Region of Interest Access with Three-Dimensional SBHP Algorithm

CIPR Technical Report TR-2006-1

Ying Liu and William A. Pearlman

January 2006

Center for
Image Processing Research

Rensselaer Polytechnic Institute
Troy, New York 12180-3590
http://www.cipr.rpi.edu

Region of Interest Access with Three-Dimensional SBHP Algorithm

Ying Liu and William A. Pearlman

Center for Image Processing Research,
Electrical Computer and Systems Engineering Department,
Rensselaer Polytechnic Institute, Troy, NY

ABSTRACT

One interesting feature of image compression is support of region of interest (ROI) access, in which an image sequence can be encoded only once and then the decoder can directly extract a subset of the bitstream to reconstruct a chosen ROI of required quality. In this paper, we apply Three-dimensional Subband Block Hierarchical Partitioning (3-D SBHP), a highly scalable wavelet transform based algorithm, for volumetric medical image compression to support ROI access. The code-block selection method by which random access decoding can be achieved is outlined and the performance empirically investigated. The experimental results show that there are a number of parameters that affect the effectiveness of ROI access, the most important being the size of the ROI size, code-block size, wavelet composition level, number of filter taps and target bit rate. Finally, one possible way to optimize ROI access performance is addressed.

Keywords: Volume image compression, 3D wavelet compression, random access decoding, ROI retrievability

1. INTRODUCTION

Three dimensional data sets, such as medical volumetric data generated by computer tomography (CT) or magnetic resonance (MR), typically contain many image slices that require huge amounts of storage. For modern multimedia applications, particularly in the Internet environment, efficient compression techniques are necessary to reduce storage and transmission bandwidth. 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 that can greatly save decoding time and transmission bandwidth.

Although volumetric images can be compressed by applying a two-dimensional compression algorithm to each slice independently, the high correlation between slices makes a three-dimension-based algorithm a better choice. Recently, many volumetric image compression algorithms based on the wavelet transform were proposed. Most of them, 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] do not naturally provide random access functionality.

SBHP[3] 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. Here, in this paper, we extend SBHP to three dimensions in such a way as to provide all the above progressive functions.

JPEG2000 can achieve three ROI coding methods: tiling, coefficient scaling and code-block selection.[7] Since code-block selection doesn't require the ROI be determined and segmented before encoding, the image sequence can be encoded only once and it's up to the decoder to extract a subset of bit stream to reconstruct an image region specified in spatial location and quality. This gives user the flexibility at decoding time, which is vital to some applications, like client/server application. In this paper, we investigate ROI access with code-block selection in detail.

The impact that the code-block size has on the compression efficiency and accessibility of a scalable codestream is assessed in.[8] However, code-block size is not the only factor which affects the effectiveness of ROI access. In this paper, we also investigate other important factors, such as wavelet compositon level, number of filter tap and decoding order.

The rest of this paper is organized as follows. We present the 3D-SBHP algorithm and the code-block selection ROI access scheme in Section 2, followed by experimental results in Section 3. Section 4 concludes this study.

Further author information: (Send correspondence to W.A.P.)
W.A.P.: E-mail: pearlw@ecse.rpi.edu, Telephone: 1 518 276 6082, Fax: 1 518 276 8715

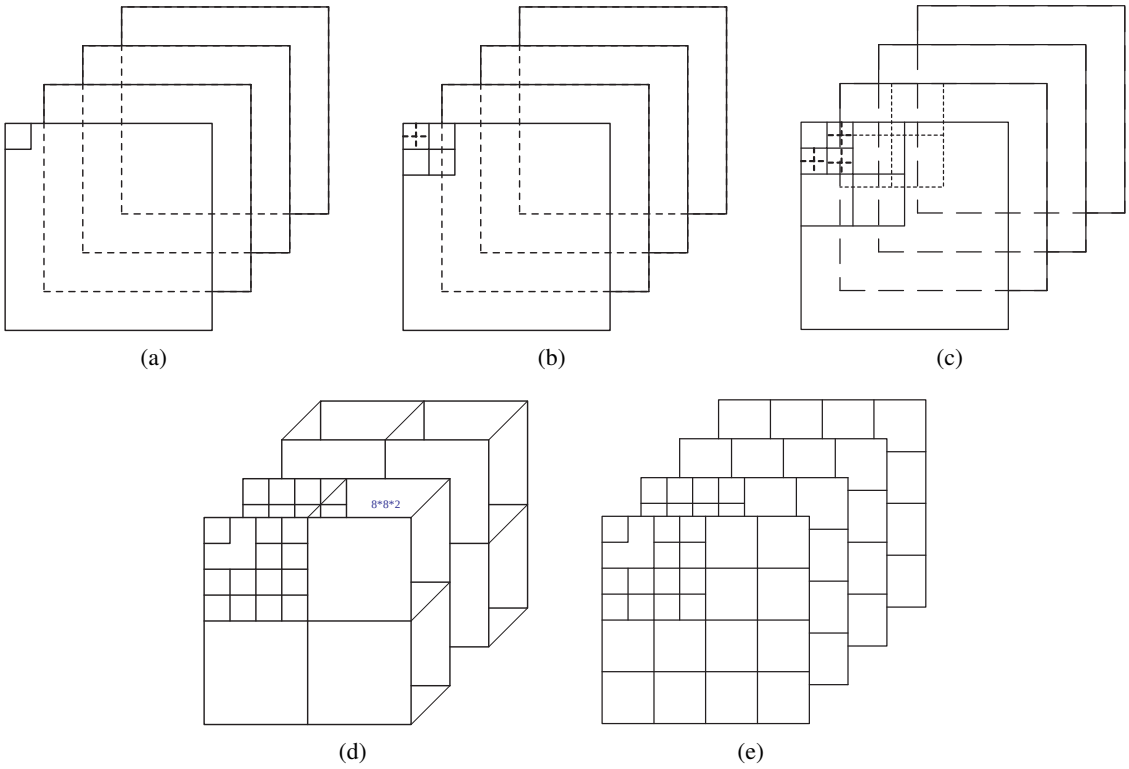


Figure 1. Set partitioning rules used by 3-D SBHP.

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]

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, 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:

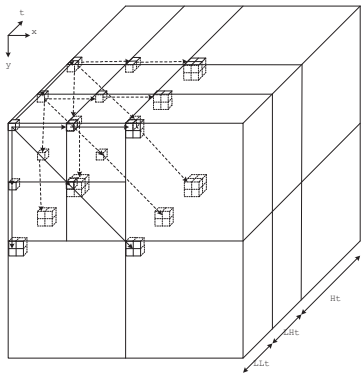


Figure 2. Parent-offspring dependencies in the 3D orientation tree.

- 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.

For each new bit plane, significance of coefficients in the LIP are tested first, then the sets in the LIS, and lastly the code refinement bits for coefficients in LSP. In order to achieve minimal complexity, 3-D SBHP does not use any entropy coding to code the sign and the refinement bits.

2.2. 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 the 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-block are of a fixed size, they represent an increasing spatial extent at lower frequency subband. Figure 2 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.

In this paper, we consider retrieveing 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 $[x_A, x_B)$ denotes the range of the cubic region in the X direction. Let $[x^F_{k,l}, x^R_{k,l})$ denotes 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 don't consider the filter length, the boundaires 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[6] 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 level K is given by

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

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 boundaires 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.

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

where,

$$t = \begin{cases} 1, i < T \\ 0, i = T \end{cases} \quad S = \begin{cases} 1, j < K \\ 0, j = K \end{cases} \quad s = \begin{cases} 3, S = 1 \\ 1, S = 0 \end{cases} \quad (2)$$

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.

2.3. Filter Implementation

In experiment, we use Haar, I(2,2), I(4,2) and I(4,4) filters for constructing wavelet transforms to map integers to integers. The equations of the filters are given below.

$$Haar \begin{cases} h_m = c_{2m+1} - c_{2m} \\ l_m = c_{2m} + \lfloor \frac{1}{2} h_m \rfloor \end{cases} \tag{3}$$

$$I(2,2) \begin{cases} h_m = c_{2m+1} - \lfloor \frac{1}{2} (c_{2m} + c_{2m+2}) + \frac{1}{2} \rfloor, \\ l_m = c_{2m} + \lfloor \frac{1}{4} (h_{m-1} + h_m) + \frac{1}{2} \rfloor \end{cases} \tag{4}$$

$$I(4,2) \begin{cases} h_m = c_{2m+1} - \lfloor \frac{9}{16} (c_{2m} + c_{2m+2}) - \\ \frac{1}{16} (c_{2m-2} + c_{2m+4}) + \frac{1}{2} \rfloor, \\ l_m = c_{2m} + \lfloor \frac{1}{4} (h_{m-1} + h_m) + \frac{1}{2} \rfloor \end{cases} \tag{5}$$

$$I(4,4) \begin{cases} h_m = c_{2m+1} - \lfloor \frac{9}{16} (c_{2m} + c_{2m+2}) - \\ \frac{1}{16} (c_{2m-2} + c_{2m+4}) + \frac{1}{2} \rfloor, \\ l_m = c_{2m} + \lfloor \frac{9}{32} (h_{m-1} + h_m) - \\ \frac{1}{32} (h_{m-2} + h_{m+1}) + \frac{1}{2} \rfloor \end{cases} \tag{6}$$

3. NUMERICAL RESULTS

We test 3-D SBHP and investigate the effect of a number of parameters on ROI access on a set of eight 8-bit medical image sequences. Table 1 shows the description of these sequence. In our experiments, all image sequences are compressed losslessly with $GOS = 16$. In Section 3.3, we compare the ROI decoding performance by use of different wavelet decomposition levels and four different wavelet filters given in Section 2.3. For other experiments, we use integer filter I(2,2) with three levels of spatial dyadic wavelet transform and two levels temporal wavelet transform.

The coding performance is measured by peak signal to noise ratio (PSNR) over the whole image sequence. When the image to be decoded contains an ROI, PSNR is calculated for the ROI.

Table 1. Description of the image volumes

File Name	Image Type	Volume Size
Skull	CT	$256 \times 256 \times 192$
Wrist	CT	$256 \times 256 \times 176$
Carotid	CT	$256 \times 256 \times 64$
Aperts	CT	$256 \times 256 \times 96$
Liver_t1	MR	$256 \times 256 \times 48$
Liver_t2e1	MR	$256 \times 256 \times 48$
Sag_head	MR	$256 \times 256 \times 48$
Ped_chest	MR	$256 \times 256 \times 64$

3.1. Lossy-to-lossless coding performance by use of different code-block sizes

We first compare the lossy-to-lossless coding performance by using different coding units. Figure 3 illustrates the effect of increasing the code-block size on the rate-distortion performance. Average PSNR is calculated over all eight image sequences. The results show there is only subtle decrease in rate-distortion performance when the code-block size reduces from $64 \times 64 \times 4$ to $32 \times 32 \times 2$. The significant loss is observed when the code-block size is less than $16 \times 16 \times 2$. This performance decrease is particularly significant at bit rates < 0.5 bpp where there is as much as 10 dB decrease in PSNR for $16 \times 16 \times 2$ code-block size and 20 dB decrease in PSNR for $8 \times 8 \times 2$ code-block size compared to the other three larger code-block sizes, for a given bit rate. The reason for the reduction of coding efficiency is that smaller code-block size increases the total overhead for the whole image sequence. As shown in the figure, for $8 \times 8 \times 2$ code-block size, when bit rate increases from 0.03125 bpp to 0.125 bpp, there is no PSNR increase. That means all decoded bits are overhead bits.

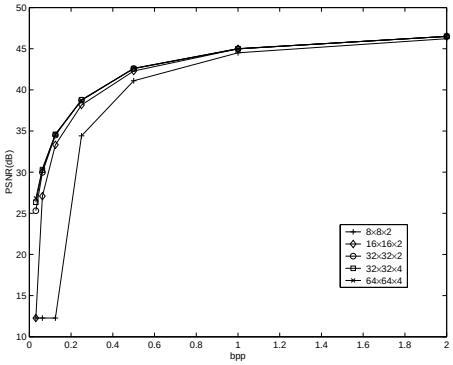


Figure 3. Rate-distortion performance with increasing code-block size.

3.2. ROI decoding performance by use of different code-block sizes and ROI sizes

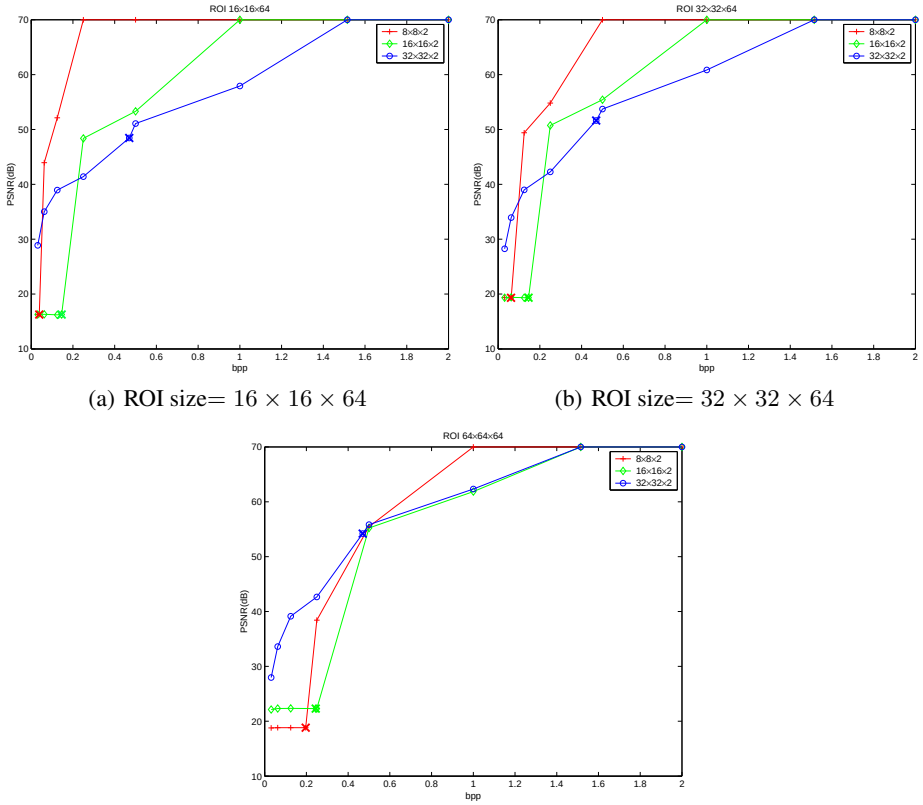
In Figure 4, we show the inter-dependence between ROI size and code-block size. The experiment is performed on the CT_Carotid sequence and the rectangular region with the lower left corner in the center of image slice and sizes of $16 \times 16 \times 64$, $32 \times 32 \times 64$ and $64 \times 64 \times 64$ are selected as ROI. For every ROI size, the figure gives performance of using $8 \times 8 \times 2$,$16 \times 16 \times 2$ and $32 \times 32 \times 2$ code-blocks. All nonfilter-length related ROI code-blocks are decoded before filter-length related code-blocks. It can be seen that code-block size $8 \times 8 \times 2$ gives the best lossless decoding performance in all three ROI size experiments. And enlarging the size of the ROI increases the bit rate at which the ROI is losslessly decoded only when the ROI size is larger or equal to the image region to which a code-block in the highest subband is correlated. In the figure, the rate-distortion at the point when all nonfilter-length related ROI code-blocks are fully decoded is labelled by black $\times$ on every curve. It is clear to observe that when filter-length related code-blocks are decoded, for a given code-block size, smaller ROI curves have higher slope, and for a given ROI size, those curves with smaller code-block size have higher slope. The higher slope indicates that the filter-length related code-blocks are more important than those in the lower slope case. It also indicates that in smaller code-block cases, higher percentage of bits decoded in the filter-length related code-blocks are used for perfect reconstruction of the ROI, while in the larger code-block case, more bits decoded in the filter-length related code-blocks contribute to background. However, the larger code-block size gives better rate-distortion performance at low bit rate. This occurs because of the overhead of a code-block. Therefore, in applications where only a high quality ROI is required or ROI size is small, smaller code-blocks (say $8 \times 8 \times 2$) should be used. Whereas if the desired bit rate is very low, or the background is also of some importance, larger code-blocks should be used.

3.3. ROI decoding performance by use of different wavelet filters and wavelet decomposition levels

Tables 2 and 3 show the effect of different filter lengths and wavelet decomposition levels on spatial direction to the ROI decoding performance. A two-level temporal wavelet transform with I(2,2) filter is used in all cases. Experiments are performed on 64 slices of the CT_Carotid image sequence with $8 \times 8 \times 2$ code-blocks and $64 \times 64 \times 64$ ROI used in the last section. The distortions are compared after all nonfilter-length related ROI code-blocks are decoded. We see that decreasing the synthesis filter length offers slightly better distortion when number of taps is larger than two. As shown in 5(b), only the 2-tap Haar filter allows perfect separation background from ROI, although it's compression performance is inferior to longer filters. For other longer filters, the quality of ROI suffers much from error penetration, shown in 5(c)-5(e). Table 3 gives the performance when spatial wavelet decomposition level is 2. It provides more than 4dB better performance for all filters than 3 level decomposition. Again, we see that fewer decomposition levels offer better ROI quality at the price of decreased compression capability. As shown in Equation 1, the number of error penetrated pixels grows exponentially in the number of decomposition level and is proportional to the synthesis filter length. To improve ROI quality, reducing the number of decomposition level and filter length is preferred. However, if the compression efficiency and quality of the background are also of some importance, a balance has to be established.

3.4. ROI access performance by use of different decoding priorities

In Figure 4(b), when $8 \times 8 \times 2$ code-block is used, all nonfilter-length related ROI code-blocks are fully decoded at rate 0.062 bpp with low $PSNR = 19.37dB$. And when the bit rate increases from 0.0315 bpp to 0.062 bpp, there is no significant



(a) ROI size= $16 \times 16 \times 64$ (b) ROI size= $32 \times 32 \times 64$

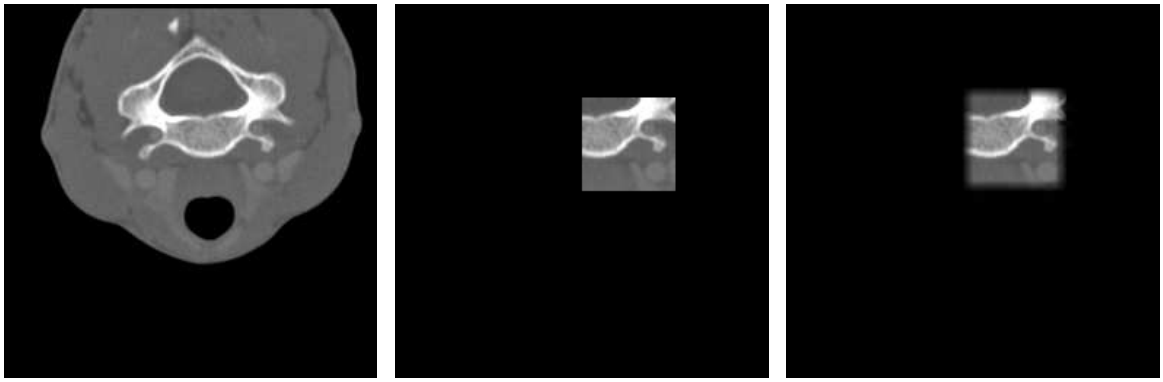
(c) ROI size= $64 \times 64 \times 64$

Figure 4. Rate-distortion performance with increasing ROI size.

Table 2. Comparison of different wavelet filter on ROI access and lossless encoding (ROI size = $64 \times 64 \times 64$, code-block size = $8 \times 8 \times 2$, spatial wavelet decomposition level = 3)

Filter Type	Decoding nonfilter-length related ROI code-blocks		Bit rate for lossless decoding ROI(bpp)	Bit rate for lossless compression(bpp)
	Bit rate(bpp)	PSNR(dB)		
2-tape S	0.2237	lossless	0.2237	2.7647
I(2,2)	0.1967	18.83	0.6150	2.5973
I(4,2)	0.1912	18.48	0.6018	2.5802
I(4,4)	0.1922	18.49	0.6058	2.5933

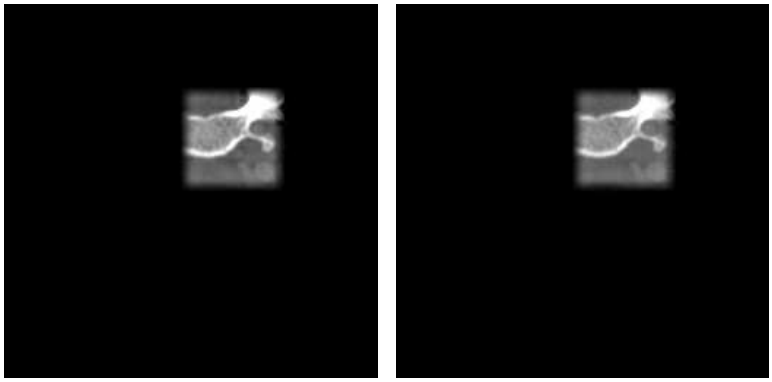
performance improvement shown in the figure, while when the filter-length related code-blocks are decoded, the quality increases sharply. That means that filter-length related code-blocks play an important role in improving the quality and low bit plane (corresponding to the least significant bit of the wavelet coefficients in the ROI) in nonfilter-length related ROI code-blocks contribute very little to the visual quality of the ROI. Therefore, it would make sense to terminate decoding nonfilter-length related ROI code-blocks before we reach the low bit planes and instead send high bit planes from the filter-length related code-blocks when the given bit rate is not enough to fully decode all ROI related code-blocks, i.e., giving high bit planes in filter-length related code-blocks higher decoding priority than low bit planes in nonfilter-length related ROI code-block. In Figure 6, we compare the rate-distortion performance of three different decoding priorities. The first, shown as *decode priority 1* in the figure, is the decoding scheme we used in the last experiment. Here we give the lowest bit plane in nonfilter-length related ROI code-blocks higher priority than the highest bit plane in filter-length related code-blocks, i.e., decoding all nonfilter-length related ROI code-blocks before decoding filter-length related code-blocks. The second,



(a) 5th slice of original CT_Carotid image sequence

(b) The ROI decoded image slice with Haar filter

(c) The ROI decoded image slice with I(2,2) filter



(d) The ROI decoded image slice with I(4,2) filter

(e) The ROI decoded image slice with I(4,4) filter

Figure 5. A visual example of ROI decoding from 3-D SBHP bit stream using different wavelet filters.

Table 3. Comparison of different wavelet filter on ROI access and lossless encoding (ROI size = $64 \times 64 \times 64$, code-block size = $8 \times 8 \times 2$, spatial wavelet decomposition level = 2)

Filter Type	Decoding nonfilter-length related ROI code-blocks		Bit rate for losslessly decoding ROI(bpp)	Bit rate for lossless compression(bpp)
	Bit rate(bpp)	PSNR(dB)		
2-tape S	0.2267	lossless	0.2267	2.7796
I(2,2)	0.1999	22.87	0.5541	2.6058
I(4,2)	0.1937	22.57	0.5397	2.5888
I(4,4)	0.1947	22.59	0.5433	2.6020

shown as *decode priority 2*, gives corresponding bit planes in those two kind of code-blocks the same priority. That means the second scheme decodes all ROI related code-blocks together from the highest bit plane to the lowest bit plane. In the third scheme, shown as *decode priority 3*, we give the fourth bit plane (from the bottom) in the nonfilter-length related ROI code-blocks the same priority as the sixth bit plane (from the bottom) in the filter-length related code-blocks. Comparing these three schemes, although they achieve lossless decoding the ROI at the same bit rate, their rate-distortion curves are significantly different. At low bit rate (< 0.1bpp), the third scheme gives at most 15dB and 5dB better performance than the first and second scheme, respectively, whereas at higher bit rate, the first scheme performs best. Therefore, given a bit rate, to get the best lossy ROI decoding performance, we need to find an optimal decoding priority according to the relative importance between filter-length related code-blocks and nonfilter-length related ROI code-blocks. As we addressed in

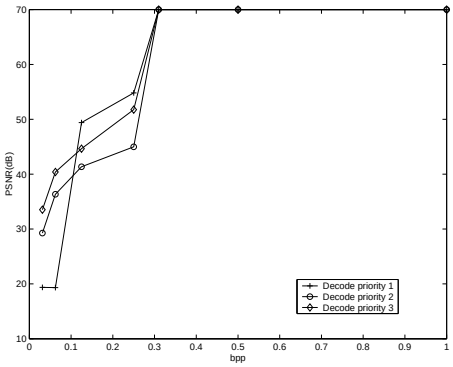


Figure 6. Rate-distortion performance with different priorities for code-blocks.

previous sections, the relative importance between those two kind of blocks depends on code-block size, ROI size, filter length and wavelet decomposition level.

4. CONCLUSIONS

In this article, we present 3D-SBHP algorithm and empirically investigated code-block selection ROI access method by applying 3D-SBHP on medical volumetrical images. Our work shows that the ROI access performance is effected by a set of coding parameters. We also outline some trade-offs in ROI access. At last, we give a possible way to optimize ROI access performance at the decoder side.

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. Chrysafis, 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. P.N. Topiwala, *Wavelet Image and video compression*, Kluver Academic Publishers, pp.254-255, 1998.
7. A.P. Bradley and F.W.M.Stentiford *JPEG 2000 and region of interest coding*, Digital Image Computing Techniques and Applications(DICTA), Melbourne, Australia, pp. 303-308, 2002.
8. R. Leung and D. Taubman, *Transform and Embedded Coding Techniques for Maximum Efficiency and Random Accessibility in 3-D Scalable Compression*, IEEE Trans. Image Processing, Vol. 14, pp. 1632-1646, Oct. 2005.
9. W. A. Pearlman, A. Islam, N. Nagaraj, and A. Said, *Efficient, Low-Complexity Image Coding with a Set-Partitioning Embedded Block Coder*, IEEE Trans. Circuits and Systems for Video Technology, Vol. 14, pp. 1219-1235, Dec. 2004.