Resolution Scalable Coding and Region of Interest Access with Three-Dimensional SBHP Algorithm

Ying Liu and William A. Pearlman

Electrical Computer and Systems Engineering Department
Rensselaer Polytechnic Institute
Troy, NY 12180

Abstract

A low-complexity three-dimensional image compression algorithm based on wavelet transforms and set-partitioning strategy is presented. The Subband Block Hierarchial Partitioning (SBHP) algorithm is modified and extended to three dimensions, and applied to every code block independently. The resultant algorithm, 3D-SBHP, efficiently encodes 3D image data by the exploitation of the dependencies in all dimensions, while enabling progressive SNR and resolution decompression and Region-of-Interest (ROI) access from the same bit stream. The code-block selection method by which random access decoding can be achieved is outlined.The resolution scalable and random access performances are empirically investigated. The results show 3D-SBHP is a good candidate to compress 3D image data sets for multimedia applications.

1. Introduction

Three dimensional (3-D) data sets, such as hyperspectral images and medical volumetric data generated by computer tomography (CT) or magnetic resonance (MR) typically contains many image slices that requires huge amount of storage. For modern multimedia applications, particularly in the Internet environment, efficient compression techniques are necessary to reduce storage and transmission bandwidth. Furthermore, it is highly desirable to have properties of SNR and resolution scalability and ROI retrievability with a single embedded bitstream per data set in many applications. SNR scalability gives the user an option of lossless decoding, which is important for analysis and diagnosis, and also allows the user to reconstruct image data at lower rate or quality to get rapid browsing through a large image data set. Resolution scalability means that a portion of the bit stream can be decoded to reconstruct the image data to the desired level of resolution. It provides image browsing with low memory cost and computational resources. For some applications, only a subsection of the image sequence is selected for analysis or diagnosis. Therefore, it is very important to have region of interest retrievability, which can greatly save decoding time and transmission bandwidth.

Although 3-D image data can be compressed by applying two-dimensional compression algorithm to each slice independently, the high correlation between slices makes an algorithm based on three-dimensional coding a better choice. To provide scalability and compression efficiency , many volumetric image compression algorithms based on wavelet transform were proposed recently, such as Three-Dimensional Context-Based Embedded Zerotree of Wavelet coefficient(3D-CB-EZW) [1] and Three-Dimensional Set Partitioning In Hierarchical Trees(3D-SPIHT)[2], Asymmetric Tree 3D-SPIHT (AT-3D-SPIHT) [11], Three-Dimensional Set Partitioned Embedded bloCK (3D-SPECK)[8], Three-dimensional Tarp [9], and Annex of Part II of JPEG2000[10] standard for multi-component imagery compression. Most of those algorithms are unable to naturally provide random access functionality or resolution scalability due to their data structure across different subbands.

SBHP is a low complexity alternative to JPEG2000. It can support all the features planned for JPEG2000, such as progressive transmission by resolution, quality, location; random access and lossy-to-lossless compression. Its encoder runs about 4 times faster, and the decoder is about 6 to 8 times faster than JPEG 2000 with only a small loss in compression performance. Here, in this paper, we extend SBHP to three dimensions. The 3D-SBHP is based on coding 3-D subblocks of 3-D wavelet subbands and can provide the aforementioned functionality and fast encoding and decoding.

JPEG2000 uses three ROI coding methods: tiling, coefficient scaling and code-block selection. Since code-block selection does not require the ROI be determined and segmented before encoding, the image sequence is encoded only once and the decoder can extract a subset of the bit stream to reconstruct the image region of required spatial location and quality. This gives the user the flexibility at decode time, which is vital to some applications, such as client/server applications.

In this paper, we investigate the scalability and the ROI access performance of 3D-SBHP in detail.

The rest of this paper is organized as follows. We present the scalable 3D-SBHP algorithm and code-block selection ROI access scheme in Section 2. Experimental results of scalable coding, ROI decoding and computational complexity are given in Section 3. Section 4 will conclude this study.

2 Scalable 3D-SBHP

2.1 Coding Algorithm

The 2-D SBHP algorithm is a SPECK[4] variant which was originally designed as a low complexity alternative to JPEG2000. 3-D SBHP is a modification and extension of 2-D SBHP to three-dimensions. In 3-D SBHP, each subband is partitioned into code-blocks. All code-blocks have the same size. 3-D SBHP is applied to every code-block independently and generates a highly scalable bit-stream for each code-block by using the same form of progressive bit-plane coding as in SPIHT[5].

Consider a 3D image data set that has been transformed using a discrete wavelet transform. The image sequence is represented by an indexed set of wavelet transformed coefficients $c_{i,j,k}$ located at the position (i, j, k) in the transformed image sequence. Following the idea in[6], for a given bit plane n and a given set B of coefficients, we define the significance function:

$$S_n(B) = \begin{cases} 1, \; if \; (\max\limits_{(i,j,k)\in B} |c_{i,j,k}|) \geq 2^n, \\ 0, \; otherwise. \end{cases} \qquad (1)$$

Following this definition, we say that set B is significant with respect to bit plane n if $S_n(B) = 1$. Otherwise, we say that set B is insignificant.

3-D SBHP is based on a set-partitioning strategy. The set-partitioning process used by 3-D SBHP is almost the same as that used by 2-D SBHP. Below we explain in detail the partition rules by using a $16 \times 16 \times 4$ code-block as an example.

The algorithm starts with two sets, as shown in Figure 1(a). One is composed of the $2 \times 2 \times 1$ top-left wavelet coefficient in the first frame, and the other contains the remaining coefficients. In the first set partitioning stage, the first set can be decomposed into 4 individual coefficients and the second set can be decomposed into three $2 \times 2 \times 1$ groups and the remaining coefficients, as shown in Figure 1(b). Figure 1(c) shows the second stage of set partitioning, each $2 \times 2 \times 1$ group can be decomposed into 4 coefficients,

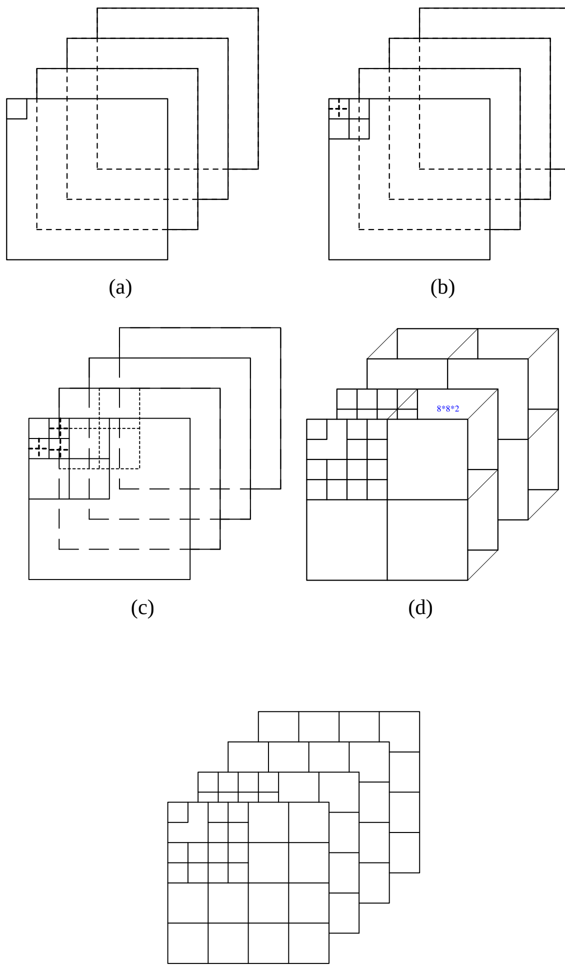


Figure 1: Set partitioning rules used by 3-D SBHP.

and the remaining set can be split into seven 4×4 groups and a remaining set. In the third stage, as shown in Figure 1(d), each $4 \times 4 \times 1$ group is split into 4 $2 \times 2 \times 1$ groups, and the remaining set is partitioned in seven $8 \times 8 \times 2$ groups. Figure1(e) shows each $2 \times 2 \times 1$ group can be decomposed into 4 coefficients, and each $8 \times 8 \times 2$ group can be split into eight 4×4 groups. This process continues until all sets are partitioned to individual coefficients.

During the coding process a set is partitioned following the above rules when at least one of its subsets is significant. To minimize the number of significant tests for a given bit-plane, 3-D SBHP maintains three lists:

- LIS(List of Insignificant Sets) - all the sets(with more than one coefficient) that are insignificant but do not belong to a larger insignificant set.

- LIP(List of Insignificant Pixels) - pixels that are insignificant and do not belong to insignificant set.

- LSP(List of Significant Pixels) - all pixels found to be significant in previous passes.

Instead of using a single large LIS that has sets of varying sizes, we use an array of smaller lists of type LIS, each containing sets of a fixed size. All the lists and list arrays are updated in the most efficient list management method - FIFO. Since the total number of sets that are formed during the coding process remain the same, using an array of lists does not increase the memory requirement for the coder. Use of multiple lists completely eliminates the the need for any sorting mechanism for processing sets in increasing order of their size and speeds up the encoding/decoding process. For each new bit plane, significance of coefficients in the LIP are tested first, then the sets in the LIS in increasing order of their sizes, and lastly the code refinement bits for coefficients in LSP. The idea behind processing LIP and LIS in increasing size can be seen as a way to achieve the same effect as the fractional bitplane coding used in JPEG2000 without having to go through several passes through the bitplane.

The way 3D-SBHP entropy codes the comparison results is an important factor that reduces the coding complexity. Instead of using adaptive arithmetic or Huffman coding, 3D-SBHP uses three 15-symbol Huffman codes to code the set partitioning results of the significant S sets. Fixed Huffman coding avoids identifying the current context model, updating models and calculating frequency. These 15-symbol Huffman codes have the longest codeword to be 6 bits, and we can use lookup tables instead of binary trees for speeding up decoding. For comparison results of coefficients, raw bits are written to the compressed stream.

2.2 Scalable Coding

3D-SBHP is applied independently to every code-block inside a subband . An embedded bit stream is generated by bitplane coding and the method has the same effect as fractional bitplane coding. To enable SNR scalability, bit stream boundaries are maintained for every bit plane, as shown in Figure 2. To get SNR scalability, bits belonging to the same fraction of the same bit planes in the the different code-blocks are extracted for decoding.

In wavelet coding systems, resolution scalability enables step increases of resolution when bits in higher frequency subbands are decoded. After N levels of wavelet decomposition, the image has N resolution scales. As shown in Figure 2, 3D-SBHP codes code-blocks from lowest subband to the highest subband. The algorithm generates progressive bit stream for each code-block, and the whole bit stream is resolution scalable. If a user wants to decode up to resolution n, bits belonging to the same fraction of the same bit planes in the code-blocks related to resolution n can be extracted for decoding.

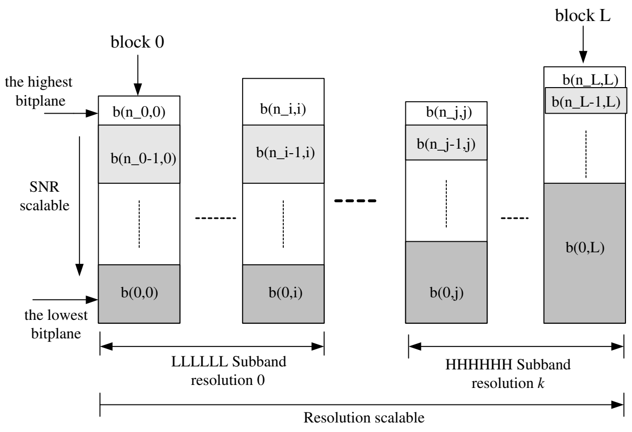


Figure 2: An example of 3D-SBHP SNR and resolution scalable coding. Each bitplane α in block β is denoted as $b_{\alpha,\beta}$. Code-blocks are encoded and indexed from the lowest subband to the highest subband.

2.3 Random Access Decoding

This section describes how to apply 3-D SBHP to achieve ROI access. Consider an image sequence which has been transformed using a discrete wavelet transform. The transformed image sequence exhibits a hierarchical pyramid structure. The wavelet coefficients in the pyramid subband system are spatially correlated to some region of the image sequence. In 3-D SBHP, code-blocks are of a fixed size, and represent an increasing spatial extent at lower frequency subband. Figure 3 gives an example of the parent-offspring dependencies in the 3D spatial orientation tree after 2-level wavelet packet decomposition(2D spatial + 1D temporal). Except those coefficients in the lowest spatial and temporal subband, every coefficient located at (i, j, k) has its unique parent at $(\lfloor\frac{i}{2}\rfloor, \lfloor\frac{j}{2}\rfloor, \lfloor\frac{k}{2}\rfloor)$ in the lower subband. All coefficients are organized by trees with roots located in the lowest subband.

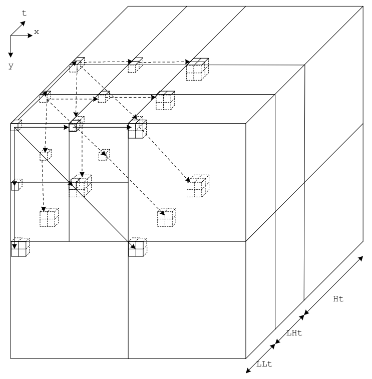


Figure 3: Parent-offspring dependencies in the 3D orientation tree.

In this paper, we consider retrieving a cubic region in a image sequence, where A denotes the upper-left corner in the first frame of the cubic region and B denotes the lower-right corner in the last frame of the cubic region. Since the wavelet transform is separable, we first consider the random access problem in one dimension.

Let $[x_A, x_B)$ denote the range of the cubic region in the X direction. Let $[x^F_{k,l}, x^R_{k,l})$ denote the X-direction interval that is related to the cubic region at DWT level k in low-pass or high-pass subbands. Let $l = \{0, 1\}$ represent the low-pass and high-pass subband respectively. Suppose the volume size of the image sequence is $W \times H \times D$. If we do not consider the filter length, the boundaries of each interval can be found recursively using

$$x^F_{k,l} = \lfloor \frac{x^F_{(k-1),0}}{2} \rfloor + l \times \frac{W}{2^k}, \quad x^R_{k,l} = \lceil \frac{x^R_{(k-1),0}}{2} \rceil + l \times \frac{W}{2^k}$$

$$x^F_{0,0} = x_A, \quad x^R_{0,0} = x_B$$

The spatial error penetration of the filter length effect around edges can be calculated from the wavelet filter length and level of wavelet decomposition. Topiwala[7] gives an approximate equation of the error penetration, by which the spread of the error D (in pixels) as a function of the wavelet filter length L and the number of wavelet decomposition levels K is given by

$$D(K, L) = \begin{cases} (2^K - 1)(2^{\frac{L-3}{2}} + 1), L \text{ even} \\ (2^K - 1)(2^{\frac{L-2}{2}} + 1), L \text{ odd} \end{cases} \tag{2}$$

Suppose we have a synthesis filter with filter lenght $L = M + N + 1$,

$$g_n = \sum_{i=-M}^{N} a_i \times f_{n+i}$$

the boundaries of each interval can become

$$x^F_{k,l} = \max\{0, \lfloor \frac{x^F_{(k-1),0} - M}{2} \rfloor\} + l \times \frac{W}{2^k},$$

$$x^R_{k,l} = \min\{\lceil \frac{x^R_{(k-1),0} + N}{2} \rceil, \frac{W}{2^k} - 1\} + l \times \frac{W}{2^k}$$

$$x^F_{0,0} = x_A, \quad x^R_{0,0} = x_B$$

Similarly, the boundaries of each interval in Y direction, $[y^F_{k,l}, y^R_{k,l})$, and in temporal direction, $[z^F_{k,l}, z^R_{k,l})$, can be found following the same principle.

Suppose that an image sequence is decomposed at level K in spatial domain and level T in temporal domain with synthesis filter length L and coded with code-block size $O \times P \times Q$. To reconstruct a $X \times Y \times Z$ ($Z \leq GOPsize$) 3D region, where

$$X = x^R_{0,0} - x^F_{0,0}, \quad Y = y^R_{0,0} - y^F_{0,0}, \quad Z = z^R_{0,0} - z^F_{0,0}.$$

The number of decoded code-blocks, denoted as N_B, is given below.

$$N_B = \sum_{j=1}^{K} \left(\sum_{l=1}^{S} s \times \left(\lceil \frac{x^R_{j,l}}{O} \rceil - \lfloor \frac{x^F_{j,l}}{O} \rfloor + 1 \right) \times \left(\lceil \frac{y^R_{j,l}}{P} \rceil - \lfloor \frac{y^F_{j,l}}{P} \rfloor + 1 \right) \right.$$
$$\left. \times \sum_{i=1}^{T} \sum_{n=1}^{t} \left(\lceil \frac{z^R_{i,n}}{Q} \rceil - \lfloor \frac{z^F_{i,n}}{Q} \rfloor + 1 \right) \right)$$

where,

$$t = \begin{cases} 1, i < T \\ 0, i = T \end{cases} \quad S = \begin{cases} 1, j < K \\ 0, j = K \end{cases} \quad s = \begin{cases} 3, S = 1 \\ 1, S = 0 \end{cases} \tag{3}$$

For example, a $32 \times 32 \times 4$ 3D region is positioned at row 64, column 90, in frame number 5 of a image sequence which is decomposed at level 2 with synthesis filter length 3 and coded with code-block size $16 \times 16 \times 2$. If we do not consider filter length, 96 code-blocks are needed for reconstruction the ROI region. Here we call these code-blocks *nonfilter-length related ROI code-blocks*. To losslessly reconstruct this region, 156 code-blocks are needed. Here, 60 more code-blocks are used for lossless reconstruction. In this paper, we call these extra code-blocks *filter-length related code-blocks*. Since subband transforms are not shift invariant, the same 3D region positioned at different locations may need different numbers of code-blocks for reconstruction.

In 3D SBHP, a 3D region can be independently reconstructed with blur at the boundaries of that region if we only select nonfilter-length related ROI code-blocks and the synthesis filter length is larger than two. In addition, the decoder can also extract filter-length related code-blocks in order to correctly perform the inverse discrete wavelet transform and construct the 3D region losslessly.

3 Numerical Results

We conduct our experiments on 4 8-bit CT medical image volumes, 4 8-bit MR medical image volumes, and 4 16-bit Airborne Visible InfraRed Imaging Spectrometer (AVIRIS) hyperspectral image volumes. AVIRIS has 224 bands and 614×512 pixel resolution . For our experiments, we cropped the scene to $512 \times 512 \times 224$ pixels. Table 1 shows the description of these sequences.

In this section, we provide simulation results and compare the proposed 3-D volumetric codec with other algorithms.

3.1 Comparison of Lossless Performance with Different Algorithms

Table 2 compares the lossless compression performance JPEG2000 on medical image sequences of the following

File Name	Image Type	Volume Size	Bit Depth (bit/pixel)
Skull	CT	$256 \times 256 \times 192$	8
Wrist	CT	$256 \times 256 \times 176$	8
Carotid	CT	$256 \times 256 \times 64$	8
Aperts	CT	$256 \times 256 \times 96$	8
Liver_t1	MR	$256 \times 256 \times 48$	8
Liver_t2e1	MR	$256 \times 256 \times 48$	8
Sag_head	MR	$256 \times 256 \times 48$	8
Ped_chest	MR	$256 \times 256 \times 64$	8
moffett scence 1	AVIRIS	$512 \times 512 \times 224$	16
moffett scence 2	AVIRIS	$512 \times 512 \times 224$	16
moffett scence 3	AVIRIS	$512 \times 512 \times 224$	16
jasper scence 1	AVIRIS	$512 \times 512 \times 224$	16

Table 1: Description of the image volumes

compression algorithms: AT-3D-SPIHT, 3D-SPECK, 3D-CB-EZW, 3D-SBHP, JPEG2000 multi-component and (2D lossless) JPEG2000.

To get these results, 3D-SBHP and AT-3D-SPIHT use $GOS = 16$, while other 3D algorithms treat the entire image sequence as one coding unit. The code-block size used by 3D-SBHP is $64 \times 64 \times 4$. For all 3D algorithms, the three level wavelet transform was applied on all three dimensions using the I(2+2,2) filter. JPEG2000 multi-component first applied the I(2+2,2) filter on axial domain, then coded every resultant spectral slice as separate file by Kakadu JPEG2000 [12] which uses integer 5/3 filter.

Comparing the average compression performance listed in the last row of the table, JPEG2000 multi-component gives the best coding efficiency. As an extension of SBHP, a low-complexity alternative to JPEG2000, 3D-SBHP on average yields 23% higher compression performance than 2D JPEG2000, and is 13% worse than JPEG2000 multi-component. Compared with the average compression results of other 3D algorithms, 3D-SBHP is 2%, 5% and 13% worse than 3D-SPECK, AT-3D-SPIHT and 3D-CS-EZW, in compression efficiency, respectively. On the other hand, 3D-SBHP outperforms most algorithms on some sequences. If we consider the fact that 3D-SBHP is applied with $GOS = 16$, while other 3D algorithms use the whole sequence as coding unit, a smaller performance gap will be expected.

Table 3 presents the lossless performances of 3D-SBHP, 3D-SPIHT, 3D-SPECK, JP2K-Multi, 2D-SPIHT and JPEG 2000 on hyperspectral image sequences. 3D-SBHP uses five-level dyadic S+P(B) filter on spatial domain and two-level 1D S+P(B) filter on the spectral axis with $GOS = 16$ and code-block size $= 64 \times 64 \times 4$. JP2K-Multi is implemented first by applying the S+P filter on spectral dimension and is then followed by application of the 2D JPEG 2000 on the spatial domain using the integer filter(5,3). For all other 3D algorithms, all 224 bands are coded as a single unit and five-level filter are applied on every dimension.

For AVIRIS test image volumes, 3D-SPIHT gives the best coding efficiency. 3D-SBHP is comparable to 3D-SPIHT on AVIRIS image sequence. On average, it is only about 2% inferior to 3D-SPIHT and 3D-SPECK. Our algorithm yields, on average, about 2%, 13% and 17% higher compression efficiency than JPEG2000 multi-component, 2D-SPIHT and JPEG2000, respectively. Again, we sacrifice coding efficiency to gain random accessibility and low memory usage by using $GOS = 16$.

File Name	AT-3D-SPIHT	3D-SBHP	3D-SPECK	3D-CB -EZW	JP2K-Multi	JPEG 2000
Skull	2.1754	2.2701	2.0170	2.0095	1.7450	2.9993
Wrist	1.3083	1.4002	1.2538	1.1393	1.1771	1.7648
Carotid	1.5844	1.6631	1.6517	1.3930	1.6785	2.0277
Aperts	1.0370	1.0876	1.1502	0.8923	0.7290	1.2690
Liver1	2.3191	2.5257	2.4331	2.2076	2.3814	3.2640
Liver2	1.7868	1.8477	1.8733	1.6591	1.6247	2.5804
head	2.2071	2.3219	2.3589	2.2846	2.5961	2.9134
chest	1.9629	2.0873	2.1160	1.8705	1.4884	3.1106
Average	1.7976	1.9004	1.8567	1.6820	1.6775	2.4912

Table 2: Comparison of different coding methods for lossless compression of 8-bit medical image volumes (bits/pixel).

File Name	3D-SPIHT	3D-SBHP	3D-SPECK	JP2K-Multi	2D-SPIHT	JPEG 2000
moffett 1	6.9411	7.0333	6.9102	7.1748	7.9714	8.7905
moffett 2	7.9174	8.4333	8.0835	8.4131	9.8503	10.0815
moffett 3	6.7402	6.8359	6.8209	7.0021	7.5874	7.7258
jasper 1	6.7157	6.7842	6.7014	6.8965	7.7977	8.8560
Average	7.0786	7.2716	7.1290	7.3716	8.3458	8.7959

Table 3: Comparison of different coding methods for lossless coding of 16-bit AVIRIS image volumes (bit/pixel).

3.2 Lossless coding performance by use of different code-block sizes

Table 4 compares the lossless compression results for all image data listed in Table 1 by using different code-block sizes: $8 \times 8 \times 2$, $16 \times 16 \times 2$, $32 \times 32 \times 4$ and $64 \times 64 \times 4$. The image sequences are compressed with GOS = 16 and I(2,2) filter. Three level of wavelet decompositon is applied on all three dimensions. The results show that for all image sequences, increasing the size of the code-block improves the performance somewhat. The main reason for the improvement of coding efficiency is that larger code-block size decreases the total overhead for the whole image sequence.

File Name	$8 \times 8 \times 2$	$16 \times 16 \times 2$	$32 \times 32 \times 4$	$64 \times 64 \times 4$
Skull	3.1066	2.4758	2.2617	2.2301
Wrist	2.1780	1.5601	1.3604	1.3347
Carotid	2.5093	1.8973	1.6952	1.6684
Aperts	1.8857	1.2718	1.0793	1.0525
Liver_t1	3.3724	2.7478	2.5287	2.5001
Liver_t2e1	2.6961	2.0709	1.86613	1.8354
Sag_head	3.1859	2.5538	2.3395	2.3091
Ped_chest	2.8729	2.2502	2.0372	2.0081
moffett 1	8.3711	7.5104	7.2282	7.1848
moffett 2	9.8242	8.9170	8.6086	8.5674
moffett 3	8.0128	7.1722	6.8960	6.8536
jasper 1	8.1417	7.2922	7.0130	6.9705

Table 4: Lossless Coding Results by Use of Different Code-block Size (bits/pixel)

3.3 Resolution scalable results

The CT medical sequence "skull" , I(2,2) integer filter, and $32 \times 32 \times 4$ code-block size are selected for this comparison. The quality of reconstruction is measured by peak signal to noise ratio (PSNR) over the whole image sequence. PSNR is defined by

$$PSNR = 10 \log_{10} \frac{x_{peak}^2}{MSE} dB \tag{4}$$

where $x_{peak} = 255$ for these medical images and MSE denotes the mean squared-error between the original and reconstructed slice. Figure 4 shows the reconstructed CT_skull sequence decoded from a single scalable code stream at a variety of resolution at 1.0 bpp. The PSNR values listed in Table 5 for low resolution image sequences are calculated with respect to the lossless reconstruction of the corresponding resolution. Table 5 shows that the PSNR values decrease from one resolution to the next lower one, while the total byte cost decreases rapidly with successive reductions in resolution as shown in Table 6. We can see that the computational cost and memory requirement of decoding reduces from one resolution level to the next lower one.

Bit Rate	PSNR (dB)		
	1/4 resolution	1/2 resolution	FULL
0.25	11.10	23.46	37.63
0.5	13.77	29.03	41.85
1.0	24.04	35.71	46.50
2.0	32.58	43.88	50.55

Table 5: PSNR for decoding CT_skull at a variety of resolutions and bit rates

Figure 4 demonstrates the first reconstructed slice of the reconstructed sequence which is decoded at 1.0 bpp to a variety of resolutions. Even at a low resolution, we can get a clear view of the image sequence.

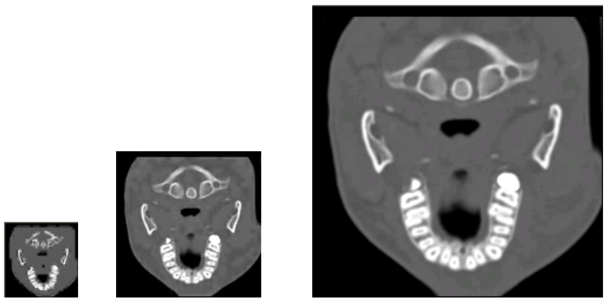


Figure 4: A visual example of resolution scalable decoding. From left to right: 1/4, 1/2 and full resolution at 1.0 bpp

Bit Rate	Byte Budget (bytes)		
	1/4 resolution	1/2 resolution	FULL
0.25	6132	48951	391351
0.5	12284	98258	785364
1.0	24575	196604	1572597
2.0	49151	39321	3145610
Lossless	137333	757110	3725185

Table 6: Byte used for decoding CT_skull at a variety of resolutions and bit rates

3.4 Random Access Decoding Results

In this experiment, we randomly chose a $64 \times 64 \times 16$ region from (134, 117, 17) to (198, 181, 32) of CT_skull sequence as the ROI. In order to decode this region, we only need to apply 3D-SBHP with the code-block selection method described in Section 2.3. For a given bit rate, the bit stream is truncated at the same fraction of the same bit plane for all selected code-blocks. In Table 7, we compare the PSNR performance of 3D-SBHP random access decoding at different code-block sizes and bit rates. The byte and number of code-blocks used for lossless decoding the ROI region are listed in Table 8. These two tables show that a smaller code-block can give higher ROI decoding performance, especially at high bit rate, while decreasing the overall compression efficiency. Therefore, the trade-off between compression efficiency and random accessibility should be considered.

Table 8 gives the number of bytes and code-blocks used for lossless reconstruction of the ROI region. As filtering is a spatially expansive operation, the samples that need to be retrieved always exceed the number of samples in the ROI region.

Figure 5(a) and Figure 5(b) give both 2D and 3D visual example of ROI decoding. In the 3D example the region of ROI is from (134, 117, 17) to (198, 181, 112).

Bit Rate (bpp)	Code-block Size	
	$32 \times 32 \times 4$	$16 \times 16 \times 2$
0.5	22.13 dB	21.71 dB
1.0	26.44 dB	26.80 dB
2.0	32.04 dB	33.81 dB
4.0	38.22 dB	40.63 dB

Table 7: PSNR for random access decoding of a $64 \times 64 \times 16$ region of CT_skull at a variety of code-block size and bit rates

Code-block Size	Byte Budget (bytes)	Code-block
$32 \times 32 \times 4$	144788	88
$16 \times 16 \times 2$	123143	440

Table 8: Bytes and code-blocks used for lossless decoding ROI

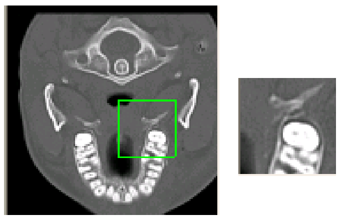
3.5 Computational Complexity

One of the main advantages of 3D-SBHP is its fast encoding and decoding. 3D-SBHP has been implemeted using standard C++ language and complied by VC++.NET compiler. Tests are performed on a laptop with Intel 1.50GHz Pentium M processor and Microsoft Windows XP. The coding speed is measured by CPU cycles. The RDTSC (read-time stamp counter) instruction is used for cycle count.

CT_Skull and MR_liver_t1 are selected for test. 3D-SBHP ues $GOS = 16$ and code-block size = $32 \times 32 \times 4$, while AT-3D-SPIHT codes the whole sequence as a coding unit. Three-levels of spatial dyadic integer wavelet transform and two-levels temporal integer wavelet transform are applied on all image sequences by using I(2,2) filter. Both 3D-SBHP and AT-3D-SPIHT schemes perform lossless encoding. In our experiments, we measure only the coding time. The wavelet transform time is not included.

The lossless encoding times of AT-3D-SPIHT and 3D-SBHP on CT_Skull and MR_liver_t1 are compared in Table 9, measured in total CPU cycles used for whole image sequence and average CPU cycles used for a single pixel. Table 10 compares the decoding times of AT-3D-SPIHT and 3D-SBHP on CT_Skull and MR_liver_t1 at the rate of 0.125, 0.25, 0.5 and 1.0 bpp. The comparison shows that 3D-SBHP encoder runs around 6 times faster than AT-3D-SPIHT encoder. As bit rate increases from 0.125 bpp to full bit rate, 3D-SBHP decoder is about 6 to 10 times faster than AT-3D-SPIHT decoder. For both schemes, the decoding time is much less than encoding time. The decoding times increase around twice when the bit rate is doubled. For these two kinds of test image sequences, the average coding times used for coding a single pixel are very similar at every bit rate.

Table 11 compares the coding times of 3D-SBHP on CT_Skull and MR_liver_t1 at a variety of resolution. The results show that total encoding and decoding times increase about 6 times at the next higher resolution, while the times used for coding a single pixel decrease around 20%.


(a) A 2D visual example of 3D-SBHP random access decoding. The left: the 17th slice of CT_skull sequence at 1/2 resolution; The right: the 17th slice in the ROI decoded image sequence, full resolution.

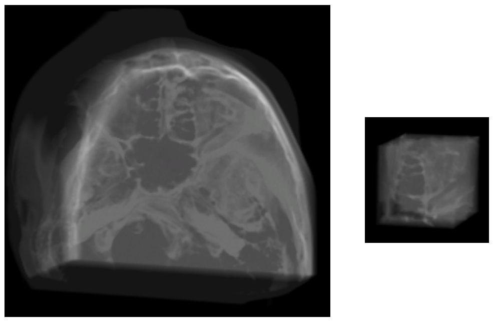

(b) A 3D visual example of 3D-SBHP random access decoding. The left: CT_skull sequence; The right: the ROI decoded image sequence.

Figure 5: An visual example of 3D-SBHP random access decoding.

File	Total Cycles ($\times 10^6$)		Cycles/pixel	
	3D-SBHP	AT-3D -SPIHT	3D-SBHP	AT-3D -SPIHT
CT_Skull	1643.162	10086.096	130.58	801.570
MR_liver_t1	449.921	2560.516	143.58	813.966

Table 9: The comparison of lossless encoding time between AT-3D-SPIHT and 3D-SBHP on image CT_skull and MR_liver_t1. (Wavelet transform times are not included.)

4. Summary and Conclusions

In this article, we present 3D-SBHP, an embedded, block based, three-dimensional wavelet transform coding algorithm of low complexity. With small loss of compression efficiency, it is able to encode an image sequence around 6 times faster than AT-3D-SPIHT. And according to the bit rate, it is able to decode a image sequence about 6 to 10 times faster than AT-3D-SPIHT. 3D-SBHP also supports resolution scalability and ROI retrievability. These features

Bit Rate	Total Cycles ($\times 10^6$)		Cycles/pixel	
	3D-SBHP	AT-3D-SPIHT	3D-SBHP	AT-3D-SPIHT
CT_Skull				
0.125	58.130	375.695	4.62	29.86
0.25	107.528	786.145	6.08	62.477
0.5	199.141	1677.159	15.82	133.29
1.0	378.820	3689.307	30.11	293.20
lossless	814.119	8333.717	64.70	662.30
MR_liver_t1				
0.125	14.451	96.860	4.59	30.79
0.25	27.797	174.739	8.837	55.55
0.5	51.634	396.864	16.41	126.16
1.0	97.215	844.629	30.904	268.50
lossless	231.21	2142.805	73.50	681.18

Table 10: The comparison of decoding time between AT-3D-SPIHT and 3D-SBHP on image CT_skull and MR_liver_t1 at a variety of bit rates. (Wavelet transform times are not included.)

Resolution	Encoding		Decoding	
	Total Cycles ($\times 10^6$)	Cycles /pixel	Total Cycles ($\times 10^6$)	Cycles /pixel
CT_Skull				
1/4	41.614	211.66	18.638	94.797
1/2	255.458	162.41	113.901	72.416
Full	1643.162	130.58	814.119	64.70
MR_liver_t1				
1/4	10.605	215.76	6.903	140.44
1/2	73.128	185.97	38.106	96.91
Full	449.921	143.58	231.21	73.50

Table 11: Coding time of 3D-SBHP on CT_skull and MR_liver_t1 at a variety of resolutions

make the proposed algorithm a good candidate for compression of 3D image data sets for multimedia applications.

Acknowledgments

We gratefully acknowledge the support of the Office of Naval Research under Grant No. N00014-05-1-0507.

References

[1] A. Bilgin, G.Zweig, and M.W. Marcllin, "Three-dimensional image compression with integer wavelet transform", *Applied Optics*, Vol. 39, No.11, April. 2000.

[2] B.Kim and W.A.Pearlman, "An embedded wavelet video coder using three-dimensional set partitioning in hierarchical tree", *IEEE Data Compression Conference*, pp. 251-260, March 1997.

[3] C. Chysafis, A. Said, A. Drukarev, A. Islam, and W.A Pearlman, "SBHP - A Low complexity wavelet coder", *IEEE Int. Conf. Acoust., Speech and Sig. Proc. (ICASSP2000)*, vol. 4, PP. 2035-2038, June 2000.

[4] A. Islam and W.A. Pearlman, "An embedded and efficient low-complexity hierarchical image coder", *in Proc. SPIE Visual Comm. and Image Processing*, Vol. 3653, pp. 294-305, 1999.

[5] J.M. Shapiro, "Embedded image coding using zerotrees of wavelet coefficients", *IEEE Trans. Image Processing*, Vol. 41, pp. 3445-3462, Dec. 1993.

[6] A. Said and W.A. Pearlman, "A new, fast and efficient image codec based on set-partitioning in hierarchical trees", *IEEE Trans. on Circuits and Systems for Video Technology*, Vol. 6, pp. 243-250, June 1996.

[7] P.N.Topiwala, "Wavelet Image and video compression", *Kluver Academic Publishers*, 1998.

[8] X. Tang, W.A. Pearlman and J.W. Modestino, "HyPerspectral image compression using three-dimensional wavelet coding", *SPIE/IS&T Electronic Imaging 2003, Proceedings of SPIE*, Vol. 5022, Jan. 2003.

[9] Yonghui Wang, Justin T. Rucker, and James E. Fowler, "Three-Dimensional Tarp coding for the compression of hyperspectral images", *IEEE Geoscience and Remote Sensing Letters*, Vol. 2, pp. 136-140, April 2004.

[10] D. Taubman, "High performance scalable image compression with EBCOT", *IEEE Trans. on Image Processing*, Vol. 9, pp. 1158-1170, July 2000.

[11] S. Cho, D. Kim, and W. A. Pearlman, "Lossless compression of volumetric medical images with improved 3-D SPIHT algorithm", *Journal of Digital Imaging*, Vol. 17, No. 1, pp. 57-63, March 2004.

[12] Kakadu JPEG2000 v3.4, http://www.kakadusoftware.com/.