Zuerst ersch. in: Computer Graphics Forum ; 20 (2001), 3. - S. 49-57

Rapid High Quality Compression of Volume Data for Visualization

Ky Giang Nguyen and Dietmar Saupe

Universität Leipzig, Institut für Informatik
Augustusplatz 10–11, 04109 Leipzig, Germany.

Abstract

Volume data sets resulting from, e.g., computerized tomography (CT) or magnetic resonance (MR) imaging modalities require enormous storage capacity even at moderate resolution levels. Such large files may require compression for processing in CPU memory which, however, comes at the cost of decoding times and some loss in reconstruction quality with respect to the original data. For many typical volume visualization applications (rendering of volume slices, subvolumes of interest, or isosurfaces) only a part of the volume data needs to be decoded. Thus, efficient compression techniques are needed that provide random access and rapid decompression of arbitrary parts the volume data. We propose a technique which is block based and operates in the wavelet transformed domain. We report performance results which compare favorably with previously published methods yielding large reconstruction quality gains from about 6 to 12 dB in PSNR for a 512^3*-volume extracted from the Visible Human data set. In terms of compression our algorithm compressed the data 6 times as much as the previous state-of-the-art block based coder for a given PSNR quality.*

1. Introduction

Volume visualization is one of the most actively researched topics in scientifi visualization and motivated by the growing need to understand and interpret large, multi-dimensional data especially in the medical application domain. The methods for volume data representation and visualization become more and more ubiquituous, and advances in computer hardware are now bringing 3D technology to nearly every computer system. However, for volume visualization techniques it is typically assumed that the whole data set is resident in memory. Since volume data can get very large, say some hundreds of megabytes, common current workstations and PCs may not be equipped with suffi cient amounts of memory and therefore are unable to visualize large data sets. This problem will not be overcome by increased memory space in future computer systems since as time progresses also the resolution of the volume data sets will increase causing the data set size to grow as well.

A popular example of large volume data that illustrates the associated problems comes from the Visible Human Project.[16] A few years ago, the National Library of Medicine (NLM) acquired CT-, MR- and color cryosection images of a representative male and female cadaver and built a very large digital library of corresponding volumetric data. This library found large interest and was used in a wide range of multimedia applications. One part of the Visible Man data set consists of 1871 axial scans of the entire body taken at 1 mm intervals and amounts to 15 Gbytes. The corresponding color cryosection images of the female data set were taken at 1/3 mm intervals and the data set is 40 Gbytes. Naturally, compression of these large data was an important issue.[1, 4, 6, 8, 14]

For the study of the human anatomy with these or similar data sets, the following scenario comes into question. The complete data sets can be held on a server in compressed format. Over the network, clients may access those data and request individual subvolumes of interest for visualization; say volumes comprizing the region of the heart, of the lungs and the bronchial tree, or of the brain. The requested subvolume should be extracted directly from the compressed data, transmitted still in compressed form over the network and then prepared for visualization at the client side. For example, a subvolume of resolution 512^3 of 16 bit intensity scalars may be requested. In uncompressed format this would amount to 256 megabytes requiring a very large transmission time and also the data set would not fi into CPU memory of most current workstations and PCs. At a compression ratio of 25:1 only about 10 megabytes need to be transmitted and stored. Then for the visualization of a slice of the volume or for an isosurface, only a small part of the 512^3 volume needs to be available in the spatial domain in uncompressed form. Therefore, an adaptive compression/decompression scheme is sought allowing region-of-interest decompression without requiring decompression of the entire data set. Overall, on the server-side volume extraction in the compressed domain is needed, while at the client side region-of-interest decoding is required. We remark that the server and the client may be processes running on the same physical machine. The compressed data may be stored on disk or external memory (CD, DVD) and the processor may select and load into memory those parts of the compressed data that is needed due to the user's query.

There are two aspects of the approach that must be taken into consideration. First, compression at the required bitrates is lossy; only an approximation of the original data can be recovered at the receiver. This may be acceptable in most cases (browsing, illustrating, education, and so on) but prohibitive in some cases such as medical diagnosis. Second, having the data in compressed format on the client requires decompression, a process that may take more time than that of plainly reading in spatial data in raw uncompressed format.

Both stated requirements for such a system — extraction in the spatial as well as in the compressed domain — can be met by block based compression techniques in which the volume is partitioned into blocks such as equal sized cubes each of which is compressed (and decompressed) independently. For any given volume query the required blocks can be identif ed and transmitted by the server or decompressed for visualization by the client. There is a tradeoff between block size and achievable compression performance. Small block sizes are preferable for extraction reducing the amount of data that is decompressed but not used for visualization. On the other hand, large blocks offer better reconstruction quality because there is more context information per voxel in large blocks which can be capitalized by the encoder achieving better reconstruction quality at the same compression ratio.

2. Previous work

Wavelets are powerful mathematical tools for approximating signals and representing functions hierarchically. Most of the recent best methods for image compression are based on wavelet transforms and achieve better rate-distortion performance than block based DCT methods, that are central to the old JPEG standard. In fact, the new JPEG2000 image compression standard relies on the wavelet transform.

Thoma and Long[14] compared experimental results where several 2D lossy compression methods were applied to the Visible Human data. The study focused on eff cient storage and transmission, rather than on run-time manipulation. A method based on the wavelet transform applied to each 2D slice of the volume data gave the best results.

While this study applied 2D compression techniques to 3D volume data there are also true 3D volume data compression techniques. Muraki[7] introduced the idea of using a 3D wavelet transformation for approximation and compression of 3D scalar data. Luo et al[5] used a modif ed embedded zerotree wavelet (EZW) algorithm[10] for compression of volumetric medical images. The EZW-approach has been improved upon by Said and Pearlman[9] with the SPIHT-algorithm for images. This approach has been extended to a 3D-SPIHT algorithm which was used for video coding and with excellent results also for medical volumetric data[17] using an integer wavelet packet transform.

While the above works aimed at compressing the entire volume in one piece, Ihm and Park[4] addressed also the adaptive decompression capability that is crucial for rapid visualization and which is considered in this paper. In their (block based) approach volume data was f rst divided in cubes of size $16 \times 16 \times 16$ called unit volumes, then 3D-transformed using the Haar-wavelet. The transformed coeff cients of each unit volume were then quantized and coded using so called cell tag table and cell information, where each cell represented a $4 \times 4 \times 4$ subregion of the unit volume. The encoding resulted in two bit streams: a code stream and an index stream, which facilitated direct access to each wavelet coeff cient. In a followup paper by Bajaj, Ihm, and Park[1] the method was improved and extended. The cell encoding was made more eff cient by a better encoding scheme for the signif cance map yielding gains in compression ratio of 10 to 15 percent and, more remarkably, speeding up the decoding by a factor of 5. Moreover, the technique was extended to handle RGB color volume data.

Recently, Rodler[8] considered the 3D volume encoding as a special case of video coding employing bidirectional temporal prediction of frames. The 2D wavelet transform using the Haar f lter was applied for the 2D slices. After transformation, coeff cients below a certain threshold were set to zero. For fast decoding, similar to the work of Ihm and Park[4] two bit streams were created, one eff ciently indicating the signif cance map exploiting the hierarchical structure of blocks in a transformed slice and the other encoding individual coeff cients. This method signif cantly improved the rate distortion performance over the results reported by Ihm and Park.

This paper proposes a new wavelet based method for compressing very large volume data, which enables good compression quality and fast decoding of any desired subvolume directly from the compressed bit stream for interactive visualization. As in the paper of Ihm and Park[4] we partition the volume data into unit volumes of size $16 \times 16 \times 16$ and apply a 3D wavelet transform. Our approach then differs in two aspects. First, instead of employing the Haar wavelets we apply the biorthogonal 9/7-tap Daubechies wavelet filter, which is well known for its superior performance in image compression applications.[15] Second, we propose a new fast and efficient method to encode the wavelet coefficients of a transformed unit volume. We can demonstrate an improvement over the method of Ihm and Park[4] of 11 to 12 dB in PSNR and about 6 to 8 dB PSNR over that of Rodler[8] for a large data set from the Visible Human project.

The rest of the paper is organized as follows. In Section 3 we briefly recall notions needed for the wavelet transform. Section 4 provides an overview of the proposed coding algorithm as well as a detailed pseudo code. In Section 5 experimental results and performance analysis will be given. Finally conclusions and future work are presented in Section 6.

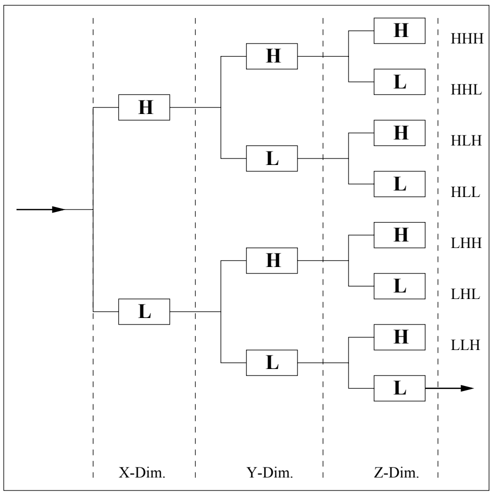

Figure 1: *Subband transform.*

3. Wavelets

Wavelets have provided very successful mathematical basis functions for representing and approximating signals because of their excellent localization properties in both space and frequency domain.[12] In the wavelet domain a signal is decomposed in several bands, one low resolution base band and several bands of increasing resolution for details in the signal. The great success of wavelet image and signal coding comes not only from the fact that the transform concentrates signal energy in only a few coefficients but is based — more importantly — on the structure of the localizations of the coefficients which are quantized to zero. This structure enables powerful coding schemes such as the EZW and SPHIT approaches that are mentioned in the last section. Today, wavelets are used in most state-of-the-art image compression algorithms. Also they have been proven useful in many areas of computer graphics.[11] An introduction to the theory and the algorithms for wavelet decompositions is beyond the scope of this paper. We refer the interested reader to the literature, e.g., the book by Strang and Nguyen.[12]

Sweldens[13] introduced the lifting scheme, an efficient implementation of wavelet transform algorithms. All classical wavelet transforms can be implemented using the lifting scheme by a factorization into so-called lifting steps.[3] Since the lifting steps speed up the transform, we use lifting to transform the unit volume. Here, the 1D lifting scheme will be first applied for all rows, then for all columns and then for the "temporal" direction (labeled "Z"), see Figure 1 for a schematic illustration. The output of the low pass filter "L" of the last part in this figure will be subsampled and decomposed just like the full signal in order to produce the second stage of the decomposition. As already mentioned we chose the biorthogonal 9/7-tap Daubechies wavelet filter for its known good performance.

4. Proposed algorithm

In this section, we describe our proposed algorithm to encode and decode scalar recti-linear volume data sets, i.e., we assume that scalar values are given on a 3D grid of some dimension $N_x \times N_y \times N_z$. At first, the volume data is segmented in to unit volumes of dimension $16 \times 16 \times 16$. Without great loss we may assume that the original dimensions N_x, N_y and N_z are multiples of 16. Otherwise we zeropad the volume data accordingly. Each unit volume will then be wavelet transformed and encoded separately. The linear unit size of 16 is suitable for the purpose of fast random access of portions of the entire data that are needed for a particular visualization. If the dimension is larger then 16, then decoding may produce a lot of volume data in the spatial domain that actually may not be needed for a particular user query, thus, wasting increased decoding time and storage space. On the other hand, if the dimension is less than 16, the unit volumes become so small that they cannot be encoded efficiently, i.e., for any given bit rate (bits per voxel) of the coder output the reconstruction quality will drop noticibly. Moreover, the number 16 is a power of 2 and therefore convenient for subband decomposition by the lifting scheme. The results of the encoding are two bit streams, the first one is the code stream which encodes the data and the other is an offset stream, which provides a directory indicating the starting position of the code bits for all unit volumes in the code stream. For visualization of a particular subvolume, volume slice, or isosurface one first needs to determine the required unit volumes, then extracts the corresponding binary code from the code stream using the directory, and finally decodes the data for each required unit volume.

The selection of the required unit volumes for the rendering of subvolumes of interest and for volume slices can be carried out using geometrical considerations. It is easy, e.g., to list all unit volumes that have a non-empty intersection with a given slice. To support also extraction of unit volumes for iso-surface visualization we append the minimum and maximum scalar voxels value of all unit volumes to the corresponding directory information in the offset stream. For a given iso-value the directory can be scanned and it is straightforward to report all those unit volumes that intersect the corresponding iso-surface.

In the remainder of this section we address the coding of an individual unit volume which must be designed to allow rapid compression and decompression as well as good rate-distortion performance. Let us f rst remark that there already exist compression techniques which achieve very good rate-distortion performance and are also very fast such as 3D-SPIHT.[17] But such methods should not be applied in our case for the following reason. The unit volume has a dimension of only 16. So the maximal eff cient decomposition level is two. In this case, the zero trees of wavelet coeff cients are not deep enough and the compression scheme does not work well. The zero tree method works well and achieves good results only when three or more decomposition levels are available.

Our proposed algorithm for encoding the small unit volumes of size $16 \times 16 \times 16$ is based on the following observation of transformed wavelet coeff cients. There exist not only zero trees, i.e., context between wavelet coeff cients of the same spatial location in different subbands, but also there is context between coeff cients in the neighborhood within a subband. Within a subband the intensity of wavelet coeff - cients typically does not change abruptly. There often exist large regions in a subband, where the intensities of all coef- f cients are below a some small value and, thus, only a few bits per coeff cient need to be encoded, uniformly in such a region. This leads to an approach that is not following the usual bit plane coding strategy, but all relevant bits for a signif cant coeff cient are coded consecutively. It also provides improved decoding speed.

The unit volume is wavelet transformed using the lifting scheme with two decomposition levels. After the transform we have 15 volume subbands as shown in Figure 2. The band with the bold outline (band 0) contains 4^3 coeff cients for the low frequency content of the volume block. Bands 1 through 7 each contain 4^3 detail coeff cients, while bands 8 to 14 each contain 8^3 coeff cients for detail at highest resolution.

The wavelet coeff cients for the 3D block are represented by f oating point numbers which f rst need to be quantized and binary coded for compression. We adopt uniform scalar quantization of coeff cients C using a f xed quantization step size $\Delta > 0$ with a dead zone of 2Δ around zero. More precisely, the quantization is given by

$$C \mapsto \text{sign}(C)\left(\left\lfloor \frac{|C|}{\Delta} \right\rfloor + \frac{1}{2}\right)\Delta.$$

The quantization step size is included as side information

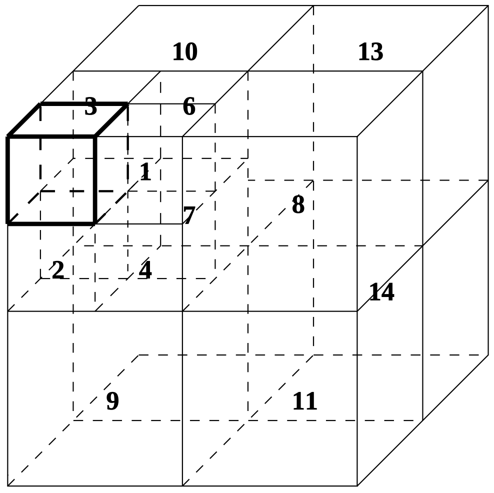

Figure 2: *The wavelet-transformed unit volume comprizes 15 bands.*

in the offset stream of the output code so that the decoder can retrieve the appropriate values for the dequantized coef- f cients. Thus, we replace each f oating point coeff cient C by the integer c given by

$$c = \text{sign}(C)\left\lfloor \frac{|C|}{\Delta} \right\rfloor.$$

For convenience we call these integer values c wavelet coef- f cients again. The decoder then dequantizes an integer coef- f cient c to the value $\hat{C} = \text{sign}(c)(|c| + \frac{1}{2})\Delta$. The choice of the quantization step size allows for coding at different bit rates, respectively compression ratios. Enlarging the step size Δ will achieve higher compression at reduced reconstruction quality. The quantization step size is the same for all unit volumes.

Next we organize the (integer) wavelet coeff cients into subbands and subsets of subbands as follows. For each of the 15 subbands we create a list of subsets. The lowest subband (subband 0) contains 4^3 coeff cients and is partitioned into 8 subsets each with 2^3 neighboring coeff cients from one of the $2 \times 2 \times 2$ cubes in the subband. Each of the remaining 14 lists initially contains only one entry, namely the entire set of 4^3 or 8^3 coeff cients from the corresponding subband. Later on, these sets may be subdivided into several subsets which will be appended to the appropriate lists. For the sets included in the 15 lists we use the notation V_k^n where n denotes the subband $n = 0, \ldots, 14$ and $k = 1, 2, \ldots$ runs as far as needed.

For the binary representations of the absolute coeff cient values bit planes 0 (least signif cant), 1, 2, and so on up to a maximum bit plane M are needed. The maximum M is given by $M = \lfloor \log_2(\max(|c_{i,j,k}|)) \rfloor$ where the maximum is taken over all coeff cients $|c_{i,j,k}|$ of the transformed unit volume.

```
Input
• A unit volume of size $16 \times 16 \times 16$ wavelet transformed with two stages (15 subbands, see Figure 2).
  Coeff cients are $C_{i,j,k}, i,j,k = 0,\dots,15$.
• A quantization step size $\Delta > 0$.
Initialization
1. Output $\Delta$.
2. Quantization and mapping: replace each coeff cient $C_{i,j,k}$ by integer $c_{i,j,k} = \mathrm{sign}(C_{i,j,k}) \lfloor |C_{i,j,k}|/\Delta \rfloor$.
3. Output maximal bit plane $M = \lfloor \log_2(\max(|c_{i,j,k}|)) \rfloor$.
4. For each subband $n = 1,\dots,15$ create the list $V^n$ of subsets $(V_1^n, V_2^n, \dots)$, see text for details.
Coding pass
• For all bit planes $m = M, M-1, \dots, 0$ do
    – For all subset lists $V^n, n = 0,\dots,14$, do
        ◦ For all subsets $V_k^n, k = 1,2,\dots$ of list $V^n$ do
            1. Signif cance: Output $b = S_m(V_k^n)$ (1 bit).
            2. Process signif cant subset: If $b = 1$ then
                a. If $|V_k^n| = 2^3$ then
                       Encode$(V_k^n, m)$
                   else
                       Partition $V_k^n$: Divide $V_k^n$ into 8 subsets and append these to the current list $V^n$.
                b. Remove $V_k^n$ from the list $V^n$.
Encoding routine Encode$(V,m)$
• Input: An integer $m \geq 0$ and a set $V$ of 8 integer coeff cients $c \in V$, $|c| < 2^{m+1}$
• For all $c \in V$ do
    1. Let $(-1)^{b_s} b_m b_{m-1} \dots b_0$ denote the binary representation of $c$ where $(-1)^{b_s} = \mathrm{sign}(c)$.
    2. Output $b_m, b_{m-1} \dots, b_0$ ($m+1$ bits)
    3. If $|c| > 0$ then output $b_s$ (1 bit)
```

Table 1: *Algorithm for encoding a wavelet transformed unit volume.*

The coding scheme operates by sequentially analyzing the coeff cient bit planes $m = M, M-1, \dots, 0$ beginning with the most signif cant one. At each bit plane m we sequentially scan each of the 15 lists of coeff cient subsets. A subset V_k^n is called signif cant w.r.t. bit plane m if it contains coeff - cients larger than or equal to 2^m in absolute value. Formally, $S_m(V_k^n) = 1$, where $S_m(\cdot)$ denotes signif cance to bit plane m,

$$S_m(V) = \begin{cases} 1 & \text{if } \max\{|c| \mid c \in V\} \geq 2^m \\ 0 & \text{otherwise} \end{cases}$$

for any set V of integers.

In the pass for each bit plane m only those subsets V_k^n are processed that are signif cant w.r.t. bit plane m, the others are ignored. Assume that V_k^n is a signif cant subset. If V_k^n contains 4^3 or 8^3 coeff cients then V_k^n is partitioned into eight subsets of size 2^3 or 4^3, respectively. These new subsets are appended to the list of subsets for the current subband n and will be processed again near the end of the current bit plane pass m. In the other case the subset contains precisely $2^3 = 8$ coeff cients c, which then will be encoded. First the absolute value $|c|$ is encoded using $m+1$ bits. If the coeff cient is not equal to zero then also the sign bit is included in the output. After processing a signif cant subset V_k^n we remove it from the list for subband n.

In the pseudo code we present the encoding algorithm for a unit volume in its entirety and more precisely. In particular additional bits must enter the code stream to indicate the location of the coeff cients in the output. The decoding is straightforward. Note that after receiving the bits for a coeff cient the decoder also knows which coeff cient these bits refer to.

5. Results

We have implemented our algorithm and report on experiments using a large volume data set that was originally extracted from the CT slices of "Visible Man" from the Visible Human Project. The resolution is $512 \times 512 \times 512$ with each voxel carrying a 12-bit gray scale value, stored in 2 bytes each and resulting in a total volume size of 256 Mbytes. The same data set was used for compression and visualization in the recent papers of Ihm and Park[4] and Rodler[8] so that we

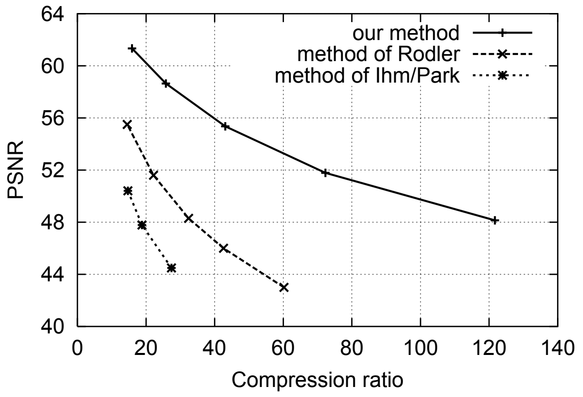

Figure 3: *Rate-distortion curves for the data in Table 2.*

method in [4]		method in [8]		our method	
ratio	PSNR	ratio	PSNR	ratio	PSNR
				121.7	48.14
		60.2	43.00	72.3	51.79
		42.6	46.00	43.1	55.36
27.4	44.49	32.4	48.30		
18.8	47.79	22.2	51.60	25.8	58.64
14.7	50.41	14.5	55.50	15.9	61.34

Table 2: *Compression ratios and qualities in PSNR for three methods. The measurements for the three methods were independent, and we arranged the results such that each row of the table contains comparable compression ratios.*

can provide an objective comparison of the performance of the various algorithms.

Using different quantization step sizes Δ we encode the data set at several compression ratios. The quality of the encoding is measured as usual in terms of the peak-signal-to-noise ratio (PSNR), expressed in decibel (dB),

$$\mathrm{PSNR} = 10\log_{10}\frac{(2^{12}-1)^2}{\frac{1}{512^3}\sum_{i,j,k=0}^{511}(v_{i,j,k}-\hat{v}_{i,j,k})^2}.$$

Here $v_{i,j,k}$ and $\hat{v}_{i,j,k}$ denote the original and reconstructed voxel intensities respectively.

The results are given in Table 2 and compared with results published in the other papers[4, 8]. Figure 3 displays the corresponding rate distortion curves. Our method is similar to that of Ihm and Park[4] in that it is based on encoding a total of 32768 wavelet transformed unit volumes of the same size 16^3. Our encoding algorithm is more eff cient yielding signif cant gains from 11 to 12 dB in PSNR. Compared to the approach of Rodler[8] which is based on encoding "motion compensated" wavelet transformed frame differences we still achieve large gains from 6 up to about 8 dB in PSNR. Therefore our encoding scheme is very successful in terms of data reconstruction quality. For a comparison of original and reconstruction Figures 4 and 5 show an original slice and the decoded slice of the volume compressed at a ratio of 43.1. Some small blocking artifacts may be seen due to the independent coding of unit volumes. compare the difference image in Figure 6.

quantization step size Δ	compression ratio	time per unit volume (msec)
128	121.7	1.70
96	72.3	1.95
64	43.1	2.05
32	25.8	2.25
16	15.9	2.60

Table 3: *Decoding times per unit volume in milliseconds. The results were attained using different quantization step sizes Δ, which are listed in the first column.*

Besides image quality the aspect of rapid compression/decompression is of importance for visualization applications. Here we report our timing results both for encoding the entire volume and for decoding a single unit volume. These measurements were carried out on a computer with a 600 MHz Pentium III processor and 196 Mbytes of CPU memory.

For the encoding process 16 slices of the volume are read into memory at a time after which the corresponding 1024 unit volumes can be extracted and encoded. The time to read in the entire data set of 256 Mbytes in this way is about 80 seconds. The time to encode the data depends on the choice of the quantization step size respectively the desired compression ratio The entire compression time is about 100 seconds, which amounts to a compression throughput of about 330 unit volumes per second or 1.35 million voxels per second.

While the compression is already fast, the decoding is yet faster. This is due to reduced complexity in that the decoder does not need to check any sets of coeff cients for signif-icance. Essentially, it just unpacks the bits from the code stream and sorts them into a 3D array. Overall, the decoding throughput is about 500 unit volumes or 2 million voxels per second. The measured times per unit volume are given in Table 3.

For visualization of a complete volume slice that is parallel to one of the three coordinate hyperplanes in 3-space one needs to decode a total of 1024 unit volumes. Thus, the decoding as a preprocess for the visualization processes 1024 unit volumes at about 2 milliseconds per unit volume taking in total only about 2 seconds. For slices that are not paral-

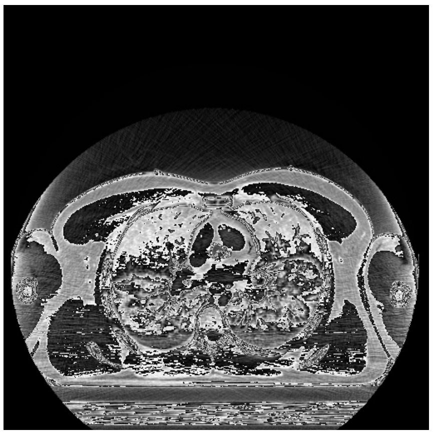

Figure 4: *An original volume slice.*

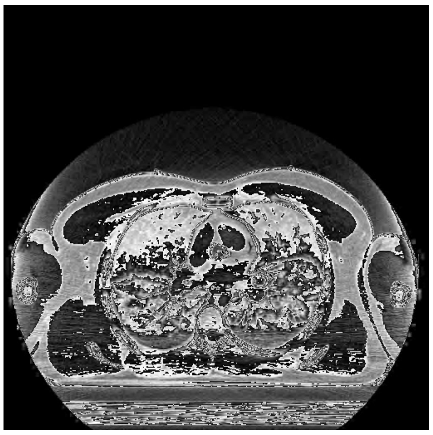

Figure 5: *The slice from Figure 4 reconstructed from the compressed volume data at compression ratio 43.1.*

Figure 6: *Difference image between original in Figure 4 and reconctruction in Figure 5. The grey levels are* $128+\delta_g/2$ *where* δ_g *is the grey level difference.*

lel to a coordinate hyperplane a slightly increased amount of decompression time is necessitated.

It is diff cult to compare our decoding time complexity to those of the methods presented in the other papers. Neither paper[4, 8] presents compression times, and only the paper of Rodler[8] lists decoding times, however, only for the entire volume. When comparing these numbers with ours we f nd that the decoding times are similar, around one minute for the whole volume. However, this comparison is of limited value, since the machines that were used for the measurements were not the same and the programs used may have been optimized with different efforts.

In the paper of Bajaj et al[1] utmost emphasis was put on decoding speed. Their data sets and computer (SGI 195 MHz MIPS R10000 CPU) differ from ours so that the speed is hard to compare. Moreover, the data set differs in type since it includes color information which was also encoded. When decoding entire cells of dimension $4 \times 4 \times 4$ a rate of about 10 million voxels per second was reported, which is about 5 times as much as in our implementation. For random access of individual voxels the difference in speed should be even larger. However, for extraction of slice, iso-surface, or subvolume data, there is a strong coherence in the voxel locations and such extreme random access performance is not necessarily required. When comparing our algorithm with that of Bajaj et al, we also take into account the achievable compression ratios, which in Bajaj et al's paper is 10 to 15 percent above that what was obtained using the algorithm of the paper of Ihm and Park. Out method, however, outperforms that one by about 500 percent, see Figure 3, producing a six-fold compression ratio at 50.41 dB PSNR.

6. Conclusion and future work

We have proposed a new method for compression of very large volume data sets with scalar data on a rectilinear grid. The method is simple yet eff cient both in terms of the rate-distortion performance and with respect to time complexity. A comparison with recently published results for the same test data set reveals superior reconstruction quality up to more than 8 dB in PSNR at the same compression ratios. We anticipate to extend our work in the following directions:

- As with any block based coder there are small blocking artifacts in the resulting decompressed volume data. We will introduce suitable post f ltering techniques that will further improve the measurable performance in PSNR as well as in terms of psychovisual quality.
- We intend to implement the 3D-SPIHT method to encode the entire volume in one piece. This is not useful for the visualization purposes addressed here. However, the resulting compression performance can be regarded as an upper limit of what one possibly can achieve by more simple and fast block based encoding schemes that one needs for the volume visualization tasks on hand.

- In order to further improve the decoding speed of out method we consider implementing the encoder using shorter wavelet f lters including the Haar wavelet.
- We plan to build a prototype volume visualizer for large scale volume data based on the compression principles presented in this paper.
- Extensions of these methods to multi-valued or vector f elds over three- and higher-dimensional domains will be pursued.

All in all it would be desirable to have a scalable method that can be adapted to the user's priorities which may be emphasizing decompression speed, compression ratio, or reconstruction quality.

7. Acknowledgments

We thank Insung Ihm, Sogang University, Korea, for providing the specially prepared Visible Human data set, which allowed us to compare our results to those reported in other papers.

References

1. C. Bajaj, I. Ihm, S. Park. "3D RGB compression for interactive applications". Accepted for publication in ACM Transactions on Graphics, 2001.
2. B. Belzer.
3. I. Daubechies, W. Sweldens. "Factoring wavelet transforms into lifting steps". *Technical report, Bell Laboratories, Lucent Technologies*, 1996.
4. I. Ihm, S. Park. "Wavelet-based 3D compression scheme of interactive visualization of very large volume data". *Computer Graphics Forum* 18,1 (1999) 3–15.
5. J. Luo, X. Wang, C. W. Chen, K. J. Parker. "Volumetric medical image compression with three-dimensional wavelet transform and octave zerotree coding". In: Visual Communication and Image Processing'96, Proc. SPIE 2727, pages 579–590, March 1996.
6. S. Mitra, S. Yang, V. Kustov. "Wavelet-based adaptive vector quantization for high-f delity compression and fast transmission of medical images". *J. Digit Imaging* 11,4 (Suppl 2) (1998) 24–30.
7. S. Muraki. "Volume data and wavelet transform". *IEEE Computer Graphics and Application* 13,4 (1993) 50–56.
8. F. Rodler. "Wavelet based 3D compression with fast random access for very large volume data." *Proceedings, Pacific Graphics'99* , pp. 108–117, IEEE Press, 1999.
9. A. Said, W. A. Pearlman. "A new, fast and eff cient image codec based on set partitioning in hierarchical trees". *IEEE Trans. Circuit and Sys. For Video Techno.*, 6,3 (1996) 243–250.
10. J.M. Shapiro. "Embedded image coding using zerotrees of wavelet coeff cients". *IEEE Trans. on Signal Processing* 41 (1993) 3445–3462.
11. E. Stollnitz, T. DeRose, D. Salesin. "Wavelets for Computer Graphics: Theory and Application". *Morgan Kaufmann Publisher* (1996).
12. G. Strang, T. Nguyen. "Wavelets and Filter Banks". *Wellesley-Cambridge Press* (1997).
13. W. Sweldens, "The lifting scheme: A custom-design construction of biorthogonal wavelets". *Journal of App. and Computer Harmonic Analysis*, 3 (1996) 186–200.
14. G.R. Thoma, L.R. Long. "Compressing and transmitting visible human images". *IEEE Multimedia* 4,2 (1997) 36–45.
15. J.D. Villasenor, B. Belzer, J. Liao. "Wavelet f lter evaluation for image compression". *IEEE Transactions on Image Processing 4 (1995) 1053–1060.*
16. "The Visible Human Project". *The National Library of Medicine*, Bethesda. www.nlm.nih.gov/research/visible/visible_human.html.
17. Z. Xiong, X. Wu, D. Y. Yun, W. Pearlman. "Progressive coding of medical volumetric data using three-dimensional integer wavelet packet transform". *Visual Communications and Image Processing'99*, pp. 327–335, (1999).