BRICS RS-99-34 F. F. Rodler: Wavelet Based 3D Compression for Very Large Volume Data

BRICS

Basic Research in Computer Science

Wavelet Based 3D Compression for Very Large Volume Data Supporting Fast Random Access

Flemming Friche Rodler

BRICS Report Series **RS-99-34**

ISSN 0909-0878 **October 1999**

Copyright © 1999, Flemming Friche Rodler.
BRICS, Department of Computer Science
University of Aarhus. All rights reserved.

Reproduction of all or part of this work
is permitted for educational or research use
on condition that this copyright notice is
included in any copy.

See back inner page for a list of recent BRICS Report Series publications.
Copies may be obtained by contacting:

BRICS
Department of Computer Science
University of Aarhus
Ny Munkegade, building 540
DK–8000 Aarhus C
Denmark

Telephone: +45 8942 3360
Telefax: +45 8942 3255
Internet: BRICS@brics.dk

BRICS publications are in general accessible through the World Wide Web and anonymous FTP through these URLs:

`http://www.brics.dk`
`ftp://ftp.brics.dk`
This document in subdirectory `RS/99/34/`

Wavelet Based 3D Compression for Very Large Volume Data Supporting Fast Random Access

Flemming Friche Rodler

October 29, 1999

Abstract

We propose a wavelet based method for compressing volumetric data with little loss in quality. The method supports fast random access to individual voxels within the compressed volume. Such a method is important since storing and visualising very large volumes impose heavy demands on internal memory and external storage facilities making it accessible only to users with huge and expensive computers. This problem is not likely to become less in the future. Experimental results on the CT dataset of the Visible Human have shown that our method provides very high compression rates with fairly fast random access.

1 Introduction

Volumetric datasets tend to demand enormous memory requirements. To exemplify we introduce the Visible Human datasets which were acquired around the mid 90's by the National Library of Medicine's (NLM). The datasets consist of computerised tomography (CT), magnetic resonance imaging (MRI) and colour cryosection images of representative male and female cadavers [10]. The effort was done in order to create three dimensional representations of the human body to further development in health, education, research and treatment applications. In the male data set, 1871 cross-sectional images taken at 1mm intervals exist for each modality (CT,MRI and cryosectional), making up about 15 Gbytes of volumetric data. The cryosectional images of the Visible Female consist of images taken at one-third the spacing of the male resulting in a dataset of about 40 Gbytes.

Because of memory issues, the working of volumes having these proportions is not practical on ordinary workstations and personal computers, even taking the rapid development of larger memory and storage capabilities into account. This makes the data available only to users with access to huge and expensive computers. To mend this and in order to make the data available to a wider audience, data compression can be utilised to reduce not only external storage but also memory requirements. What is needed is a method allowing the user to load a compressed version of the volume into a small amount of memory and enable him to access and visualise it as if the whole uncompressed volume was present. Such a compression scheme must necessarily allow fast random access to individual voxels within the compressed volume.

Until recently, most of the research effort in lossy compression has mainly been focusing on lossy compression of still images or time sequences of images such as movies. The aim of these methods is to obtain the best compression rate while minimising the distortion in the reconstructed images. Often this limits the random accessibility. The reason being that often these compression schemes employ variable-bitrate techniques such as Huffman (used in JPEG and MPEG) and aritmethic coders [21] or differential encoders such as the adaptive differential pulse code coder [21].

Recently, though, techniques dealing with the issue of compression with fast random access in volumetric data have been emerging and in this paper we propose a new novel method.

The rest of this report is divided as follows. In Section 2 we present the necessary preliminaries which will explain the underlying wavelet theory used throughout the paper. The theory will be presented by means of multiresolution analysis, since this is a convenient framework for describing not only wavelets and their properties but also how the fast wavelet transform is computed. Section 3 contains a short survey of previous work done in this area, while Section 4 presents our contribution - a newly developed scheme that allows fast random access to individual voxels within the compressed data. Finally we make concluding remarks in Section 5.

2 Preliminaries

In this section we present the underlying mathematical foundations of wavelet theory and hierarchical decomposition. We start by presenting the multiresolution analysis (MRA) concept which can be used as a method for describing hierarchical bases. This will lead to the definition of wavelets and the fast wavelet transform.

2.1 Multiresolution Analysis

The theory of multiresolution analysis was initiated by Stéphane Mallat [13] and Yves Meyer[1].

2.1.1 Definition

Definition 1 (Multiresolution Analysis) *A multiresolution analysis (MRA) is a sequence* $(V_j)_{j\in Z}$ *of closed subspaces of* $L^2(R)$ *satisfying the following five properties:*

1. $\forall j \in Z : V_j \subset V_{j+1}$
2. $\bigcap_{j\in Z} V_j = \{0\}$
3. $\overline{\bigcup_{j\in Z} V_j} = L^2(R)$
4. $f(x) \in V_j \Leftrightarrow f(2x) \in V_{j+1}$
5. $\exists \phi \in V_0$ *such that* $\{\phi(x-k)\}_k$ *constitutes an orthonormal basis for* V_0.

The first and third property of the definition gives us the approximation feature of the MRA. The third states that any function in $L^2(R)$ can be approximated arbitrarily well by its projection onto V_j for a suitably large j, i.e. if P_{V_j} denotes the projection operator onto V_j then:

$$\lim_{j\to\infty} \|f - P_{V_j} f\| = 0. \tag{1}$$

The first property is a causality property which guarantees that an approximation at a resolution 2^j contains all the information necessary to compute an approximation at a coarser resolution 2^{j-1}. The second

[1] Yves Meyer, 1986 - Ondelettes, fonctions splines et analyses graduées. Lectures given at the University of Torino, Italy

property proves that as j tends toward $-\infty$ the projection of f onto V_j contains arbitrarily little energy, or stated differently we lose all the details of f when the resolution 2^j goes to zero:

$$\lim_{j\to-\infty} \|P_{V_j} f\| = 0. \tag{2}$$

The aspect of MRA comes with the fourth property which tells us that the resolution increases with j and that all the spaces are scaled versions of each other[2].

As a direct consequence of properties (4) and (5) we have that $\{\phi_{j,k} = 2^{\frac{j}{2}}\phi(2^j x - k)\}_{k\in Z}$ constitutes an orthonormal basis for V_j. Since the orthogonal projection of f onto V_j is obtained by the following expansion:

$$P_{V_j} f = \sum_{k=-\infty}^{\infty} \langle f, \phi_{j,k}\rangle \phi_{j,k}, \tag{3}$$

we have that

$$a_j[k] = \langle f, \phi_{j,k}\rangle, \tag{4}$$

provides a discrete approximation of f at scale 2^j. The functions $\phi_{j,k}$ are called scaling functions. The requirement for the $\phi_{j,k}$'s to generate orthonormal bases for the V_j's is a bit strict and it can be shown that we need only require that they generate a Reisz basis for the MRA spaces, see e.g. [4].

2.1.2 Example

As an example of a multiresolution analysis we introduce the Haar MRA [7] where the approximations are composed as piecewise constant functions. The MRA spaces are defined as $V_j = \{f \in L^2(R) : \forall k \in Z\ ,\ f|_{[k2^{-j},(k+1)2^{-j}]} = \text{Constant}\}$ and the scaling functions are generated from the box function $1_{[0,1]}$. The properties in the definition of a MRA are easily verified.

2.2 Wavelets

The basic tenet of multiresolution analysis is that whenever there exists a sequence of closed subspaces satisfying Definition 1 then there exists an orthonormal wavelet basis $\{\Psi_{j,k}\}_{j,k}$ given by the following definition:

[2]Henceforth resolution and scale will be use interchangeably

Definition 2 (Wavelet) *A wavelet is a function* $\Psi \in L^2(R)$ *chosen such that the dyadic family:*

$$\Psi_{j,k}(x) = 2^{\frac{j}{2}}\Psi(2^j x - k), \quad k, j \in Z, \tag{5}$$

of functions constitutes an orthonormal basis for $L^2(R)$. Ψ *is often referred to as the mother wavelet since it generates the whole family.*

This family of functions is called the dyadic wavelet family since it is generated by dyadic dilates and translates of a single function Ψ. Given the above definition we can write the *dyadic wavelet transform* as:

$$C_{j,k} = \langle f, \Psi_{j,k}\rangle = \int_R f(x)\Psi^*_{j,k}(x)dx, \quad k, j \in Z, \tag{6}$$

with $\Psi^*_{j,k}(x)$ denoting the complex conjugate of $\Psi_{j,k}(x)$. The *reconstruction formula* becomes:

$$f(x) = \sum_j \sum_k C_{j,k}\Psi_{j,k}(x). \tag{7}$$

The connection to multiresolution analysis arises because the function f at scale 2^j, i.e. $P_{V_j} f$, can be written as:

$$P_{V_j} f = P_{V_{j-1}} f + \sum_k \langle f, \Psi_{j,k}\rangle \Psi_{j-1,k}, \tag{8}$$

where Ψ is a wavelet and the summation describes the "detail" necessary to go from the coarser space $P_{V_{j-1}}$ to the finer space P_{V_j}. In the following this will be formalised.

For every $j \in Z$ we define the *complement space* W_j as the orthogonal complement of V_j in V_{j+1}, i.e.: $W_j = V_{j+1} \cap V_j^{\perp}$. We can then write V_{j+1} as

$$V_{j+1} = V_j \oplus W_j, \tag{9}$$

where $\oplus$ is the orthogonal direct sum. By definition all of the spaces W_j satisfy

$$W_j \perp W_{j'} \quad \text{for } j \neq j', \tag{10}$$

since for $j > j' : W_{j'} \subset V_j \perp W_j$. By the definition of the complement spaces and by iteration it follows that for $J < j$:

$$V_j = V_J \oplus \bigoplus_{l=J}^{j-1} W_l. \tag{11}$$

Because $V_J \to \{0\}$ for $J \to -\infty$ in $L^2(R)$ this implies

$$V_j = \bigoplus_{l=-\infty}^{j-1} W_l. \tag{12}$$

Again by noticing that $V_j \to L^2(R)$ when $j \to \infty$ we get

$$L^2(R) = \bigoplus_{l=-\infty}^{\infty} W_j, \tag{13}$$

which by virtue of (10) implies that we have decomposed $L^2(R)$ into a set of mutually orthogonal subspaces. Since it is easy to prove that the complement spaces W_j inherit the scaling property:

$$f(x) \in W_j \Leftrightarrow f(2x) \in W_{j+1}, \tag{14}$$

all we need to do in order to construct a wavelet basis is to find a function $\Psi \in L^2(R)$ such that $\{\Psi(x-k)\}_{k \in Z}$ constitutes a basis for W_0. Then for a fixed $j \in Z$ we have that $\{\Psi_{j,k}\}_{k \in Z}$ is an orthonormal basis for W_j. Finally we have by means of (13) that $\{\Psi_{j,k}\}_{(j,k) \in Z^2}$ constitutes an orthonormal basis for $L^2(R)$. Finding the functions ϕ and Ψ is generally nontrivial and it requires some mathematical work.

2.2.1 Example

In the previous example we introduced the Haar MRA. The corresponding Haar wavelet is given as:

$$\Psi(x) = \begin{cases} 1 & \text{for } 0 \leq x < \frac{1}{2} \\ -1 & \text{for } \frac{1}{2} \leq x < 1 \\ 0 & \text{otherwise} \end{cases} \tag{15}$$

2.2.2 Properties

In compression and many other applications we use the ability of wavelets to efficiently approximate many different signals with few nonzero wavelet coefficients. Much of this ability can be attributed to the properties of the wavelets given in the following list.

- **Support:** The support of a wavelet is given by the following closure:

$$\text{supp}(\Psi) = \overline{\{x \in R : \Psi(x) \neq 0\}} \tag{16}$$

  If there exist $a, b \in R$ such that $\text{supp}(\Psi) \subset [a, b]$ then Ψ is said to be compactly supported. There are two main reasons why a wavelet with a small support is preferable in compression. Firstly if the function $f(x)$ that we want to compress has a singularity at x' within the support of $\Psi_{j,k}$ then $\langle f, \Psi_{j,k} \rangle$ might have large magnitude. Now if Ψ has compact support with width S then at each scale j the support of $\Psi_{j,k}$ will include x' S times. This makes it desirable to have S as small as possible. Secondly as we shall see in section 2.3.2 a small support for the wavelet implies faster decomposition and reconstruction algorithms. This is essential since we are aiming for small reconstruction times.

- **Vanishing moments:** A function Ψ is said to have n vanishing moments if the following holds true

$$\int_R x^k \Psi(x) dx = 0 \quad \text{for } k = 0, .., n - 1. \tag{17}$$

  Many signals can be approximated well piecewisely by low order polynomials of degree p. So if the analysing wavelet has $n > p$ vanishing moments this results in wavelet coefficients close to zero. Unfortunately vanishing moments come at the expense of wider support ([14, p.241-245]). In fact, for orthogonal wavelets with n vanishing moments the width of the support will be at least $2n - 1$. The Daubechies wavelets are optimal in this respect. If we expect our signals to be highly regular with only a few isolated singularities then wavelets with many vanishing moments are preferable. On the other hand wavelets with small support might be the better choice if our signals are expected to contain many singularities.

- **Smoothness:** The smoothness or regularity of a wavelet is usually measured in terms of the number of continuous derivatives it has. Smooth wavelets are important in lossy compression of images. In lossy wavelet based compression, errors are mostly introduced during quantisation of the wavelet coefficients. We see from the reconstruction formula:

$$f(x) = \sum_{j,k} C_{j,k} \Psi_{j,k}(x) \tag{18}$$

  that if the wavelet coefficient $C_{j,k}$ is changed by ϵ the error $\epsilon \Psi_{j,k}$ will be added to $f(x)$. If Ψ is smooth the introduced artefact will also be smooth and smooth artefacts are perceptually less annoying than irregular ones. Smoothness comes at the expense of larger support [14, pp 241-245].

- **Symmetry:** Wavelets that are symmetric or antisymmetric are important for several reasons. The wavelet transform is a transform over signals in $L^2(R)$. In order to transform a signal with finite support (i.e. in $L^2([0, N])$) it must be extended to $L^2(R)$. To this end several ways of performing the extension exist, many resulting in boundary effects in the wavelet domain (i.e. high coefficients) near 0 and N. This is undesirable for many applications especially compression. Symmetric or antisymmetric wavelets allow for (anti)symmetric extensions at the boundaries which partly solves the problem.

  Symmetric and antisymmetric wavelets are synthesised with filters having linear phase. Wavelets and their synthesising filters are introduced in section 2.3. The linear phase property of the filters are important for some applications.

- **Orthogonality:** Daubeschies [4] proved that except for the Haar basis there exists no symmetric or antisymmetric real orthogonal compactly supported wavelet bases. By giving up orthogonality and allowing for biorthogonal wavelet bases it is possible to construct compactly supported (anti)symmetric wavelet bases [4, 14].

- **Localisation:** Wavelets with small support or rapid decay toward zero are said to be well localised in the spatial domain. In contrast is localisation in the spectral or frequency domain which is important for some applications. The Heisenberg uncertainty principle[3] [8] gives a bound on how localised a function can be simultaneously in time and frequency:

$$\sigma_t^2 \sigma_\omega^2 \geq \frac{1}{4}, \tag{19}$$

where

$$\sigma_t^2 = \int_R (t - t_c) \, |g(t)|^2 \, dt \qquad \sigma_\omega^2 = \int_R (\omega - \omega_c) \, |\hat{g}(\omega)|^2 \, d\omega, \tag{20}$$

denotes the standard deviation from the centres of gravity given by:

$$t_c = \int_R t \, |g(t)|^2 \, dt \qquad \omega_c = \int_R \omega \, |\hat{g}(\omega)|^2 \, d\omega. \tag{21}$$

Thus good spatial localisation comes at the expense of poorer localisation in frequency. For a wavelet basis, which consists of scaled versions of the mother wavelet, this means that high frequencies (small scale) are analysed with good positional accuracy whereas low frequencies (large wavelets) are analysed more accurately in frequency. This adaption to frequency is contrary to other time-frequency analysis methods such as the windowed Fourier transform and it is an important property in e.g. sound processing and analysis.

2.3 The Fast Wavelet Transform and its Inverse

The fast wavelet transform (FWT) decomposes the signal f in the wavelet basis by recursively convolving the signal with filters H and G. Assume that we have a function $f_J \in V_J$ given by

$$f_J(x) = \sum_k a_{J,k} \phi_{J,k} \in V_J, \tag{22}$$

with

$$a_{J,k} = \langle f_J, \phi_{J,k} \rangle = \int_R f_J(x) \phi^*_{J,k}(x) dx. \tag{23}$$

[3] The Heisenberg uncertainty principle originate from quantum mechanics, but is in fact a general property of functions.

The fast wavelet transform then computes the wavelet coefficients of the discrete signal

$$a_J[k] = a_{J,k}, \tag{24}$$

where each sample of $a_J[k]$ according to (23) is a weighted average of f around a neighbourhood of f with averaging kernel $\phi_{J,k}$. We will now look at the FWT of a signal f_J.

We have $f_J \in V_J = V_{J-1} \oplus W_{J-1}$ and thus there exist sequences $a_{J-1,k}$ and $d_{J-1,k}$ such that:

$$f_J(x) = \sum_k a_{J-1,k}\phi_{J-1,k}(x) + \sum_k d_{J-1,k}\Psi_{J-1,k}(x). \tag{25}$$

Because of property (1) of an MRA the sequences $a_{J-1,k}$ and $d_{J-1,k}$ can be computed as:

$$a_{J-1,k} = \langle f_J, \phi_{J-1,k}\rangle = \int_R f(x)\phi^*_{J-1,k}(x)dx. \tag{26}$$

$$d_{J-1,k} = \langle f_J, \Psi_{J-1,k}\rangle = \int_R f(x)\Psi^*_{J-1,k}(x)dx. \tag{27}$$

Repeating on $a_{J-1,k}$ the coefficients $d_{J-2,k}, d_{J-3,k}, \ldots,$ can be computed. So the FWT successively decomposes the approximation $P_{V_j}f$ into a coarser approximation $P_{V_{j-1}}$ plus the wavelet coefficients $P_{W_{j-1}}$.

Unfortunately computing the wavelet coefficients by means of (26) and (27) would not be very efficient. We find hope in the following theorem, which is due to Mallat [13].

Theorem 1 (The fast orthogonal wavelet transform) *For an orthogonal wavelet basis there exist filters $H = \{h_n\}_n$ and $G = \{g_n\}_n$ such that:*

$$a_{j-1,k} = \sum_n h_{n-2k}a_{j,n}. \tag{28}$$

$$d_{j-1,k} = \sum_n g_{n-2k}a_{j,n}. \tag{29}$$

Similarly the inverse computation is given by:

$$a_{j,k} = \sum_n h_{k-2n}a_{j-1,n} + \sum_n g_{k-2n}a_{j-1,n}. \tag{30}$$

Proof: See [14, pp. 254-256].

Theorem 1 connects wavelets with *filter banks*. The convolution in (28) and (29) can be interpreted as filtering the signal a_j with filters H and G respectively as illustrated in Figure 1. Note that because of the $2k$ in the sum a dyadic downsampling takes place. This is important since the downsampling ensures that the data is not doubled. The inverse transform in (30) first upsamples by inserting zeros and then interpolates by filtering its input signals a_{j-1} and d_{j-1} to obtain the reconstructed signal a_j.

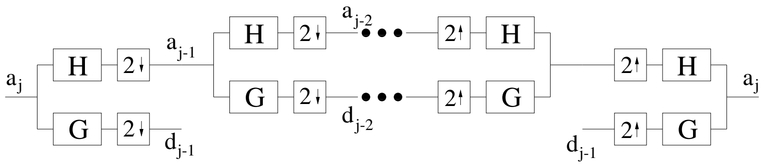

Figure 1: A 3-channel filter bank for computing a two level wavelet decomposition and its inverse.

2.3.1 Example

The Haar wavelet corresponds to two-tap filters given by $H = [\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}]$ and $G = [\frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}}]$

2.3.2 Complexity

A finite signal $a_J[k]$ cannot be decimated indefinitely. The iterative process must at least terminate when there is only one approximation coefficient a_0 left. Normally, for compression purposes only a few iterations, say 2 to 5, are applied. For L iterations the wavelet decomposition is composed of the wavelet coefficients at scale $2^{J-L} \leq 2^j < 2^J$ plus the remaining approximation coefficients at scale 2^{J-L}. The time complexity of the algorithm is easy to analyse. If we start with N samples and two filters H and G having at most K filter coefficients then:

$$KN + \frac{KN}{2} + \frac{KN}{4} + \cdots + 1 \leq 2KN,$$

is an upper bound on the number of additions and multiplications that will be performed. Table 1 shows the filter length given the support of some well known wavelets.

Wavelet family	Haar	Daubechies (dbN)	Coiflets	Symlets
Order	1	N	N	N
Support width	1	2N-1	6N-1	2N-1
Filter length	2	2N	6N	2N
Vanishing moments	1	N	2N	N

Table 1: How filter length depends on the support of some well known wave lets.

2.3.3 Initialisation

As mentioned in the beginning of section 2.3 the FWT computes the wavelet coefficients of a discrete signal $a_J[k]$ given by

$$a_J[k] = \langle f, \phi_{J,k} \rangle, \tag{31}$$

meaning that $a_J[k]$ is a local average of $f \in V_J$ around k but not precisely equal to $f(k)$. So we need to find $a_J[k]$ from f in order to start the algorithm. Often this is omitted and the algorithm is applied directly on a sampled version $f[n]$ of f. Frequently $f[n]$ is given as samples recorded by a device of finite resolution such as a CCD camera or NMR scanner that averages and samples an analog signal, so without information about the averaging kernel of the sampling device, decomposing the samples $f[n]$ is justified.

2.4 Multidimensional Bases

The above constructions have been concerned with the task of decomposing functions in $L^2(R)$. In order to analyse multidimensional functions these techniques must be extended to $L^2(R^n)$. We now illustrate how this extension is done for two dimensions, higher dimensions are analogous.

If $\{V_j\}_j$ is a MRA of $L^2(R)$ then the tensor spaces defined as $\{V_j^2 = V_j \otimes V_j\}_{j\in Z}$ constitute a separable two dimensional MRA for $L^2(R^2)$ and in the case where $\{\phi_{j,k}\}_{k\in Z}$ is an orthonormal basis for V_j, the product functions $\{\phi^2_{j,k,l} = \phi_{j,k}(x)\phi_{j,l}(x)\}_{(k,l)\in Z^2}$ form an orthonormal basis for V_j^2. As for the one dimensional case this allows for the definition of the complement spaces W_j^2. We have

$$\begin{aligned} V_{j+1}^2 &= V_{j+1} \otimes V_{j+1} \\ &= (V_j \oplus W_j) \otimes (V_j \oplus W_j) \\ &= V_j^2 \oplus [(V_j \otimes W_j) \oplus (W_j \otimes V_j) \oplus (W_j \otimes W_j)]. \end{aligned} \tag{32}$$

This shows that

$$\{\Psi^{2,\lambda}_{j,k,l} : \lambda \in \{v, h, d\}\}_{(k,l)\in Z^2}, \tag{33}$$

with the mother wavelets

$$\Psi^{2,v} = \phi(x)\Psi(y) \quad , \quad \Psi^{2,h} = \Psi(x)\phi(y) \quad , \quad \Psi^{2,d} = \Psi(x)\Psi(y), \tag{34}$$

is an orthonormal basis for W_j^2 and therefore $\{\Psi^{2,\lambda}_{j,k,l}\}_{(j,k,l)\in Z^3}$ is a orthonormal basis for $L^2(R^2)$.

The interpretation of the wavelets in terms of filters carries over from the one dimensional case and we obtain separable 2D filters

$$\begin{aligned} h[x,y] = h[x]h[y] \quad &, \quad g^v[x,y] = h[x]g[y] \\ g^h[x,y] = g[x]h[y] \quad &, \quad g^d[x,y] = g[x]g[y]. \end{aligned} \tag{35}$$

Filtering with these corresponds to filtering first along the rows of the discrete signal and then along the columns. After filtering downsampling in each direction is performed. This is illustrated in Figure 2.

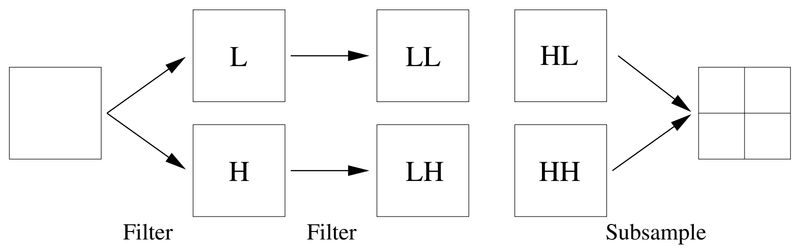

Figure 2: One decomposition level of a 2D wavelet transform.

2.4.1 Example

Using the filters of the previous example we see that the two dimensional Haar decomposition is given by

$$a_{ll} = ((a_1 + a_2)/\sqrt{2} + (a_3 + a_4)/\sqrt{2})/\sqrt{2} = (a_1 + a_2 + a_3 + a_4)/2, \tag{36}$$

$$d_{lh} = ((a_1 + a_2)/\sqrt{2} - (a_3 + a_4)/\sqrt{2})/\sqrt{2} = (a_1 + a_2 - a_3 - a_4)/2, \tag{37}$$

$$d_{hl} = ((a_1 - a_2)/\sqrt{2} + (a_3 - a_4)/\sqrt{2})/\sqrt{2} = (a_1 - a_2 + a_3 - a_4)/2, \tag{38}$$

$$d_{hh} = ((a_1 - a_2)/\sqrt{2} - (a_3 - a_4)/\sqrt{2})/\sqrt{2} = (a_1 - a_2 - a_3 + a_4)/2, \tag{39}$$

where a_i on the right side of the equations are coefficients in a 2×2 sub-block. a_{ll} is the average and the d_{xx}'s are the detail coefficients. Reconstruction is given by

$$a_1 = (a_{ll} + d_{lh} + d_{hl} + d_{hh})/2, \tag{40}$$
$$a_2 = (a_{ll} + d_{lh} - d_{hl} - d_{hh})/2, \tag{41}$$
$$a_3 = (a_{ll} - d_{lh} + d_{hl} - d_{hh})/2, \tag{42}$$
$$a_4 = (a_{ll} - d_{lh} - d_{hl} + d_{hh})/2. \tag{43}$$

Instead of dividing by 2 in Equation (36)-(39) we can divide by 4. That way the 2 in the reconstruction equation becomes 1 for easier computation.

2.5 Thresholding

In 2.2.2 we pointed out that the wavelet representation for many functions is able to concentrate most of the energy in a small number of wavelet coefficients with the rest of the coefficients being zero or close to zero. By setting all the small valued coefficients to zero a very sparse representation can be obtained and exploited for compression purposes. For an orthonormal wavelet transform this thresholding of the coefficients corresponds to a global optimal approximation in terms of the L^2-norm and the introduced error expressed as a mean square error (MSE) is given by:

$$\begin{aligned} MSE &= \frac{1}{N}\sum_k (x_k - \hat{x}_k)^2 = \frac{1}{N}(x-\hat{x})^T \cdot (x-\hat{x}) \qquad (44) \\ &= \frac{1}{N}(W^T W(x-\hat{x}))^T \cdot (W^T W(x-\hat{x})) \\ &= \frac{1}{N}((y-\hat{y})^T \cdot WW^T \cdot ((y-\hat{y}) \\ &= \frac{1}{N}\sum_k (y_k - \hat{y}_k)^2, \end{aligned}$$

where W is an orthonormal transform and the x_k and $\hat{x}_k$ are the original and reconstructed signal and the y_k and $\hat{y}_k$ are the transform coefficients before and after thresholding. As a consequence one can explicitly give the exact error that occurs due to thresholding. For the case of biorthogonal bases property (44) fails. Note that if we want to divide by 4 in (36)-(39) and by 1 in (41)-(43) the property also fails. Instead we can use equation (36)-(39) as is, and then after the thresholding do a rescaling of the coefficients.

3 Survey of Previous Work

In [15, 16] Muraki introduced the idea of using wavelets to efficiently approximate 3D data. The 2D wavelet transform was extended to 3D and it was shown how to compute the wavelet coefficients. By setting small coefficients to zero the author showed that the shape of volumetric objects could be described in a relatively small number of 3D functions. The motivation for the work was to obtain shape descriptions to be used in for example object recognition and no results on storage savings were reported. The potential of wavelets to reduce storage is evident, though.

Motivated by the need for faster visualisation, a method for both compressing and visualising 3D data based on vector quantisation was given by Ning and Hesselink [17]. The volume is divided into blocks of small size and the voxels in each block are collected into vectors. The vectors are then quantised into a codebook. Rendering by parallel projection is accelerated by preshading the vectors in the codebook and reusing precomputed block projections. Since accessing of a single voxel is reduced to a simple table lookup in the codebook fast random access is supported. Compressing two volumes of size 128^3, a compression factor of 5 was obtained with blocking and contouring artifacts being reported.

Burt and Adelson proposed the Laplacian Pyramid [2] as a compact hierarchical image code. This technique was extended to 3D by Ghavamnia and Yang [6] and applied to volumetric data. Voxel values can be accessed randomly on the fly by traversing the pyramid structure. Since there is high computational overhead connected with the reconstruction, the authors suggest a cache structure to speed up reconstructions during ray casting. They achieve a compression factor of about 10 with the rendered images from the compressed volume being virtually indistinguishable from the images rendered from the original data.

Several compression methods, both lossless and lossy, for compressing and transmitting Visible Human images were presented and compared by Thoma and Long [25]. Among the lossy schemes, which as expected outperformed the lossless ones, the scheme based on wavelets did best. The wavelet method that Thoma and Long suggest compresses the images individually and consists of three steps comprised of a wavelet transform followed by vector quantisation with Huffman or runlength coding of the quantisation indices. This makes it a traditional 2D subband coder and compression factors of 20, 40 and 60 were reported with almost no visible artifacts for a factor of 20. The coder does not allow for fast random access to individual voxel as it was mainly designed for storage and transmission purposes. Also there is no exploitation of interpixel correlation/redundancies between adjacent slices.

The above methods all have in common that they support either high compression ratios or fast random access, but not both. This situation changed with the method proposed by Ihm and Park [12, 11]. This state of the art algorithm is able to compress CT slices of the Visible Human and compression ratios of up to 28 were presented, still with good fidelity of the reconstructed slices. The algorithm utilises a 3D wavelet transform setting insignificant wavelet coefficients to zero. This results in a very sparse representation which is coded by dividing the transformed volume into unit blocks and coding the blocks separately using a data structure that takes the spatial coherency of the wavelet coefficients into consideration.

As one of the three spatial dimensions can be considered as similar to time, 3D compression can borrow good ideas from research in video compression.

Chen and Said suggested to use a three-dimensional subband transform to code video sequences [3]. After the transform and quantisation has been applied the surviving wavelet coefficients are coded using the zero-tree technique developed in [22] and further improved and refined in [19, 20]. However, in [24] it is shown that better performance can be realised if motion compensation is performed before the 3D transform. These observations have led us to propose a new coding scheme for wavelet based compression of very large volume data with fast random access. In the following section we present this method.

4 New Coder

4.1 General Overview

First we present a black-box overview of both the encoder and the decoder, depicted in Figure 3, for our new compression scheme of volumetric data. Detailed explanation of each step will be given afterwards. A new observation making our coding strategy significantly different from the earlier methods is that even though the data is genuinely volumetric in nature, we are not confined to treating it as such. Instead we can treat it as a sequence of two dimensional slices in position or time and draw on results developed in the area of video coding. A major issue in video coding is the removal or exploitation of temporal redundancy or correlation. As mentioned in the previous section it was reported in [24] that combining a 3D wavelet transform with motion compensation techniques exploit temporal correlation to a greater extent than a 3D transform alone.

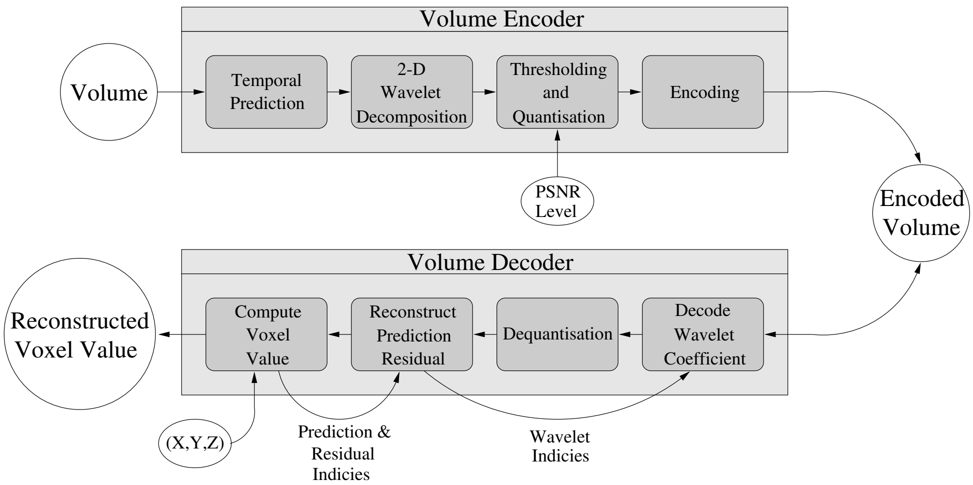

Figure 3: Overview of the encoder and the decoder.

4.1.1 Volume Encoder

Figure 3 depicts the four basic stages of our encoder together with the corresponding stages of the decoder. When designing our new coding scheme we constantly have to consider the trade-off between compression rate, distortion and decoding speed.

When coding the volume we assume that we have divided it into two-dimensional slices in the z-direction. The first step of the encoder will then be the removal of correlation along this z-direction. We call this stage *temporal prediction*. Ideally the next step should then according to [24], perform a three dimensional wavelet decomposition to further remove correlations in both the spatial and temporal directions. Since a 3D transform is computationally more expensive than a 2D transform and under the assumption that the prediction stage has removed enough of the temporal correlation, we adopt a 2D wavelet transform to handle the spatial redundancy as the next step. The third step, which is typical for a lossy subband coder, removes insignificant coefficients to make the representation even sparser. This step also quantises the remaining coefficients restricting these to a small number of possibilities. Finally in the last stage the wavelet transformed data is encoded efficiently in a way that allows fast retrieval of the data needed to reconstruct individual voxels.

4.1.2 Volume Decoder

In general the decoder consists of the inverse stages of the encoder but in reverse order. However there is a small but significant difference. Since the decoder acts on input from the user or another program some of the stages have to communicate in order to retrieve the desired voxels from the encoded data. This it not necessary in the encoder since it encodes the whole volume at once.

4.2 Test Data

The data that we have used for our experiments are the same as in [12] and was kindly made available by Professor Insung Ihm. The dataset is a volume of size $512 \times 512 \times 512$ rebuilt from the fresh[4] CT slices of the Visible Human. Rebuilding the volume was necessary since the fresh CT slices have varying pixel size and spacing. Each voxel is represented as a 12 bit grayscale value in the interval [0,4095] and is stored in 16 bits resulting in a total volume size of 256 Mbytes.

4.3 Temporal Prediction

Many different methods for motion estimation and motion compensation have been reported and investigated in the literature. Among the most popular are block-matching algorithms [5, 23], parametric motion models [9, 18, 24], optical flow [23] and pel-recursive methods [23].

The scheme that we have chosen to adopt, depicted in Figure 4, is the simplest possible, yet very effective. It corresponds to a best matching neighbour scheme. Every F-th slice (called *R-slices*) is used as a reference frame and is coded without temporal prediction. The prediction of a

[4] There exist both fresh and frozen CT and MRA images

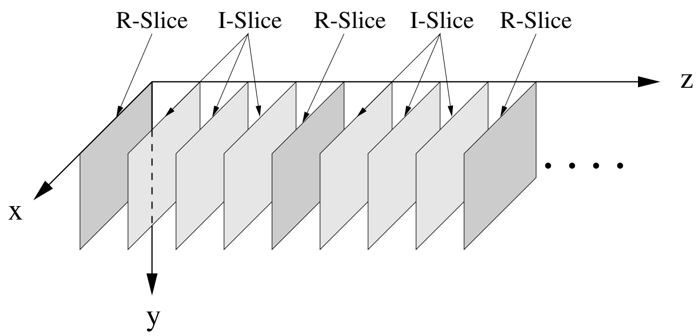

Figure 4: Ordering of R-Slices and I-slices.

frame in between two R-slices (called *I-slices*) is then chosen to be the neighbouring R-frame giving the best match in terms of the Mean Square Error. Finally the residual is constructed and sent to the next stage of the coder for further processing. The voxel values of the original data resides in the integer interval [0,4095] so the possible values of the residual lies in the interval [-4095,4095]. By using both backward and forward R-slices in the prediction, we effectively half the longest distance between an I-frame and an R-frame. Experiments have shown this gives significantly better compression results than just using forward prediction. The distance F between two R-slices is chosen to be $F = 2^k$ for faster processing. The issue of selecting a good value for k will be discussed in Section 4.7.

4.4 Wavelet Decomposition, Thresholding and Quantisation

An important objective for our compression scheme is to code the volume with high accuracy in a minimum number of bits. Wavelets have proven to be very effective at achieving this goal. As mentioned in Section 2.2.2 they have the ability to efficiently concentrate the energy of a signal in few wavelet coefficients with large magnitude creating a compact or sparse representation and thus making it easier to code in a smaller amount of bits.

When implementing the wavelet transform we have to decide which wavelet to use and how many levels of decomposition to perform. Different wavelets correspond to filters of different length as described in Section 2.3.2. Table 2 shows how the choice of wavelet and the number of decomposition levels affects the number of wavelet coefficients that are used in the reconstruction.

		Filter length			
		2	4	6	8
Level	1	4	16	36	64
	2	7	31	71	127
	3	10	46	106	190
	4	13	61	121	253

Table 2: The influence of filter length and decomposition level on the number of coefficients needed for reconstruction.

Even though the Haar wavelet does not perform so well in terms of quality as other wavelets it certainly allows for faster reconstruction, only being a two-tap[5] filter, and thus it becomes our choice of wavelet. For the number of decomposition levels we restrict ourselves to two levels, despite the fact that in wavelet based compression it is common to perform three or four levels. With a slice size of 512×512 and two levels of decomposition more than 93 percent of all the coefficients are already decomposed into wavelet coefficients and according to Table 2 we only need 7 coefficients to reconstruct a voxel in an R-slice (14 in an I-slice).

After applying the wavelet transform to each slice all wavelet coefficients below a certain threshold are set to zero in order to make the representation even sparser. The threshold is determined such that the PSNR[6] does not drop below a user determined value (see Figure 3). According to Section 2.5 the coefficients from all the slices should be sorted and then in ascending order set to zero until the desired PSNR is obtained. This would be similar to choosing a threshold τ that is ϵ greater than the last coefficient that was set to zero. Unfortunately sorting the coefficients for large volumes quickly becomes impractical. Instead we use different thresholds for each slice with the thresholds being determined by sorting as just described. This of course is not globally optimal for the volume but it provides a reasonable solution to the problem. When calculating the PSNR of each slice we use the global maximum of the whole volume.

After thresholding we rescale as described in Section 2.5 and round the

[5]Tap is the term used to denote the filter length of a finite impulse response (FIR) filter.

[6]PSNR $= 10\log_{10}(\frac{\max(x_i^2)}{MSE})$, x_i being the original image

remaining coefficients to the nearest integer. Rounding the coefficients is negligible with respect to MSE since few coefficients are remaining. The rescaling insures that the coefficients now lie in the interval $[-4095, 4095]$, i.e. use Equations (36)-(39) with division by 4 to see this. Rather than use 13 bits to represent the values in this interval we quantise the nonzero coefficients to the interval [0,255] so they fit into one byte. As it turns out in the next section not all coefficients need to be quantised and those that do come in blocks G of at most 64 coefficients. We use a uniform scalar quantiser of the form

$$\tilde{C} = \left\lfloor \frac{A-B}{255} \times x + 0.5 \right\rfloor + B, \tag{45}$$

where A and B are 16 bit integer parameters to the quantiser, the byte x is the quantisation value and $\tilde{C}$ is the reconstructed value. For all blocks G: A, B and x can be determined as:

$$A = \max_{c \in G}(c), B = \min_{c \in G}(c), x = \left\lfloor 255 \times \frac{C-B}{A-B} + 0.5 \right\rfloor. \tag{46}$$

4.5 Encoding Wavelet Coefficients - Data Structure

The last stage of the compression process is the encoding of the remaining wavelet coefficients. In addition we also need to encode the positional information about the coefficients - this is often referred to as the significance map. Shapiro [22] showed that the bits needed to code the significance map is likely to consume a large part of the bit budget with the situation becoming worse at low bitrates. In his article Shapiro proposed a technique called zerotree coding for storing the significance map. Although effective this method does not lend itself to fast retrieval of individual wavelet coefficients. The same problem is true for run-length coding, Huffman coding or arithmetic coding since they produce variable length codes [21].

We now propose a method for storing the significant coefficients and their significance map. The method is illustrated in Figure 5 and is designed to fit preprocessed CT images of the Visible Human dataset but is easily extended to other volumes.

Given that most of the wavelet coefficients have been set to zero and the usually spatial coherence of a wavelet decomposed image (e.g. see Figure 7 and 8 Appendix A) it is likely that the zero coefficients appear in dense clusters. So the initial idea will be to split each slice into

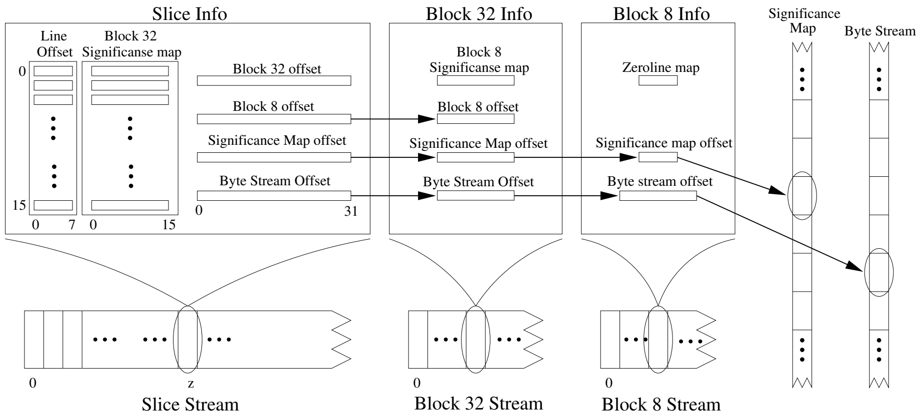

Figure 5: Datastructure used for encoding significant wavelet coefficie nts.

quadratic blocks of size (length) B and then use a bitmap to differentiate between the blocks containing at least one nonzero coefficient (*nonzero B block*) and the blocks having all coefficients equal to zero (*zero B block*). Spending one bit on each block in the bitmap is optimal with respect to first order entropy if the block size B is chosen such that the ratio between zero blocks and nonzero blocks is one half. Table 3 shows the percentage of nonzero blocks for different sized blocks and PSNR levels, with a fixed R-slice spacing of F=4. From the table we notice that $B = 32$ is the best choice.[7] The bitmaps (*Block 32 Significance Map*) for

		PSNR Level				
		43	46	49	52	53
Block Size	4	5.5%	7.9%	10.6%	15.5%	22.9%
	8	10.8%	14.7%	18.4%	24.5%	32.6%
	16	19.8%	24.8%	29.3%	35.8%	**43.7%**
	32	32.8%	**38.4%**	**43.3%**	**50.0%**	58.2%
	64	**55.8%**	61.7%	66.0%	70.6%	78.0%

Table 3: Number of nonzero blocks in percent for different block sizes and PSNR levels

each slice are collected into records called *Slice Info*, together with other information which we explain shortly, and kept in an array. Retrieval of a Slice Info record is then simple array lookup since each record has a

[7]B is a power of two for efficient processing.

fixed size. For each of the nonzero 32 blocks, further information needs to be stored and we allocate a new record (*Block 32 Info*) for each of them. These records are also stored in a stream. Each slice contains a variable number of nonzero 32 blocks, so in order to quickly find a given Block 32 Info record in the stream we add the following offsets to the Slice Info Record. *Block 32 offset* holds the index in the Block 32 stream of the first Block 32 Info record for that particular slice. The *Line Offset* array contains counts of all nonzero 32 blocks that precede a given line in the Block 32 Significance Map. In order to find the position of some Block 32 record in the stream, we first compute where it is in the bitmap. Then we count the number of bits that precedes it in the respective bitmap-line. This count is then added to the Block 32 Offset together with the correct Line Offset and the result is used for lookup. Counting the number of bits in a bitmap line can be done efficiently by using a precomputed table with $2^{16} = 65536$ entries. Following the same

		PSNR Level				
		43	46	49	52	53
Block Size	4	16.7%	20.6%	24.5%	31.0%	39.3%
	8	33.1%	**38.1%**	**42.4%**	**48.9%**	**56.0%**
	16	**60.2%**	64.5%	67.6%	71.5%	75.1%

Table 4: Empirical probability of a block being nonzero given that it is contained in a nonzero block of size 32.

idea, the nonzero 32 blocks can be split into sub-blocks. Table 4 shows the empirical conditional probabilities of a block with size 4, 8 or 16 being nonzero given that it is contained in a nonzero block of size 32. It follows that a sub-block of size 8 is a good choice. The information about these sub-blocks is stored in records (*Block 8 Info*) and kept in a new stream (*Block 8 Stream*). Similar to the problem of addressing the 32 blocks we need offsets to quickly access a Block 8 Info record. To this end we add a *Block 8 Significance Map* and a *Block 8 offset* to the Block 32 Info record. There are at most $\frac{512^3}{8^2}$ nonzero sub-blocks of size 8 in the volume. To offset all of these we need at least 21 bits. We observe that there are at most $\frac{512^2}{8^2}$ nonzero blocks of size 8 in each slice so instead of using 32 bits in the Block 32 Info record[8] we divide the offset in two using 32 bits in the Slice Info record and only 16 bits in the Block 32 record. That way the total overhead of storing all the offsets is reduced. Finally

[8]Offsets are 8, 16 or 32 bits to ease programming.

each nonzero 8 Block is divided into lines keeping a bitmap *Zeroline Map* in the Block 8 Info record marking the lines which contain all zeros. For all other lines containing at least one nonzero coefficient we keep one byte as a *Significance Map*. This way we need between 0 and 8 bytes for the map, hence we store them in a stream (*Significance Map Stream*) introducing a new offset, the *Significance Map Offset*, which is similarly divided between the three Info records for efficient storage. Table 5 shows that dividing the sub-blocks into lines is a good idea since about half of the lines in an 8 sub-block are zero. The Significance Maps give the positions of the significant coefficients.

	PSNR Level				
	43	46	49	52	53
Zerolines	60.7%	57.2%	53.6%	47.6%	40.1%

Table 5: Empirical probability of a line in a nonzero sub-block of size 8 being all zero

In the following it will be explained how the coefficients are stored. As stated in Section 4.4 the wavelet coefficients have been scaled and rounded to the integer interval [-4095,4095] and we store them in a stream (*Byte Stream*) pointed to by offsets *Significance Offset* located in all three Info records. Inspired from [12, 11] we observe that the coefficients within a size 8 sub-block are likely to be numerically close so whenever the coefficients in a sub-block all belong to either the interval $[\theta, \theta + 127]$ or the interval $[-\theta - 128, -\theta]$, where θ is a two-byte offset, we code them by storing θ in the Byte Stream followed by a signed displacement (1 byte) for each coefficient. For all other sub-blocks not satisfying this property we quantise as explained in Section 4.4, storing A and B in two bytes and x in one byte.

How a coefficient is stored can be coded in 1 bit and placed in the most significant bit (MSB) of the Byte Stream Offset in the Block 8 Info record. This bit is free since there are only $32^2 = 1024$ coefficients in each 32 block and we use 16 bits for the offset. Similarly for all I-slices we use the MSB of the Block 32 Offset in the Slice Info record to indicate the direction of the prediction.

Looking at the sizes of the offsets used throughout the data structure, it is quite easy to verify that they are large enough, except for the Byte Stream Offset in the Block 32 Record. We need 18 bits to offset all coefficients in a slice but we have only allocated 16 bits for the task. We put the last 2 MSB in the MSB of the Block 8 Offset.

4.6 Analysis of Performance

In this section we analyse the work needed for decoding a single voxel. The reconstruction filter for the Haar wavelet is a two-tap filter. Performing a two level reconstruction we therefore need 7 wavelet coefficients (four for the first level and three for the second). If we are decoding a voxel in an I-slice we must also decode the same number of coefficients for the R-slice resulting in the retrieval of 14 wavelet coefficients. Table 6 shows average values for different R-slice spacings. It is observed that the amount of information that needs to be retrieved and the number of additions that must be performed is not critical with respect to the slice spacing.

			Weighted Average for a spacing			
	R-Slice	I-Slice	4	8	16	32
Wavelet Coefficients	7	14	12.25	13.13	13.56	13.78
Additions	6	12	10.50	11.25	11.63	11.81

Table 6: Number of coefficients and additions needed for reconstruction of a voxel.

For accessing a wavelet coefficient the approximate workload can be assessed. First we look at the reconstruction algorithm:

function Wavelet_coefficient(x, y, z)
  S := Lookup_Slice_Info(z);
(1) **if** is_in_zero_block(x, y, **S**) **then return** 0;

  B32 := Lookup_Block32_Info(**S**, Block 32 offset, offset, bitma p);
  Calculate relative index (x′, y′) in **B32**;
(2) **if** is_in_zero_block(x′, y′, **B32**) **then return** 0;

  B8 := Lookup_Block8_Info(**B32**, **S**, Block 8 offset, bitma p);
  Calculate relative index (x″, y″) in **B8**;
(3) **if** is_zeroline(y″, **B8**) **then return** 0;

  M := significance_map(M, offsets);
(4) **if** bit(x″, M) = 0 **then return** 0;

  Access bytestream using offsets;
(5) **return** value;

end;

The division of each slice into blocks and lines was designed so the empirical probability of going from (1)→(2), (2)→(3), or (3)→(4) in the reconstruction algorithm is roughly one half. Combining these probabilities with the fact that less than 6 percent of the wavelet coefficients are nonzero (see next section) we obtain the numbers in Table 7. From this

Stage	1	2	3	4	5
Probability	100%	50%	25%	12.5%	6%

Table 7: Empirical probability of reaching a certain stage in the algorit hm.

table we estimate that on average $1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{16} \approx 2$ lookups in the encoded data have to be performed. In addition, one lookup in the bitcount table is performed on average.

According to the table, roughly 13 wavelet coefficients must be extracted from the compressed data in order to reconstruct a single voxel. Observing from Equations (41)-(43) that 4 values can be reconstructed from 4 wavelet coefficients this seems inefficient. Figure 6 shows how the wavelet coefficients on different levels are related to the reconstructed coefficients. If the voxels are accessed in some regular pattern increased efficiency can be achieved by reconstructing all voxels in the 4×4 neighbourhood reusing the extracted coefficients. Furthermore the grouping of the first level detail coefficients ensures that the information about each group resides in the same Block 8 Info record resulting in fewer lookups in the data structure.

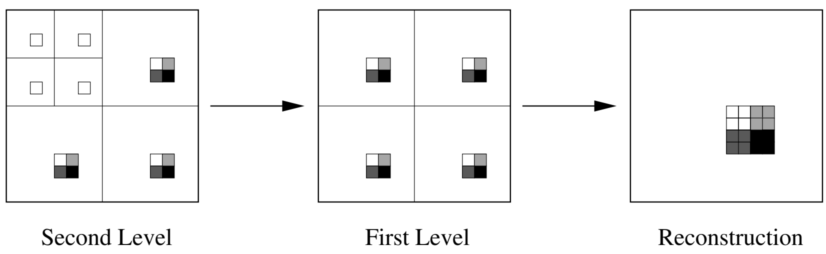

Figure 6: Relationship between average, detail and reconstructed coeffi cients.

4.7 Results

Before presenting the main results we first deal with the issue of selecting a k such that the distance $F = 2^k$ between two R-slices is best possible. Table 8 illustrates the size of the compressed volume for different choices of k with a desired PSNR level fixed at 46. According to the table selecting $k = 2$, resulting in R-slice spacings of 4, produces the best result. This is quite surprising because it means that 1/4 of the slices are coded without temporal prediction suggesting that a better prediction scheme might prove fruitful. For the rest of this section we keep $F = 4$ fixed.

Spacing $F = 2^k$	Original	2	4	8	16	32
Compressed size (Mb)	256	7.11	**6.01**	6.28	7.14	8.24

Table 8: Compressed size for different R-slice spacings. The desired PSNR level is 46.

The main results of our new coding scheme is presented in Table 9. We tested with desired PSNR levels of 43.0, 46.0, 48.03, 51.6, and 55.8 achieving compression factors between 60.2 and 14.5. On average this is about a 50% reduction in size compared with [12, 11] which were the previously best known results. A complete comparison between our new scheme and [12] is presented in Table 10. One of the main reasons for achieving an increased compression rate is that our method succeeds in setting more wavelet coefficients to zero for a given PSNR level, see Table 10.

We note, in Table 9, that the actual PSNR is not equal to the one desired, but close. This is because thresholding stops before the desired PSNR level is exceeded. Also rounding and quantisation of the coefficients introduce additional errors.

		Desired lower bound on PNSR (dB)				
		43.0	46.0	48.3	51.6	55.8
Compression	Factor	60.2	42.6	32.4	22.2	14.5
	Size (Mb)	4.26	6.01	7.9	11.56	17.68
Errors	Actual PSNR	43.00	46.00	48.29	51.55	55.62
	SNR	24.98	27.98	30.27	33.53	37.60
	MSE	839.59	421.39	248.83	117.34	45.92

Table 9: Results on compression ratio and quality

		Desired lower bound on PNSR (dB)				
		43.0	46.0	48.3	51.6	55.8
Non-zero coefficients	Old method	2.78%	4.57%	6.32%	8.94%	13.30%
	New method	1.44%	2.13%	2.94%	4.62%	7.73%
Compressed size (Mb)	Old method	9.08	13.25	17.01	22.17	29.99
	New method	4.26	6.01	7.9	11.56	17.68
	Improvement	53.1%	53.6%	53.6%	47.9%	41.0%

Table 10: Comparison between previously best known and new method.

Figure 9 shows sample slices from the original volume and from different compressed volumes[9]. It is observed that for a PSNR level of 43 some blockiness and loss of small level details occur. In all cases edges are preserved extremely well. We have also generated ray-cast rendered images from the original and compressed volumes. Figure 10 and 11 depicts rendered images of skin and bone. To generate the images we used Volvis 2.1 [1][10]. The images are essentially indistinguishable from the original except for the volumes with PSNR level of 43 and 46. For these two cases the introduced artefacts are pleasing and most of the details are well preserved.

We evaluate the reconstruction overhead by loading, in turn, the uncompressed and compressed data into memory and measure the time it takes to access 1,000,000 randomly selected voxels in the volume. We tested with different CPU speeds and cache sizes yielding the results in Table 11. The machines used were all from SGI. It is observed that our algorithm on average is about 3 to 5 times slower for the compressed data. This corresponds only to a slight increase in the time needed to reconstruct an arbitrary voxel compared with [12, 11]. Considering the time it would take to access the voxels from a harddisk or over a network this is a very small slowdown. Applications working with volumetric data hardly make accesses purely at random. Instead accesses are done in some regular way. As explained in Section 4.6 this might lead to more efficient decoding. We have performed an experiment where voxels are reconstructed in $4 \times 4 \times 4$ blocks in the following way. A $4 \times 4 \times 4$ block is considered to contain voxels from 3 I-slices followed by an R-slice. We start by decoding the 4×4 voxels in the R-slice. Assuming that we have

[9] We refer to http://www.brics.dk/~ffr for better viewing of all images

[10] The rendering system that we used only supports byte data, so the volumes were uniformly quantised from 12 bits to 8 bits before rendering

	CPU Speed	2nd level Cache	Original Data	Desired lower bound on PNSR (dB)				
				43.0	46.0	48.3	51.6	55.8
Random	180	1 Mbytes	3.05	9.82	10.85	11.66	13.10	15.01
	180	2 Mbytes	2.87	8.77	9.75	10.72	11.82	13.67
	250	4 Mbytes	1.99	6.01	6.59	7.06	8.11	9.35
Block wise	180	1 Mbytes	20.37	48.63	50.22	51.90	54.75	59.13
	180	2 Mbytes	13.67	47.05	48.72	50.13	52.97	57.29
	250	4 Mbytes	10.07	34.60	36.16	36.75	38.81	41.92

Table 11: Results on voxel reconstruction times in seconds.

kept the R-slice voxel values from the block above the one we are reconstructing we compute the 4×4 voxels of each I-slice reusing the already computed R-slice values. Results on reconstruction time for the whole volume is shown in Table 11. Roughly an extra 30-40 seconds are spent on the compressed volumes than on the original. This is an increase of approximately 20-25 seconds compared to [12, 11]. Considering that the images in Appendix C and D each took about 5 minutes to render we find this acceptable.

5 Concluding Remarks

We have presented a wavelet based 3D compression scheme for very large volume data supporting fast random access to individual voxels within the volume. Experiments on the CT data of the Visible Human have proven that our method is capable of providing high compression rates with fairly fast decoding of random voxels. Similar to [11] our intension is to provide a method that allows a wider range of users the ability of working with and visualising very large volume data.

Some aspects of our coder need further research and attention. For example we only employ a very simple prediction scheme. By using a more advanced prediction method we might be able to reduce the number of slices that are coded without temporal prediction and thereby increase the compression ratio. Also how other wavelet types will affect compression ratio and quality must be evaluated. Since our method only performs a 2D wavelet transform it scales better with respect to the number of coefficients that is needed for reconstruction when the wavelet filters become longer than methods using a 3D transform. This is attractive since the number of coefficients that must be extracted directly affect the decoding speed. To make our compression method useful in an interactive visualisation environment decoding speed must be improved, either by developing an efficient cache structure temporary holding voxel values or by adding redundancy to the data structure to decrease lookup overhead. In the future we intend to investigate these issues. We will also undertake an investigation as how to expand our method to colour volumes.

6 Acknowledgements

I wish to thank my supervisor Professor Brian H. Mayoh for giving encouragement and advice. Also I would like to thank Professor Insung Ihm for providing the preprocessed Visible Human data. Likewise thanks to the team developing VolVis for supplying it freely.

References

[1] VolVis 2.1. Department of computer science, the state university of new york at stony brook. http://www.cs.sunysb.edu/~vislab/volvis_home.html.

[2] Peter J. Burt and Edward H. Adelson. The laplacian pyramid as a compact image code. *IEEE Trans. on Communications*, 31(4):532–540, April 1983.

[3] Yingwei Chen and William A. Pearlman. Three-dimensional subband coding of video using the zero-tree method. *Proc. of SPIE – Visual Communications and Image Processing '96*, pages 1302–1310, March 1996.

[4] Ingrid Daubechies. *Ten Lectures on Wavelets*. SIAM, Philadelphia, Pennsylvania, 1992.

[5] Didier J. Le Gall. The MPEG video compression algorithm. *Signal Proc.: Image Communications*, 4:129–140, October 1992.

[6] Mohammad H. Ghavamnia and Xue D. Yang. Direct rendering of laplacian pyramid compressed volume data. *Proceedings of Visualization '95*, pages 192–199, October 1995.

[7] A. Haar. Zur Theorie der orthogonalen Funktionensysteme. *Math. Annal.*, 69:331–371, 1910.

[8] W. Heisenberg. Über den anschaulichen Inhalt der quantentheoretischen Kinematik und Mechanik. *Zeitschrift für Physik*, 43:172–198, 1927.

[9] Michael Hoetter. Differential estimation of the global motion parameters zoom and pan. *Signal Processing*, 16:249–265, 1989.

[10] NLM homepage. http://www.nlm.nih.gov/research/visible/-visible_human.html.

[11] Insung Ihm and Sanghun Park. Wavelet-based 3D compression scheme for very large volume data. In *Graphics Interface*, pages 107–116, Vancouver, Canada, June 1998.

[12] Insung Ihm and Sanghun Park. Wavelet-based 3D compression scheme for interactive visualization of very large volume data. *Computer Graphics Forum*, 18(1):249–265, March 1999.

[13] Stéphane Mallat. A theory for multiresolution signal decomposition: The wavelet representation. *IEEE Trans on Patt. Anal. and Mach. Intell.*, 11(7):674–693, July 1989.

[14] Stéphane Mallat. *A Wavelet Tour of Signal Processing.* Academic Press, 1998.

[15] Shigeru Muraki. Approximation and rendering of volume data using wavelet transforms. *Proceedings of Visualization '92*, pages 21–28, October 1992.

[16] Shigeru Muraki. Volume data and wavelet transforms. *IEEE Computer Graphics & Applications*, 13(4):50–56, July 1993.

[17] Paul Ning and Lambertus Hesselink. Fast volume rendering of compressed data. *Proceedings of Visualization '93*, pages 11–18, October 1993.

[18] C. A. Papadopoulos and Trevor G. Clarkson. Motion compensation using second-order geometric transformations. *IEEE Trans. on Circuits and Systems for Video Technology*, Vol. 5:319–331, August 1995.

[19] Amir Said and William A. Pearlman. Image compression using the spatial-oriantation tree. *IEEE Proc. of Intl. Symp. on Circuits and Systems*, pages 279–282, May 1993.

[20] Amir Said and William A. Pearlman. A new, fast, and efficient image codec based on set partioning in hierarchical tress. *IEEE Trans on Circuits and Systems for Video Technology*, 6(3):243–250, June 1996.

[21] Khalid Sayood. *Introduction to Data Compression*. Morgan Kaufmann Publishers, Inc., USA, 1996.

[22] Jerome M. Shapiro. Embedded image coding using zerotress of wavelet coefficients. *IEEE Trans. on Signal Processing*, 41(12):3445–3462, December 1993.

[23] A. M. Tekalp. Digital video processing. *Prentice Hall Signal Processing Series*, 1995.

[24] Jo Yew Tham, Surendra Ranganath, and Ashraf A. Kassim. Scalable very low bit-rate video compression using motion compensated 3-D wavelet decomposition. *IEEE ISPACS Workshop*, 3:38.7.1–38.7.5, November 1996.

[25] Geoge R. Thoma and L. Rodney Long. Compressing and transmitting visible human images. *IEEE Multimedia*, 4(2):36–45, 1997.

A Significance Map of Wavelet Decomposed Slices

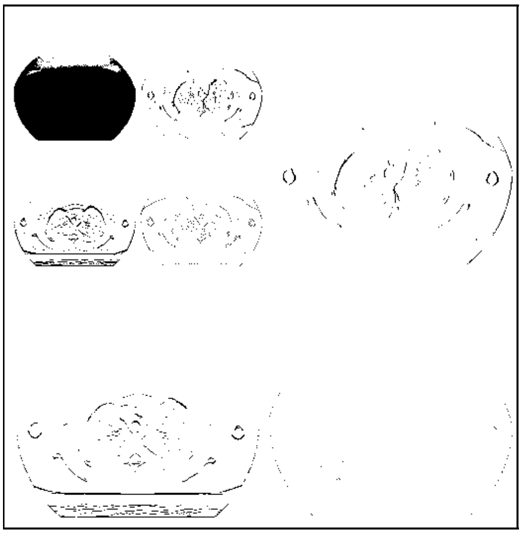

Figure 7: Significance map of thresholded wavelet decomposition of slice 344 with PSNR level 46.

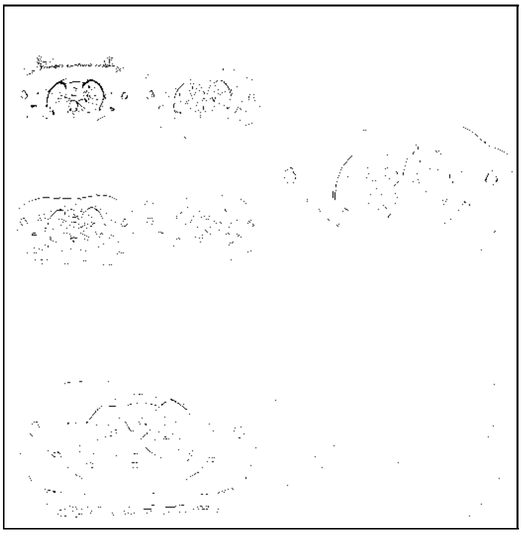

Figure 8: Significance map of thresholded wavelet decomposition of slice 343 residual with PSNR level 46.

B Decompressed Sample Slice

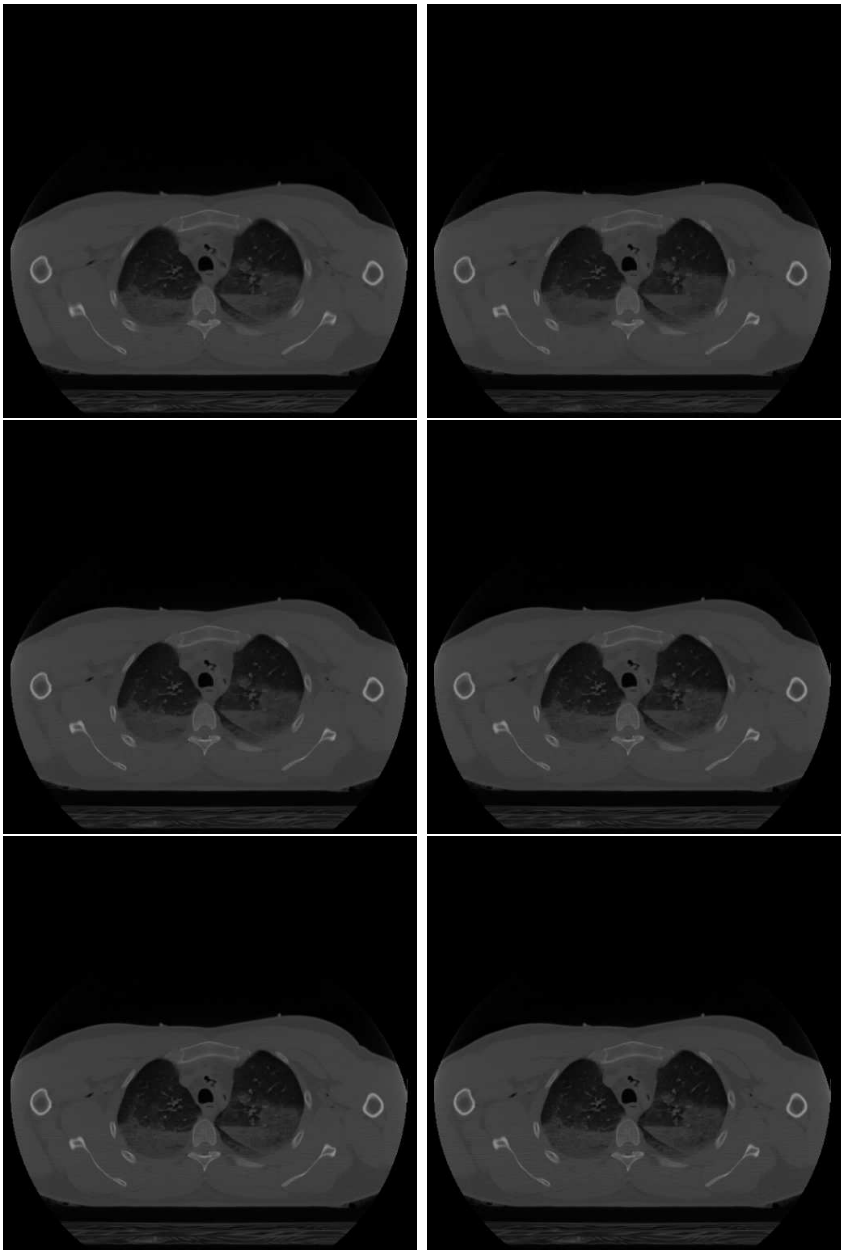

Figure 9: Sample slice (no. 345). From left to right and top to bottom *Original volume*, *Compressed volumes* with PSNR levels of *55.8, 51.6, 48.3, 46.0 43.0.*

C Rendered Images - Skin

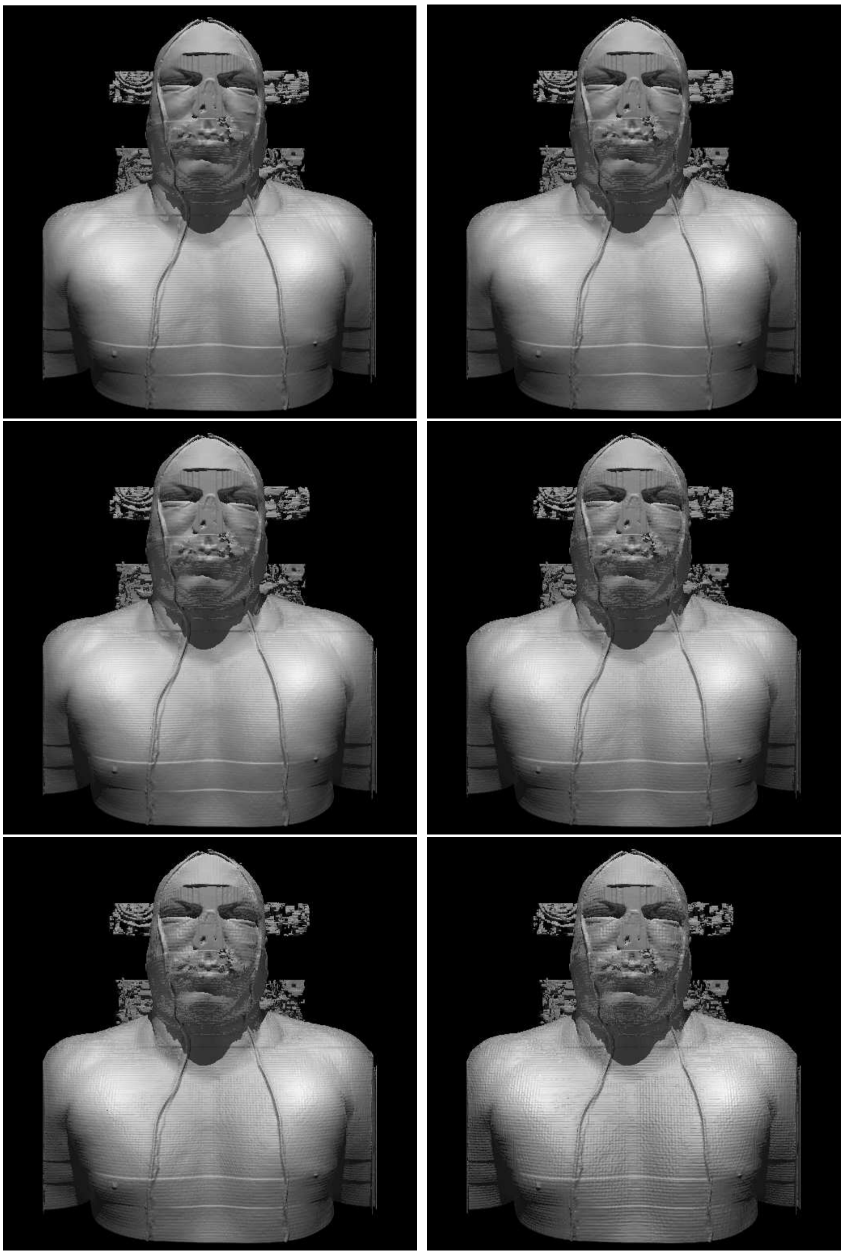

Figure 10: Rendered images. From left to right and top to bottom *Original volume*, *Compressed volumes* with PSNR levels of *55.8, 51.6, 48.3, 46.0 43.0.*

D Rendered Images - Bone

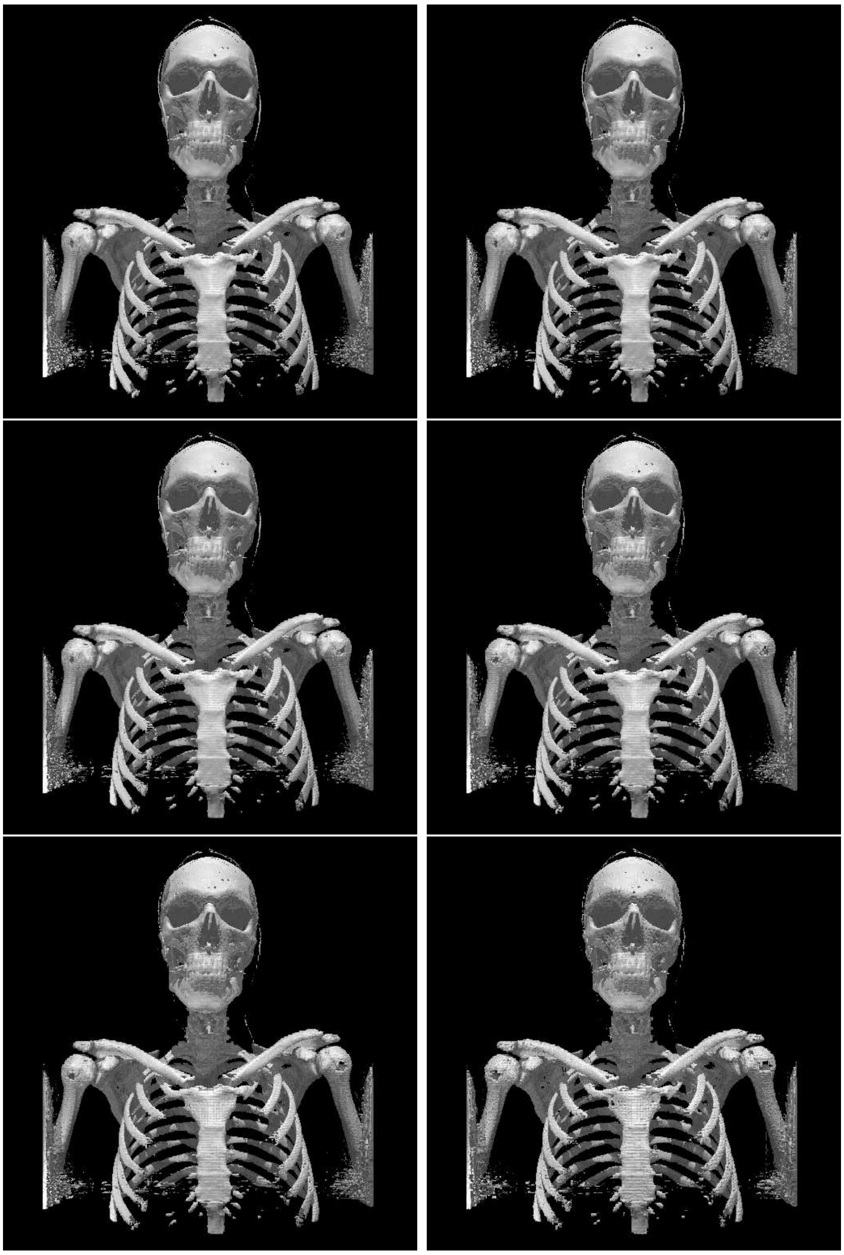

Figure 11: Rendered images of bone. From left to right and top to bottom *Original volume*, *Compressed volumes* with PSNR levels of *55.8, 51.6, 48.3, 46.0 43.0.*

Recent BRICS Report Series Publications

RS-99-34 **Flemming Friche Rodler.** ***Wavelet Based 3D Compression for Very Large Volume Data Supporting Fast Random Access*****. October 1999. 36 pp.**

RS-99-33 **Luca Aceto, Zoltán Ésik, and Anna Ingólfsdóttir.** ***The Max-Plus Algebra of the Natural Numbers has no Finite Equational Basis*****. October 1999. 25 pp. To appear in** ***Theoretical Computer Science*****.**

RS-99-32 **Luca Aceto and François Laroussinie.** ***Is your Model Checker on Time? — On the Complexity of Model Checking for Timed Modal Logics*****. October 1999. 11 pp. Appears in Kutyłowski, Pacholski and Wierzbicki, editors,** ***Mathematical Foundations of Computer Science: 24th International Symposium*****, MFCS '99 Proceedings, LNCS 1672, 1999, pages 125–136.**

RS-99-31 **Ulrich Kohlenbach.** ***Foundational and Mathematical Uses of Higher Types*****. September 1999. 34 pp.**

RS-99-30 **Luca Aceto, Willem Jan Fokkink, and Chris Verhoef.** ***Structural Operational Semantics*****. September 1999. 128 pp. To appear in Bergstra, Ponse and Smolka, editors,** ***Handbook of Process Algebra*****, 1999.**

RS-99-29 **Søren Riis.** ***A Complexity Gap for Tree-Resolution*****. September 1999. 33 pp.**

RS-99-28 **Thomas Troels Hildebrandt.** ***A Fully Abstract Presheaf Semantics of SCCS with Finite Delay*****. September 1999. 37 pp. To appear in** ***Category Theory and Computer Science: 8th International Conference*****, CTCS '99 Proceedings, ENTCS, 1999.**

RS-99-27 **Olivier Danvy and Ulrik P. Schultz.** ***Lambda-Dropping: Transforming Recursive Equations into Programs with Block Structure*****. September 1999. 57 pp. To appear in the November 2000 issue of** ***Theoretical Computer Science*****. This revised report supersedes the earlier BRICS report RS-98-54.**

RS-99-26 **Jesper G. Henriksen.** ***An Expressive Extension of TLC*****. September 1999. 20 pp. To appear in Thiagarajan and Yap, editors,** ***Fifth Asian Computing Science Conference*****, ASIAN '99 Proceedings, LNCS, 1999.**