [43]. The two first introduce extra memory requirements and extra computational complexity whereas the third limits the flexibility of the implementation. Therefore it was chosen in this implementation to use point symmetric extension instead of the normal symmetric extension for reducing block artifacts [43] also shows that a point symmetric extension will lead to a non-expanding signal. For a signal \( x[n] \), where \( n = 0, 1, \ldots, N - 1 \), point symmetric extension is defined as

\[
x[i] = \begin{cases} 
2x[0] - x[-i] & i < 0 \\
\end{cases}
\]

The decrease in the errors introduced by blocking by using point symmetric extension (PSE) instead of the normal whole sample symmetric extension (WSSE) has been examined. Once again a CT stack consisting of 256 images with 256*256 pixels per image and 12 bpp was used to perform the computations on. In table 7.2 we can see the results of this and how the point symmetric extension decreases the error introduced by blocking.

A simple figure can help us to understand why the point symmetric extension reduces the block artifacts. In figure 7.5(a) we can see a signal where the block boundary is located at a sharp edge in the signal. With the normal symmetric extension this will introduce errors in the low-pass data near the boundary i.e. the boundary points from the two blocks will not converge, as can be seen in figure 7.5(b). If we instead use point symmetric extension the boundary points from the two blocks will converge much better, see figure 7.5(c).
7.4 Achieving fast wavelet transformations

Table 7.2. Error reduction due to the use of point symmetric extension instead of whole sample symmetric extension.

<table>
<thead>
<tr>
<th>Wavelet filter</th>
<th>Transform order</th>
<th>PSE/WSSE</th>
<th>MSE</th>
<th>PSNR [dB]</th>
</tr>
</thead>
<tbody>
<tr>
<td>(5,3)</td>
<td>zyx</td>
<td>WSSE</td>
<td>101</td>
<td>25.1</td>
</tr>
<tr>
<td>(5,3)</td>
<td>zyx</td>
<td>PSE</td>
<td>39.1</td>
<td>29.2</td>
</tr>
<tr>
<td>(5,3)</td>
<td>zyxzyx</td>
<td>WSSE</td>
<td>655</td>
<td>17.0</td>
</tr>
<tr>
<td>(5,3)</td>
<td>zyxzyx</td>
<td>PSE</td>
<td>412</td>
<td>19.0</td>
</tr>
</tbody>
</table>

Figure 7.5. Reducing block artifact with the help of point symmetric extension (solid line = signal, dotted line = extension, dashed line = low-pass data)

7.4 Achieving fast wavelet transformations

The execution time of wavelet transformations does not only consist of the actual computation of the wavelet coefficients but also of read and write operations, i.e. moving data between different memories and registers. As we have already mentioned, data is stored sequentially on disc, thus an image stack/block is stored in the order of the rows and the images, first pixel first row first image, second pixel first row first image, . . . , last pixel last row last image. This will cause major time issues when performing transformations along the columns and images of a
block if not the whole block can be placed in the cache memory. Increasing the speed of wavelet transformations consists of two parts:

- Decrease computation time
- Optimize access patterns of data

Decreasing the computation time can be done in various ways. We have already shown that taking advantage of the inherited non-linearity in integer-to-integer transforms can help us to reduce the number of wavelet transformations. Computation time is also affected by the number of lifting steps in the transform. In [44] the (5,3) wavelet is shown to be one of the wavelets with the least computational complexity. The (5,3) wavelet is also shown to have a good visual quality. This seems to further support the idea of using the (5,3) wavelet, see also section 7.3. Other suggestions for decreasing computation time are using packed integer-to-integer transforms [45] or using streaming SIMD extensions [46]. The basic idea of these two suggestions is to perform several computations simultaneously by loading shorter integers (in terms of number of bits) into longer integers or registers (packing integers). The only problem being that packed integers as suggested in [45] are not lossless and instructions for utilizing streaming SIMD extensions are only available in C and C++ (not in C# which was used for this implementation).

There exist many suggestions for optimizing the access patterns in wavelet transforms that are based on 2D wavelet transforms and are especially effective for performing a wavelet transform along the rows and then along the columns. Optimizing this transform pattern has resulted in a line-based wavelet transform, see [40] and [47]. Another suggestion for decreasing the number of read and write instructions can be found in [48] where as many as possible transform steps along one transform direction is performed at the same time. Although both of these suggestions contain useful insights, none of them has been used in this implementation since they both call for a fixed set of inverse transform steps to be used. In [49] we can find what seems to be a more useful suggestion, namely blocking. [49] suggests that in order to optimize the consequences of data stored sequentially, blocks should have a large number of columns, less rows and very few images. For our usage and wavelet packet structure this suggestion is of little help since it is preferable for us to have a block with equal side lengths. Choosing a small block size will, however, help in improving the access pattern of data. Finding the right size of the blocks is difficult but performed studies show that a block size of 32*32*32 seems to yield a dataset where inverse transformations along the rows (which have the best access patterns) needs approximately the same time as

---

1The line-based wavelet transform also requires the use of the normal pyramid structure, which is not used in the suggestion. The idea of performing as many levels of decomposition at the same time whilst the data is held in the cache memory could of course be useful for the analysis process since the transform order is set. This can, however, not be utilized in the synthesis part since we do not know in advance how many reconstructions steps the practitioner will need at the time, i.e. will the inverse transform steps be xyzzy and then zzz or xy and then xyzzy? Since the execution time of the synthesis process is of greatest relevance it seems to be of little interest to use this.

2The same number of columns and rows are important for the ROI-viewing or, to be more accurate, the precision of the ROI-viewing.
inverse transformations along the images\(^3\) (which have the worst access patterns). A smaller block size could of course be chosen as well but with the risk of dividing the whole volume into many small pieces and thereby causing an increased computation time. It was also tried to use blocks with the same number of columns and rows but with less images as suggested in [49]. Performance tests showed however no or little gain.

### 7.5 Organizing wavelet coefficients on disc

One of the objectives of this thesis was to investigate how to organize the wavelet coefficients on the disc. The idea is to arrange the coefficients on the disc so that, whilst reading data from the disc, as much data as possible can be read sequentially in order to skip time consuming seek operations. Since it was decided for this implementation to go with a wavelet packet structure that is intended to follow the inverse transform order, it was chosen to store the wavelet coefficients in the order of the subbands. This approach is very intuitive since it is known that only whole subbands will be transmitted and in which order. Organizing the wavelet coefficients after each transformation in this manner can also help to improve the speed of the transformations since it will improve the access patterns of the data.

![Memory array diagram](image)

**Figure 7.6.** 1D example of how the data is rearranged after a transformation.

### 7.6 Handling the dynamic coefficient interval

A problem that has not been mentioned so far (when it comes to wavelet transformations) is that the coefficient interval grows whilst performing wavelet transformations. This causes a problem since this will increase the needed memory

\(^3\)The transformations along the rows are 10% faster than the transformations along the images of the block.
Implementation of wavelet packet structure

for storage of the data. Few papers in the literature discuss this problem since most of them use some sort of compression where this fact causes no problem. However, [13] suggests an integer transform that possesses a property of precision preservation to avoid this. The suggested integer transform is basically built in the same way as the integer-to-integer transform presented in chapter 4 and thus it can also utilize this property of precision preservation. The property of precision preservation uses 2-complement notation and overflow computations to preserve the precision, i.e. letting very large positive numbers become negative and very large negative numbers become positive. Although this can be used for perfect reconstruction it has some limitations whilst viewing intermediate reconstructions where air (small number) might appear as bone (large number). Thus this property is practically useless since the idea of the proposed representation is to view intermediate reconstructions.

The problem of the dynamic coefficient interval is dependent on the precision of the used data, i.e. 8, 12 or 16 bpp. Since only data with 12 bpp was used in this thesis (which is very common for data from CT examinations), it was worth investigating how the coefficient interval grows when the (5,3) wavelet was applied to two datasets\(^4\). The first dataset was a CT stack and the second dataset was an MR stack, both consisting of 256 images with 256*256 pixels per image and 12 bpp in the interval \([0, 4095]\). The reason for using two datasets from different imaging modalities was their different noise characteristics.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Transform order</th>
<th>min val</th>
<th>mean &lt; 0</th>
<th>% &lt; 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>CT</td>
<td>z</td>
<td>−3961</td>
<td>−19</td>
<td>29</td>
</tr>
<tr>
<td>CT</td>
<td>zz</td>
<td>−4382</td>
<td>−27</td>
<td>42</td>
</tr>
<tr>
<td>CT</td>
<td>zzz</td>
<td>−4382</td>
<td>−35</td>
<td>48</td>
</tr>
<tr>
<td>CT</td>
<td>zzzz</td>
<td>−4382</td>
<td>−33</td>
<td>52</td>
</tr>
<tr>
<td>CT</td>
<td>zzzzy</td>
<td>−4382</td>
<td>−32</td>
<td>53</td>
</tr>
<tr>
<td>CT</td>
<td>zzzzyx</td>
<td>−4382</td>
<td>−32</td>
<td>54</td>
</tr>
<tr>
<td>CT</td>
<td>zzzzyxy</td>
<td>−4382</td>
<td>−32</td>
<td>54</td>
</tr>
<tr>
<td>MR</td>
<td>z</td>
<td>−3742</td>
<td>−74</td>
<td>27</td>
</tr>
<tr>
<td>MR</td>
<td>zz</td>
<td>−3742</td>
<td>−90</td>
<td>39</td>
</tr>
<tr>
<td>MR</td>
<td>zzz</td>
<td>−3742</td>
<td>−90</td>
<td>45</td>
</tr>
<tr>
<td>MR</td>
<td>zzzy</td>
<td>−3742</td>
<td>−95</td>
<td>48</td>
</tr>
<tr>
<td>MR</td>
<td>zzyx</td>
<td>−3742</td>
<td>−93</td>
<td>50</td>
</tr>
<tr>
<td>MR</td>
<td>zzyxy</td>
<td>−3742</td>
<td>−93</td>
<td>51</td>
</tr>
<tr>
<td>MR</td>
<td>zzyzyx</td>
<td>−3742</td>
<td>−93</td>
<td>51</td>
</tr>
</tbody>
</table>

Table 7.3. Min val is the smallest pixel value which is smaller than 0. Mean < 0 is the mean value of all pixels smaller than 0. % < 0 is the percentage of all pixels with a value less than 0.

\(^4\)In [44] the (5,3) wavelet is showed to be one of the better wavelets in terms of growth of the coefficient interval, better meaning that it does not grow so much. This is another reason for using the (5,3) wavelet.
7.6 Handling the dynamic coefficient interval

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Transform order</th>
<th>max val</th>
<th>mean &gt; 4095</th>
<th>% &gt; 4095</th>
</tr>
</thead>
<tbody>
<tr>
<td>CT</td>
<td>z</td>
<td>5027</td>
<td>4305</td>
<td>0</td>
</tr>
<tr>
<td>CT</td>
<td>zz</td>
<td>5085</td>
<td>4298</td>
<td>0</td>
</tr>
<tr>
<td>CT</td>
<td>zzz</td>
<td>4710</td>
<td>4295</td>
<td>0</td>
</tr>
<tr>
<td>CT</td>
<td>zzzzy</td>
<td>4851</td>
<td>4351</td>
<td>0</td>
</tr>
<tr>
<td>CT</td>
<td>zzzzyx</td>
<td>5007</td>
<td>4332</td>
<td>0</td>
</tr>
<tr>
<td>CT</td>
<td>zzzzyxy</td>
<td>4851</td>
<td>4351</td>
<td>0</td>
</tr>
<tr>
<td>CT</td>
<td>zzzzyxyx</td>
<td>5039</td>
<td>4431</td>
<td>0</td>
</tr>
<tr>
<td>MR</td>
<td>z</td>
<td>4408</td>
<td>4177</td>
<td>0</td>
</tr>
<tr>
<td>MR</td>
<td>zz</td>
<td>4243</td>
<td>4172</td>
<td>0</td>
</tr>
<tr>
<td>MR</td>
<td>zzz</td>
<td>–</td>
<td>–</td>
<td>0</td>
</tr>
<tr>
<td>MR</td>
<td>zzzzy</td>
<td>4176</td>
<td>4176</td>
<td>0</td>
</tr>
<tr>
<td>MR</td>
<td>zzzzyx</td>
<td>4219</td>
<td>4206</td>
<td>0</td>
</tr>
<tr>
<td>MR</td>
<td>zzzzyxy</td>
<td>4338</td>
<td>4254</td>
<td>0</td>
</tr>
<tr>
<td>MR</td>
<td>zzzzyxyx</td>
<td>4245</td>
<td>4245</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 7.4. Max val is the largest pixel value which is larger than 4095. Mean > 0 is the mean value of all pixels larger than 4095. % > 4095 is the percentage of all pixels with a value larger than 4095.

Comparing the results in tables 7.3 and 7.4 we can see no significant difference between the two intervals although the interval for the CT stack is a bit larger at both ends. Noteworthy is that the lower end of the intervals is approximately constant for the last five or six transformations. This is due to the fact that it is the detail coefficients that affect the lower end of the interval. Since the data used for the computations of the detail coefficients becomes smoother and smoother after each transformation, the detail coefficients will not be large. The upper end of the interval is on the other hand fairly stable for all transformation steps. Looking at the tables 7.3 and 7.4 we see that it is safe to say that 16 bits are sufficient for data with 12 bpp. This is rather soothing since for data with 12 bpp 16 bits are allocated for each pixel, thus there is no increase in storage space needed. However, this is not the case for data with 8 bpp or 16 bpp and thus calls for another solution (although not provided in this thesis).
Chapter 8

Results of the proposed wavelet representation

In this chapter we present some results from using the proposed wavelet representation and its implementation.

8.1 Memory allocation

One of the reasons for blocking was to decrease the amount of data that must be held at the same time in the work memory. If we look at the pseudo code of the implementation in section 7.1 it is obvious that we only need to hold a fixed number of wavelet blocks and one image at the same time in the memory. The number of wavelet blocks are dependent on the size of the blocks and the size of the image. Say that we have images with 512*512 pixels per image and a preferred block size of 32*32*32 coefficients, then we would need 16*16 blocks. The amount of needed memory during the analysis is therefore not dependent on the number of images in the stack but only on the size of the images and the preferred block size. For the case of images with 512*512 pixels per image and a preferred block size of 32*32*32 coefficients we need approximately 33 MB. This is under the assumption that we use four bytes to represent each coefficient. Of course there is a small amount of data needed for the metadata of the wavelet blocks but this is negligible for us. Important to note is that this is the theoretical memory allocation. Since this implementation was done in C# the user has no real control of when allocated memory is released. This is instead controlled by the garbage collector. Thus observing the actual memory allocation using this implementation is likely to reveal something else than stated here.

The memory allocation for the synthesis process is practically the same under the assumption that blocks that do not contribute to the image to be reconstructed are dismissed. Dismissing data comes with the risk of being forced to request the data once again from the server if the practitioner wishes to view the already dismissed data again. This problem is however not specific for our wavelet repre-
sentation and is therefore not considered any further.

8.2 Computation time

In order to measure the computation time a CT stack consisting of 512 images with 512*512 pixels per image and 12 bpp was decomposed and fully reconstructed. As a timing software JetBrains dotTrace 2.0 was used. The time measurements were performed on a Intel P4 2.66 GHz processor with 1 GB RAM and with a 7200 rpm HD. Each measurement was conducted ten times. The results were then used to compute a mean and a standard deviation. Alongside with these results the fastest results are also presented\(^1\).

<table>
<thead>
<tr>
<th>Method</th>
<th>Mean [ms]</th>
<th>Std [ms]</th>
<th>Fastest [ms]</th>
</tr>
</thead>
<tbody>
<tr>
<td>insertImageIntoWaveletBlocks</td>
<td>22183</td>
<td>1592</td>
<td>19318</td>
</tr>
<tr>
<td>transformWaveletBlocks</td>
<td>16658</td>
<td>3927</td>
<td>9186</td>
</tr>
<tr>
<td>writeWaveletBlocksOnDisc</td>
<td>30329</td>
<td>1104</td>
<td>28137</td>
</tr>
</tbody>
</table>

Table 8.1. Time results for decomposing a CT stack.

<table>
<thead>
<tr>
<th>Function</th>
<th>Mean [ms]</th>
<th>Std [ms]</th>
<th>Fastest [ms]</th>
</tr>
</thead>
<tbody>
<tr>
<td>requestDataFromServer</td>
<td>42440</td>
<td>2391</td>
<td>36217</td>
</tr>
<tr>
<td>inverseTransformWaveletData</td>
<td>8496</td>
<td>185</td>
<td>8000</td>
</tr>
<tr>
<td>reconstructImageFromDataAtClient</td>
<td>27845</td>
<td>1129</td>
<td>27360</td>
</tr>
</tbody>
</table>

Table 8.2. Time results for fully reconstructing a CT stack.

There are a few things that should be commented before we start discussing the results in tables 8.1 and 8.2. The two methods insertImageIntoWaveletBlocks and writeWaveletBlocksOnDisc basically perform read respectively write operations on the disc and are therefore dependent on the speed of the HD. Time measurements have also been conducted on other systems with much faster HDs and a substantial decrease in the execution time for the read and write operations was observed. In section 2.1 we said that in the normal setup of a PACS the archive server is equipped with high speed discs and therefore the time results for the two methods insertImageIntoWaveletBlocks and writeWaveletBlocksOnDisc are not representative for a real system. The method requestDataFromServer would also have a completely different setup in a real setup, since here the built-in class List in C# has been used to store the metadata of the wavelet blocks whereas in

\(^{1}\)Only time results from methods implemented by the author is presented.
a real setup a database would be used. Also this method uses read operations on
the disc and is therefore affected by the speed of the HD.

The two most interesting methods are naturally `transformWaveletBlocks`
and `inverseTransformWaveletData`. Observing just these two methods would
give the `transformWaveletBlocks/coder` a performance of 8.06 Msample/s and
the `inverseTransformWaveletData/decoder` a performance of 15.8 Msample/s.
Noteworthy here is that the two methods theoretically should have the same per-
formance since both execute the same operations, although in reversed order\(^2\).
We can also observe that `transformWaveletBlocks` has a fastest time which
is close to the mean of `inverseTransformWaveletData`. Interesting as well is
that `inverseTransformWaveletData` is very stable in its execution time whereas
`transformWaveletBlocks` is not stable.

\(^2\)The reason for the difference between the performance of the coder and decoder is difficult to
explain without more comprehensive performance tests. Plausible reasons are how the garbage
collector acts or difference in write and read operations.
Results of the proposed wavelet representation
Chapter 9

Discussion, conclusion and future work

This chapter presents a discussion of the proposed wavelet representation and its implementation followed by a conclusion. Some suggestions for future work ends the chapter and this thesis.

9.1 Discussion

We began this thesis with the aim of finding an efficient representation for medical image stacks to be used in a PACS and that would support the following features:

- Lossless viewing (perfect reconstruction)
- Arbitrary 2D viewing (random access of original images)
- Progressive transmission (data received by client should be reusable)
- Scalable resolution (ability to change resolution)
- Thick slab viewing (ability to view thicker slices)
- ROI-viewing (ability to cut out and view a certain region from the original image without transmitting the whole image)

These features were not the only important factors for an efficient representation but also its implementation was of great relevance. Important criteria for a feasible implementation were the amount of data transmitted between server and client, the computation time for processing the data (both at server and client) and the memory allocation.

Compared with the Gaussian pyramid representations in section 2.3 it is foremost the progressive transmission, the thick slab viewing, the scalability in resolution and the ROI-viewing that are the advantages of our proposed wavelet representation. Progressive transmission offers the possibility to send only data
that is necessary for a certain reconstruction, i.e. for a certain position and resolution, and to send information only once. This is a major advantage when compared to the Gaussian pyramids where a whole new set of data (containing already sent information) has to be sent when the resolution is changed whereas in our representation we only need to add data (i.e. missing information) to already transmitted data in order to change the resolution. This means that we will never send more data than what was contained in the original representation but often less if the other features are taken advantage of. The Gaussian pyramids can as well be used to provide the practitioner with features such as thick slab viewing. This comes though with a computational burden for the client, as discussed in section 2.2, which for large image stacks drastically reduces the usability of this feature and other important features. The reason for the suggested representation to be more effective in terms of computational complexity is that for large image stacks the amount of allocated memory is not dependent on the number of images but only on the size of the images and the size of the blocks.

The suggested wavelet representation has a major disadvantage though and that is when the practitioner wishes to view the original images in full resolution. It is questionable though whether this case is a reasonable one for comparison of the different representations\(^1\).

In section 5.2 we discussed some 3D codecs that possessed some of the above mentioned features but none that supported them all. The strength of these codecs are of course their compression, which both decreases needed storage space and the amount of data that is transmitted between server and client. However, the compression is also the disadvantage since it comes with the cost of coding and decoding. The suggested codec in [38] that claims to be one of the fastest, approximately 3 times faster then JPEG2000, has a decoding performance of 4.19 Msamples/s. This should be compared with the suggested representation and its implementation, which has a decoding performance of 15.8 Msample/s.

The proposed wavelet representation appears to outperform both the Gaussian pyramids and the 3D codecs in terms of how well it supports the workflow of a practitioner whilst viewing large image stacks in the original plane\(^2\). It also has advantages when comparing the amount of transmitted data and the memory allocation.

The most important issue though is whether the implementation of the proposed wavelet representation is fast enough. In section 8.2 it was shown to have a decoding performance of 15.8 Msample/s, which corresponds to 60 or 15 images/s with 512\(^2\)512 or 1024\(^2\)1024 pixels per image. However, these numbers do not tell the whole truth. The decoding is not something that occurs continuously per sample that is transmitted to the client but is performed on whole blocks. In this implementation the decoding starts first when all relevant blocks have been received at the client, thus there will be a pause whilst requesting and receiving

---

\(^1\)The reason for finding another representation was partly based on the statement that it is not necessary for the practitioner to view all original images in full resolution.

\(^2\)This is however not entirely true since most practitioners today cine through all images at full resolution. What this means is that the suggested representation supports the workflow of using low resolution images to navigate in and then have high resolutions images for diagnostic purposes (thick slab viewing).
blocks from the server and inverse transforming/decoding the blocks. For 16*16 blocks of the size 32*32*32 coefficients per block this corresponds to a processing time of 0.55 s. This means that after receiving the data from the server it takes approximately 0.55 s before the data is inverse transformed and can be used to reconstruct 32 images. We must remember though that the data is still partitioned into smaller blocks and thus requires some sort of algorithm for concatenating the data from the different blocks into images. In the suggested implementation, see section 8.2, this algorithm requires 0.055 s per image. This implementation is however not optimal and therefore this time should not be taken as representative for an optimized implementation. Using the time for requesting and receiving data from table 8.2 we see that it takes approximately 2.65 s to receive data from the server and another 0.55 s to inverse transform it. Once this is done we can show 32 images at a rate of 20 images/s. Thus this implementation has a periodicity of five seconds, three seconds where nothing happens (at least in the eyes of the user) and two seconds where 32 images are displayed. Hardly an implementation that someone would use.

In section 8.2 we said that the time measurements of how long time it takes to request and receive data from the server to the client were not representative for a real setup. We can however approximate this time by assuming that the database requests are handled instantaneously by the server and we thus only consider the time needed for transmitting the data between server and client. Assuming a 100 Mbps network, a bandwidth usage of 50 % and data with 16 bpp we have a network transmission performance of 3.125 Msample/s. Taking into account that the data is sent block-wise we could change the implementation so that the client starts transforming data as soon as a whole block has been received. Since the performance of the decoder is about five times better than the performance of the network, the bottleneck will be the network and not the decoder. The pauses that will appear whilst requesting and receiving data from the server to the client can be avoided or decreased by pre-loading data from the server. The client can simply observe which images the practitioner is viewing and can then before the practitioner reaches the last image of the already received data request new data from the server and thus avoid long waiting times.

We have already objected against using the case of complete reconstruction of an image stack for evaluating the proposed wavelet representation and its implementation. A better scenario would be to evaluate the case for thick slab viewing. Assume that we wish to view an image stack consisting of 512 images with 512*512 pixels per image and 12 bpp (remember though that 16 bits are allocated per sample) with full planar resolution and an axial resolution of one eighth of the original axial resolution. Whilst using a normal setup this would mean that the whole stack has to be transmitted and then averaged before it can be viewed. If we once again assume a 100 Mbps network and a bandwidth usage of 50 % we would need roughly 45 s to transmit the whole dataset. For our proposed representation we would only need 5.5 s to transmit the required data and another 1 + 3.5 s to inverse transform the data and to concatenate the inverse transformed data into images, which makes a total of 10 s. This example shows that the proposed wavelet representation is well suited for thick slab viewing.
In this thesis a wavelet representation is proposed that is scalable in resolution although many wavelet based codecs are scalable in quality. It is important to understand why resolution scalability is preferable compared to quality scalability, even though the image coder might consider the two as the same. First of all scalability in resolution is often easier to understand and accept whilst viewing medical images. The degradation of the visual quality due to low resolution data is acceptable whereas claims for equal visual quality for low quality data is perceived suspiciously and therefore seldom accepted by practitioners. Often the best quality is requested regardless of the resolution. Second of all in terms of computational complexity for processing the data it is preferable to work with scalability in resolution since lower resolution will mean less data to process whereas scalability in quality will not affect the amount of data that has to be processed.

9.2 Conclusion

In this thesis the use of a wavelet representation for efficiently representing a medical image stack is proposed. The reasons for using a wavelet representation are the inherit multiresolution representation (scalable resolution) and the suitability for progressive transmission. Integer-to-integer wavelet transforms are implemented with the lifting scheme and rounding in order to have perfect reconstruction (lossless viewing). The (5,3) wavelet belonging to the biorthogonal Daubechies wavelet family is used as wavelet filter. It is used because of its low computational complexity and its visual qualities. The rounding operation in the integer-to-integer transform causes the transform to be non-linear, which limits the flexibility of the inverse transform order, but this can be utilized for creating a certain wavelet packet structure with a lower computational burden than normal by avoiding unnecessary wavelet transformations. Together with a set transform order this wavelet packet structure is suitable for thick slab viewing, in case of transversal slices a suitable transform order would be zzyxyxz. To achieve arbitrary 2D viewing (random access) and ROI-viewing the medical image volume is partitioned into smaller blocks of the size 32*32*32 before the stack is wavelet transformed. Blocking also helps in decreasing the computation time of the wavelet transforms by improving access patterns of the data. The non-linearity has further been utilized for how to organize the wavelet coefficients on the disc in order to achieve efficient access patterns of the wavelet coefficients for reconstruction. The wavelet coefficients are on the disc ordered according to the order of the subbands.

The proposed wavelet representation was implemented in order to evaluate the feasibility of this representation. Performance analysis showed that the wavelet transforms can be done at a sufficiently high rate. Other operations such as read, write and request data operations were difficult to evaluate since the implementation of this operations does not match a real setup. However, with the aid of the time results for the wavelet transforms it has been shown that a wavelet representation is likely to achieve an acceptable performance in a real setup. The real advantage of the proposed wavelet representation is achieved when using intermediate reconstructions for navigating in the stack and full reconstructions for
9.3 Future work

The following ideas are suggested for future work:

- Implement the proposed wavelet representation in a real setup. The time measurements performed in this thesis are not sufficient for rejecting or accepting the performance of this representation. Therefore further time measurements on a real setup are necessary.

- Find ways to improve the computation time. Even though the presented implementation has a reasonably high decoding performance it is important to further improve it. Some suggestions for how this can be done is by utilizing the rising multi-core technology (most of the computations can be done in parallel) or utilizing streaming SIMD extensions as suggested in [46] (i.e. write the implementation in C or C++).

- Implement wavelet packet structure #2. The real advantage of using a wavelet representation can be found by using wavelet packet structure #2 since it is so flexible in terms of reconstruction. Since it has a higher computational burden than the proposed representation, it is important to first improve the computation time of the wavelet transformations.

- Handle the dynamic coefficient interval. This has been avoided in this thesis and is especially important for data with 8 or 16 bpp. A way of avoiding this problem is of course to use compression. Important to evaluate is also how the dynamic coefficient interval affects the diagnostic work whilst using intermediate reconstructions.

- Implement lossless compression. Although we have discussed that the decoding time is a major issue for a PACS in a high work pace environment, it is still important to find ways of decreasing the needed storage and the amount of data transmitted between server and client.
Bibliography


Upphovsrätt
Detta dokument hålls tillgängligt på Internet — eller dess framtida ersättare — under 25 år från publiceringsdatum under förutsättning att inga extraordinära omständigheter uppstår.
Tillgång till dokumentet innebär tillstånd för var och en att läsa, ladda ner, skriva ut enstaka kopior för enskilt bruk och att använda det oförändrat för icke-kommersiell forskning och för undervisning. Överföring av upphovsrätten vid en senare tidpunkt kan inte upphäva detta tillstånd. All annan användning av dokumentet kräver upphovsmannens medgivande. För att garantera äktheten, säkerheten och tillgängligheten finns det lösningar av teknisk och administrativ art.
Upphovsmannens ideella rätt innefattar rätt att bli nämnd som upphovsman i den omfattning som god sed kräver vid användning av dokumentet på ovan beskrivna sätt samt skydd mot att dokumentet ändras eller presenteras i sådan form eller i sådant sammanhang som är kränkande för upphovsmannens litterära eller konstnärliga anseende eller egenart.
För ytterligare information om Linköping University Electronic Press se förlagets hemsida http://www.ep.liu.se/

Copyright
The publishers will keep this document online on the Internet — or its possible replacement — for a period of 25 years from the date of publication barring exceptional circumstances.
The online availability of the document implies a permanent permission for anyone to read, to download, to print out single copies for his/her own use and to use it unchanged for any non-commercial research and educational purpose. Subsequent transfers of copyright cannot revoke this permission. All other uses of the document are conditional on the consent of the copyright owner. The publisher has taken technical and administrative measures to assure authenticity, security and accessibility.
According to intellectual property law the author has the right to be mentioned when his/her work is accessed as described above and to be protected against infringement.
For additional information about the Linköping University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its www home page: http://www.ep.liu.se/

© Daniel Forsberg