Fast Near-Lossless or Lossless Compression of Large 3D Neuro-Anatomical Images

Rongkai Zhao[a], Michael Gabriel[b], Geneva G. Belford[a]

[a]Computer Science Department, University of Illinois at Urbana-Champaign, 3310 DCL, MC 258, 1304 W. Springfield, Urbana, IL 61801, USA
[b]Beckman Institute, University of Illinois at Urbana-Champaign, 2357 Beckman Institute, MC 251, 405 N. Mathews, Urbana, IL 61801, USA

ABSTRACT

3D neuro-anatomical images and other volumetric data sets are important in many scientific and biomedical fields. Since such sets may be extremely large, a scalable compression method is critical to store, process and transmit them. To achieve a high compression rate, most of the existing volume compression methods are lossy, which is usually unacceptable in biomedical applications. Our near-lossless or lossless compression algorithm uses a Hilbert traversal to produce a data stream from the original image. This data stream enjoys relatively slow image context change, which helps the subsequent DPCM prediction to reduce the source entropy. An extremely fast linear DPCM is used; the prediction error is further encoded using Huffman code. In order to provide efficient data access, the source image is divided into blocks and indexed by an octree data structure. The Huffman coding overhead is effectively reduced using a novel binning algorithm. Our compression method is designed for performance-critical digital brain atlas applications, which often require very fast data access without prior decompression and for which a modest compression rate is acceptable.

Keywords: volume data, 3D medical imaging, near-lossless compression, lossless compression, octree, entropy coding, predictive coding, Hilbert traversal.

1. INTRODUCTION

Image compression has been extensively studied for more than two decades. Many compression methods, lossy or lossless, were developed for 2D images.[1] Various standards have emerged. In spite of all that work, very few volume compression methods have been developed. Lossy volume compression methods have been developed in order to achieve high compression rates. Data fidelity is a critical issue in medical imaging, which often renders lossy methods unacceptable. Due to the probable loss of

Further author information: (Send correspondence to R. Zhao)
R. Zhao: E-mail: rongkai@uiuc.edu, Telephone: 1 217 351 9798
M. Gabriel: E-mail: mgabriel@uiuc.edu, Telephone: 1 217 244 3463
G. G. Belford: E-mail: gbelford@uiuc.edu, Telephone: 1 217 333 6684

information in the many steps towards preparing a 3D neuro-anatomical image, lossless compression on the other hand is too strict and sometimes overkill. Near lossless seems to be a natural compromise. Very little work has been done in near-lossless 3D neuro-anatomical image compression. Even though 3D image compression seems to be a simple extension of 2D image compression, there are many unique properties that are ignored in 2D. Scalability, scan order, overhead control, along with near-lossless quantization are discussed in this paper.

Among the handful of volume compression methods,[2–10] most are lossy.[3–5,7–10] Different methods such as 2D wavelet transformation,[9] 3D wavelet transformation,[5,8] 3D Laplacian pyramid,[7] fractal decomposition[4] are used in lossy volume compression. Integer 3D wavelet transformations are used to offer progressive volume decompression[3,10] where lossy and lossless information are built into the same data stream. However, when lossless compression is desired, the integer 3D wavelet transformation no longer offers good compression rate nor efficiency.

Generic data compression software such as GZIP, COMPRESS, PKZIP, and WinZip use dictionary-based methods[1] to compress computer files disregarding the type of the data source. When they are used to compress image data, they often fail to efficiently de-correlate the spatial redundancy within the image source. Lossless volume compression such as COMPVOX[2] compresses the volume using linear DPCM and truncated Huffman code. The entire volume must be decoded to access the volume data. The non-linear DPCM-based lossless method[6] achieves a better compression rate yet is slower than the linear DPCM.

There are two definitions of near lossless compression. One definition is based on the percentage of altered pixels/voxels. For example, less than 5% of the pixels/voxels altered in the decompressed image is considered near lossless. This definition is not very convincing because if the 5% altered pixels/voxels are the edges in the image and the intensity changes are significant, then the image can look totally different. A more reasonable and widely accepted definition concerns the visual appearance of the decompressed image. A compression method is near lossless if no pixel/voxel is changed in magnitude by more than d gray levels. Our near-lossless method is based on the second definition.

Our compression method is designed with a performance-critical digital brain atlas application in mind. Speed and compression rate are weighted equally towards a balanced solution. The remainder of this paper is organized as follows. In section 2, we describe the outline of our method. Section 3 is the index data structure. Section 4 discusses our unique voxel scan order. Near-lossless quantization and overhead control are discussed in section 5 and section 6, respectively. Section 7 shows some results and concludes the paper.

2. OUTLINE OF THE COMPRESSION METHOD

The compression method is outlined as follows. The 3D image is divided into subvolume blocks and then indexed by an octree data structure. Each block is traversed

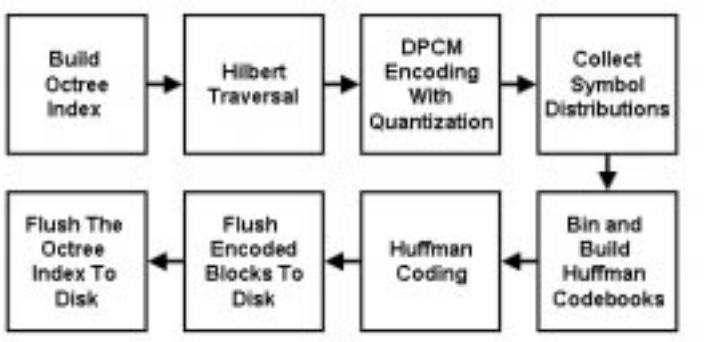


Figure 1: Data flow of the compression method.

following a Hilbert curve. While traversing the block, a near-lossless or lossless DPCM prediction is performed. The prediction error is recorded, as is the histogram of the prediction error. Each sub-volume block has its unique prediction error distribution. To form a Huffman code book of the prediction error for the entire volume is inefficient. On the other hand, to produce a Huffman code book for every sub-volume block also introduces heavy coding overhead. We characterize each block's error distribution as a point in a high-dimensional space and then bin the points using a novel binning method. All the error distributions that fall into the same bin are summed together to form a summed error distribution. We build a Huffman code book for this distribution. The total number of Huffman code books is the number of bins. The coding overhead is therefore effectively reduced. All the sub-volume blocks' prediction error is coded according to its own Huffman code book. Finally, the code books, the upper level index and encoded sub-volume blocks are flushed to disk. Figure 1 is the data stream flow chart of the compression method.

When the 3D image is accessed by digital brain atlas applications, based on the index structure, only those sub-volume blocks that are actually needed will be decoded. To decode a block, its own Huffman code book is loaded. The decompressed data stream is the prediction error, which is translated back to the voxel intensity. The last step is to translate the Hilbert scan order back to normal raster scan order. It is faster to decompress a sub-volume block than to compress it.

3. UPPER LEVEL DATA STRUCTURE

A typical digitized 3D animal brain atlas may be as big as a few hundred gigabytes. Larger 3D images will appear in the foreseeable future. It is clear that encoding or decoding the entire volume is forbidden. The volume must be divided into cubic blocks. The block size is a parameter defined by the user, or the application program may assign a default value. We use the octree data structure as the index. Sub-volume blocks that are pure background will be reduced to terminal nodes in the octree. If the volume is too big, a single octree may be inefficient, due to the linear code used

to identify the location of the octree nodes. We use the SHSF data structure[11] in extreme cases. SHSF is an octree variant that uses two levels of indirection to reduce the octree linear code overhead. Sub-dividing the volume also helps to increase the compression rate. This will be discussed in section 6.

4. HILBERT TRAVERSAL

A Hilbert curve is an open space filling curve as shown in figure 2A,B. It can be used in n-dimensional space where $n \geq 2$. The common image scan order is the raster scan. In compressing 2D images, the raster scan order is only slightly less efficient than the Hilbert scan order; therefore, the literature pays very little attention to this technique. However, in 3D images, Hilbert traversal offers a very appealing property. In a Hilbert curve, every pixel/voxel is the immediate neighbor of its subsequent pixel/voxel. There isn't any abrupt image context change along a Hilbert traversal. In addition, the speed of context change is also much slower than that of raster scan. Let's define an image context as a group of pixel/voxels that occupy a small area. In a 3D image, suppose there is a 3D image context of N^3 voxels. A raster scan will leave the context after visiting N voxels. For a Hilbert scan, in the best case, it leaves the context after visiting all the N^3 voxels.

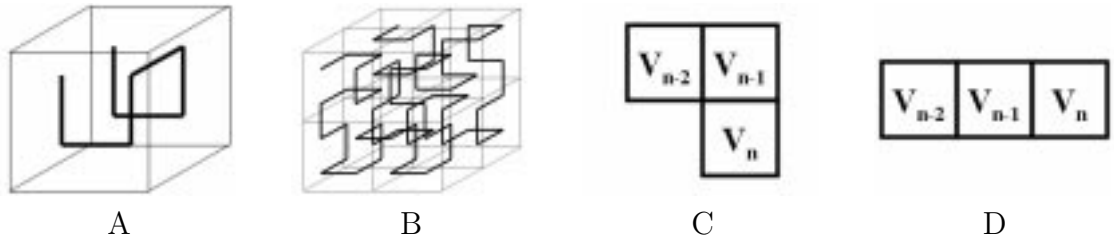


Figure 2. A) 3D Hilbert curve (Level 1); B) 3D Hilbert curve (Level 2); C) Voxels' relative locations along a Hilbert curve; D) Voxels' relative locations along a raster scan.

A DPCM prediction is used to reduce the data source's entropy by predicting the current pixel/voxel's intensity based on the previously visited pixel/voxels. The recorded prediction error can be used to reconstruct the data source and the prediction error's entropy is usually much smaller than the data source's entropy. The greater the occurrence of big prediction errors, the bigger the prediction error's entropy. Usually, a big prediction error occurs when the scan line passes through a high frequency image component such as an edge. Assume the same 3D image context defined in the previous paragraph is surrounded by a 3D image edge. Smooth regions are on both sides of the 3D edge. Using raster scan, the scan line passes through the 3D edge $2N^2$ times, yet in the best case, the Hilbert scan line pass through the 3D edge only twice. Depending on the location of the image context, the Hilbert scan line may pass through the context more than twice; but usually much less than $2N^2$ times.

Since the Hilbert scan line intersects the image edges less frequently and the image context change is also much slower than for the raster scan, a less complicated and

faster DPCM can be used. Along the Hilbert scan line, suppose the current voxel is V_n and prediction error E is:

$$E = V_n - (V_{n-1} + V_{n-2})/2 \tag{1}$$

Figure 2C,D shows the voxels' relative locations in Hilbert scan order and raster scan order. In both, the distance between V_{n-1} and V_n is 1. Along the Hilbert curve, the distance between V_{n-2} and V_n is $\sqrt{2}$. In raster order, the distance between V_{n-2} and V_n is 2. If there is a context jump, the prediction is invalid. Because V_{n-2} is closer to V_n in Hilbert scan order than in raster scan, the DPCM in Hilbert scan order tends to predict the intensity better than the same method in raster scan. This is observed through experiments.

Hilbert traversal can be easily implemented using a recursive algorithm.[12] Its running time is $O(n)$, where n is the total number of voxels.

5. NEAR-LOSSLESS QUANTIZATION

Suppose d levels of intensity change is allowed in the near-lossless quantization. Let m be an integer. The quantization function is:

$$Q(E) = \begin{cases} (2d+1)m & \text{if } E = (2d+1)m + i \text{ for } i = 1, 2, ..., d \\ (2d+1)(m+1) & \text{otherwise} \end{cases} \tag{2}$$

Two examples are illustrated in figure 3A. Inter-dependency of the prediction error requires development of a complicated prediction quantization algorithm.[13] This problem can be avoided by looking at both the predicted voxel intensity and the decoded voxel intensity. We decode the visited voxel's intensity while we encode the next voxel; therefore, we can guarantee that the quantization error will not accumulate. Let's denote V_n as the current voxel's intensity, V'_n as the decoded voxel's intensity, E as the prediction error, and E' as the quantized prediction error. Assuming $V_{-2} = V'_{-2} = V_{-1} = V'_{-1} = 0$.

$$\begin{aligned} E_0 &= V_0 - (V'_{-2} + V'_{-1})/2 \\ E'_0 &= Q(E_0) \\ V'_0 &= E'_0 + (V'_{-2} + V'_{-1})/2 \end{aligned} \tag{3}$$

for the nth voxel, we have

$$\begin{aligned} E_n &= V_n - (V'_{n-2} + V'_{n-1})/2 \\ E'_n &= Q(E_n) \\ V'_n &= E'_n + (V'_{n-2} + V'_{n-1})/2 \end{aligned} \tag{4}$$

The prediction error has a Laplace distribution.[14] The Laplace distribution's probability density function is

$$P(x) = \frac{1}{2b} e^{-|x-\mu|/b} \tag{5}$$

where μ is the population mean; in our case $\mu \approx 0$. b is the parameter that controls the shape of the distribution. For prediction error E, the probability of occurrence is

$$P(E) = \int_{E-0.5}^{E+0.5} \frac{1}{2b} e^{-|x|/b} dx = \begin{cases} \frac{e^{\frac{0.5-E}{b}} - e^{\frac{-0.5-E}{b}}}{2} & \text{if } E > 0 \\ 1 - e^{-\frac{1}{2b}} & \text{if } E = 0 \\ \frac{e^{\frac{0.5+E}{b}} - e^{\frac{E-0.5}{b}}}{2} & \text{otherwise} \end{cases} \tag{6}$$

The entropy H of the prediction error is

$$H = \sum_{E} -P(E) log_2 P(E) \tag{7}$$

When the prediction error is quantized using equation 2, the probability of quantized prediction error E_q is

$$P(E_q) = \sum_{E=E_q-d}^{E_q+d} P(E) \tag{8}$$

The entropy H_q of the quantized prediction error is

$$H_q = \sum_{E_q} -P(E_q) log_2 P(E_q) \tag{9}$$

Therefore, our near-lossless quantization method can improve the compression rate by $r\%$ where

$$r = 100 - 100 H_q / H \tag{10}$$

Figure 3B,C are the plots that demonstrate the relationship between the maximum quantization error d, quantization error distribution's b value, the distribution's entropy, and the percentage of compression improvement. Increasing the maximum quantization error d can improve the compression rate; however, it degrades the decompressed image's quality. On the other hand, increasing d does not increase the compression rate linearly. Accurate intensity prediction yields a small b value in the prediction error distribution, near-lossless quantization performs better when b is small. Hilbert traversal helps to reduce b.

6. REDUCING CODING OVERHEAD

Huffman code is a prefix code which is uniquely decipherable.[15] Huffman tree is a positional tree. A general positional tree can be a σ-ary tree.[15] For clarity, all the following discussions and proofs only consider binary case, namely $\sigma = 2$. Let's denote the symbol distribution by D_s, the Huffman tree by t, each symbol c_i's probability by P_i, and its prefix code's length by l_i. The Huffman tree is a function of the symbol distribution, namely $t = t(D_s)$. The average code length $\overline{l}$ is a function of the symbol distribution D_s and the Huffman tree t, therefore it can be represented as

$$\overline{l} = l(D_s, t(D_s)) = \sum_{i=1}^{n} P_i l_i \tag{11}$$

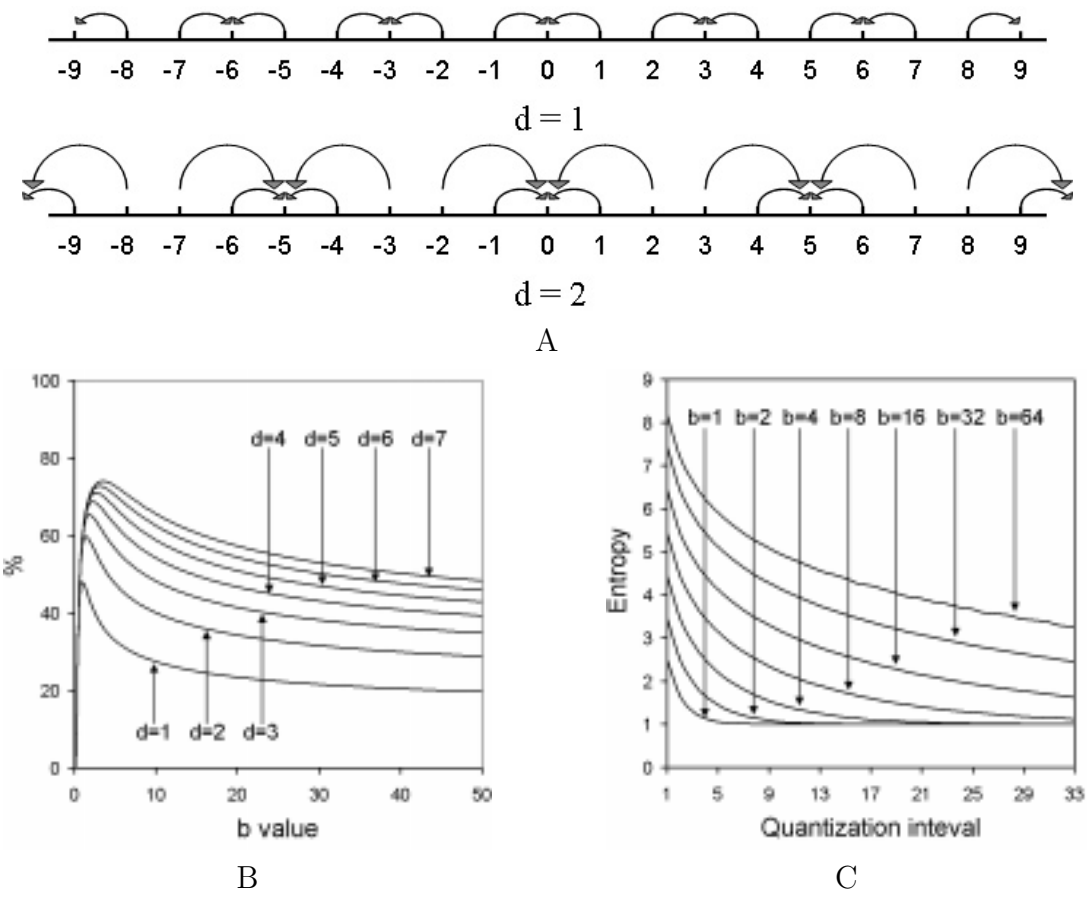


Figure 3. A) Near-lossless quantization schemes when d=1 and d=2. B) The compression rate improvement for different Laplace distributions (with different b values) where d is the maximum allowed quantization error. C) The entropy decrease when different quantization intervals are used. The quantization interval equals 2d+1.

DEFINITION 6.1. *A prefix code is optimal if its average code length* $\overline{l}$ *is minimum*

$$\overline{l} = min \tag{12}$$

DEFINITION 6.2. *A symbol distribution* D_s *has a set of symbols* $S_{Ds} = \{c_1, c_2, ..., c_n\}$, *we say* $D_s \geq D'_s$ *iff* $S_{Ds} \supseteq S_{Ds'}$.

LEMMA 6.3. *A Huffman tree* t *built from a symbol distribution* D_s *can generate an optimum prefix code for* D_s.

$$\overline{l} = l(D_s, t(D_s)) = min \tag{13}$$

In other word, $\forall t' = t(D'_s)$ *where* $D'_s \geq D_s$. *If* D_s *is encoded using the prefix code generated from* t' *then*

$$\overline{l'} = l(D_s, t(D'_s)) \geq \overline{l} \tag{14}$$

Lemma 6.3 and its proof can be found on page 74 through 79 of *Graph Algorithms*.[15] Arbitrary decisions made in constructing the Huffman tree affect the individual codes but not the average size of the codes.[1]

Assuming the Huffman coding overhead is ignorable, it is easy to prove that subdividing the volume into multiple blocks, building one Huffman tree for each block, and compressing the volume on a block basis can yield a better compression rate than treating the volume as a whole. Denote the size of the Huffman coded block by s and the block by b. The number of voxels in s is n_v. Let s be a function of b and a Huffman tree t, $s = s(b,t) = \overline{l}n_v$. Given a volume b, we can partition it into n small blocks $b_1...b_n$ where $b = \sum_{i=1}^{n} b_i$ and for each block there is a corresponding Huffman tree $t_1...t_n$.

LEMMA 6.4. *For any volume b, if we subdivide it into n smaller blocks, then*

$$s(b,t) \geq \sum_{i=1}^{n} s(b_i, t_i) \tag{15}$$

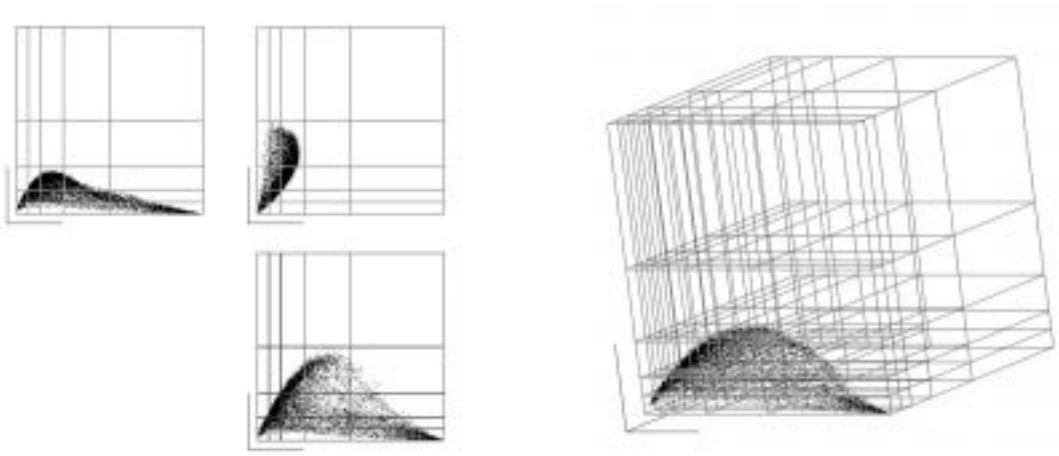
Proof: According to Lemma 6.3, the Huffman tree t_a is a binary positional tree which generates optimum prefix code for a symbol distribution D_s. The Huffman tree t'_a for a different symbol distribution D'_s ($D'_s \geq D_s$) may not be optimal for D_s. Given the Huffman tree t for volume b, for every Huffman tree t_i and corresponding b_i, $s(b_i, t_i) \leq s(b_i, t)$, therefore $s(b,t) = \sum_{i=1}^{n} s(b_i,t) \geq \sum_{i=1}^{n} s(b_i,t_i)$. QED.

LEMMA 6.5. *Disregarding the Huffman code book size, the best compression rate is obtained when the volume is divided into voxels.*

We can prove Lemma 6.5 by recursively applying Lemma 6.4 to the volume. In real life, the size of the Huffman code book cannot be ignored. Huffman code book is stored as a serialized Huffman tree. It can be easily obtained by a tree traversal. The size of a Huffman code book is $S_{codebook} = C \times n_{symbols}$ where C is a constant. Depending on the implementation, C can be slightly different. When the subdivision is too fine, the Huffman coding overhead is going to exceed the original volume size because each block has its own Huffman code book attached to the block's encoded data stream. It is therefore important to choose a suitable size for the subdivided volume blocks. From the discussion above, it is obvious that if we can efficiently avoid the coding overhead, a better compression rate can be achieved.

Building one Huffman tree for each block jeopardizes the compression improvement by sub-dividing the volume. The individual Huffman trees are very similar in terms of the code word length. Therefore, dividing the sub-volume blocks into a few groups based on their prediction error's distribution and building one Huffman tree for each block can reduce the coding overhead and also provide the benefit of using small volume blocks. The prediction error distribution can be characterized by its most probable prediction errors, such as $0, \pm(2d+1), \pm(4d+2)$. We build a three-dimensional space and use the probability of error 0 to label dimension 1 and use the sum of the probabilities of error $\pm(2d+1)$, and the sum of the probabilities of error $\pm(4d+2)$ to label dimension 2 and dimension 3. For each dimension, we use

a logarithmic scale to divide the space thus formed into a grid of rectangular units. Each unit is considered a bin. Every volume block is therefore abstracted as a point in three-space. All the blocks that fall in the same bin are considered to have a similar error distribution. We recalculate the error distribution and build a Huffman tree for each bin. Figure 4 is a visualization of our binning method. The optimal volume block size varies depending on the particular data set. We found through experiments that side length of 32 is a good choice for the block size.



A B

Figure 4. Visualization of the binning method: (A)The three projected views of the three-dimensional binning space. (B)The three-dimensional view of the binning space.

7. RESULTS & CONCLUSION

Table 1. Comparison of different compression approaches. The compression performance (bits per voxel) is obtained by compressing a rabbit brain volume data set.

	GZIP	COMPVOX	SPIHT	PNG image series	ours
bits per voxel	3.02	3.51	3.14	4.45	2.18
dynamic access	No	No	No	No	Yes
parallellism	No	No	No	No	Yes
fast previewing	No	No	Yes	No	Yes

Table 1 shows the comparison between our near-lossless compression method versus other approaches. Our near-lossless compression method uses an octree to index the sub-volume blocks. To encode individual blocks, a Hilbert traversal is followed by linear DPCM and near-lossless quantization. The quantized prediction error is further encoded using Huffman code. The Huffman coding overhead is reduced by

comparing the prediction error distribution using our binning method. Our approach has a better compression rate than the existing lossless volume compression methods; it is also fast and scalable. The compressed 3D images can be accessed dynamically without prior decompression of the entire volume. It is also possible to handle a very large (gigabytes) data set.

ACKNOWLEDGMENTS

This work was partially supported by NSF DBI-9870821.

REFERENCES

1. D. Salomon, *Data Compression*, Springer, New York, 2000.
2. J. Fowler and R. Yagel, "Lossless compression of volume data," in *1994 Symposium on Volume Visualization*, A. Kaufman and W. Krueger, eds., pp. 43–50, 1994.
3. A. Bilgin, G. Zweig, and M. Marcellin, "Three-dimensional image compression with integer wavelet transforms," *Applied Optics* **39**, pp. 1799–1814, 2000.
4. W. O. Cochran, J. C. Hart, and P. J. Flynn, "Fractal volume compression," *IEEE Transactions on Visualization and Computer Graphics* **2**, pp. 313–322, December 1996.
5. T. Chiueh, C. Yang, T. He, H. Pfister, and A. Kaufman, "Integrated volume compression and visualization," in *IEEE Visualization '97*, R. Yagel and H. Hagen, eds., pp. 329–336, 1997.
6. R. Zhao, T. Tao, M. Gabriel, and G. Belford, "Lossless compression of very large volume data with fast dynamic access," in *Electronic Imaging and Multimedia Technology III*, **4925**, pp. 179–190, SPIE, Oct 2002.
7. M. Ghavamnia and X. Yang, "Direct rendering of laplacian pyramid compressed volume data," in *Proceedings of Visualization '95*, pp. 192–199, IEEE, 1995.
8. I. Ihm and S. Park, "Wavelet-based 3D compression scheme for very large volume data," in *Graphics Interface*, pp. 107–116, 1998.
9. F. F. Rodler, "Wavelet based 3D compression with fast random access for very large volume data," in *Proceedings of the The Seventh Pacific Conference on Computer Graphics and Applications*, IEEE, 1999.
10. A. Said and W. A. Pearlman, "A new fast and efficient image codec based on set partitioning in hierarchical trees," *IEEE Transactions on Circuits and Systems for Video Technology* **6**, pp. 243–250, 1996.
11. R. Zhao, T. Tao, M. Gabriel, and G. Belford, "Brain slicer: A high performance internet-based neuro-medical imaging system," in *Visualization and Data Analysis 2002*, pp. 327–338, SPIE, 2002.
12. J. K. Lawder, "Using state diagrams for hilbert curve mappings," Tech. Rep. JL2/00, School of Computer Science and Information Systems, Birkbeck College, University of London, 2000.
13. L. Ke and M. W. Marcellin, "Near-lossless image compression: Minimum-entropy, constrained-error," in *Proceedings of the 1995 International Conference on Image Processing*, IEEE, ed., pp. 298–301, 1995.
14. H. Stark and J. W. Woods, *Probability and Random Processes with Applications to Signal Processing*, Prentice Hall, third ed., 2002.
15. S. Even, *Graph Algorithms*, Computer Science Press, Rockville, Maryland, 1979.