Trends of Tree-Based,Set-Partitioning Compression Techniques in Still and Moving Image Systems

William A. Pearlman
Center for Next Generation Video
Electrical Computer and System Engineering Department
Rensselaer Polytechnic Institute
Troy NY 12180-3590
pearlw@rpi.edu

ABSTRACT

In addition to high compression efficiency, future still and moving image coding systems will require many other features. They include fidelity and resolution scalability, region of interest enhancement, random access decoding, resilience to errors due to channel noise or packet loss, fast encoding and/or decoding speed, and low computational and hardware complexity. Moreover, there are emerging new venues for the application of image compression techniques to data associated with points of a two or three-dimensional grid that will demand these features and perhaps others. We shall discuss these new venues along with the usual ones and show how tree-based, set-partitioning wavelet coding methods, such as SPIHT (Set Partitioning in Hierarchical Trees) and SPECK (Set Partitioning Embedded bloCK) will fulfill most of the demands of current and future applications. We shall also discuss the emerging JPEG-2000 in this framework.

1 INTRODUCTION

Compression of images saves storage capacity, channel bandwidth, and transmission time. Its use is ubiquitous, as almost every digital still or moving image has been compressed before it reaches the user. For example JPEG and GIF, for still images, and MPEG-1, MPEG-2, DVD, H.263 and more for video are all compressed formats. The demands for compression are bound to increase, as production of images and the need to transmit increasingly large files continue to grow. For example, in an article appearing in the January 2001 issue of the magazine, *Advanced Imaging*, it was estimated that 80 billion (80×10^9) new images were produced in the year 2000 alone.

In order to obtain the required degree of compression, lossy coding techniques have to be utilized. The current JPEG still and MPEG moving image compression standards are based on encoding the discrete cosine transform (DCT) of 8x8 blocks independently. As such, the easily observed artifacts of blocking appear at low to medium bit rates (high to moderate compression ratios). Features needed in current applications, such as rate and resolution scalability and others to be discussed, are not naturally produced and can only be delivered with limited functionality and flexibility with accompanying loss of compression performance. Hence there is underway a migration to wavelet transform coding, in which blocking artifacts are eliminated, low to medium rate performance is superior, and needed additional features are inherent and delivered with no loss of compression performance.

2 FEATURE REQUIREMENTS

Our wish list for the features delivered by the bit streams in compression systems include the following items:

- low mean squared error (MSE) for a given bit rate
- rate (fidelity) scalability
- resolution (spatial, temporal) scalability
- random access decoding
- enhanced region-of-interest (ROI) encoding
- idempotency - perfect reversibility of decompression from compressed bitstream to reconstruction
- robustness/resillience to channel errors and packet loss
- low memory usage
- fast, simple encoding and decoding

The first item above is obvious, that of achieving the lowest possible MSE or highest possible PSNR (peak signal-to-noise ratio) for a given bit rate. The modern wavelet transform-based coding methods almost always do better in this regard than DCT-based methods, especially as the bit rate decreases. However, the next two items of scalability in rate and spatial/temporal resolution can only be achieved in a limited way for the DCT-based methods of the standards, while the wavelet-based methods can provide fine-grain rate scalability and resolution scalability in several octave steps. The rate scalability derives from bit plane coding of the wavelet coefficients and the resolution scalability from the decomposition by scale of the wavelet transform. Enabling bit plane coding of DCT coefficients would be one ingredient to accomplish fine grain rate scalability [16], but it is not allowed in the current coding standards.

Random access decoding is the ability to select for decoding portions of the bit stream corresponding to a given region of the image. Since JPEG encodes 8x8 image blocks independently, it is easy to select the portions of the bit stream belonging to a region that is comprised of a union of 8x8 blocks. It is harder to do in principle for a wavelet coder, but certainly quite feasible if one can identify the bits of wavelet coefficients belonging to a given region. Here, there is inherently more flexibility in defining shapes of regions for random access decoding.

Enhanced region of interest (ROI) encoding can be accomplished by identifying a given region at encoding time and assigning more bits per pixel to this region than to the remainder of the image. Baseline JPEG does not allow the flexibility of assigning a different quantization parameter to diffent image block, but the JPEG enhancements do. The wavelet coding methods can integrate ROI enhancement very naturally into the encoding process.

Idempotency is a feature often overlooked. Often an image will undergo many cycles of compression and decompression. Idempotency means that the image will not degrade in these cycles. So if one decompresses a bitstream by a given method and then compresses the reconstruction by the same method and rate that produced the original compressed bitstream, the resulting compressed bitstream will be identical to the original bitstream when the method is idempotent. The quantization in JPEG, for example, is an idempotent method.

Compressed bit streams tend to be fragile, since channel bit errors in variable length codes cause the decoder to lose synchronization and propagate decoding errors to the end of the bit stream. Video compounds this sensitivity due to the feedback of motion compensation and interframe prediction errors that will also propagate. One way to create a less error-sensitive bit stream is to encode the source in many independent units, so that an error in one unit will not affect others.

A method's memory usage is an important issue, especially for large images and implementation in small devices, such as cameras. The most common method is to partition the image into stripes or tiles and encode these partitions independently. JPEG and MPEG encode 8x8 blocks independently, so their coding algorithms do use a small memory space. The disadvantage of this method is that the partition boundaries become noticeable, as bit rate drops below some value, so these methods are not suitable for low rate applications. A wavelet transform requires a full image transform, but the full image is not required in memory since the filters are finite in length. For computing a transform coefficient one needs only as many rows or columns in memory to cover the extent of the filters for each level of decomposition. There are such memory-saving transforms based on putting into memory at any instant a minimum numbers of image lines or a minimum numbers of rows and colums in a block. They are called *line-based* [4] and *block-based* [2, 1] transforms, respectively. They yield the same coefficients as a full image transform, at the expense of some extra processing.

Although hardware processing speeds continue to accelerate, the demands for greater speeds for image commpression and decompression seem to outpace the accelerating processing speeds of current hardware. Greater resolution of source images and the need for rapid transmission and retrieval are a mark of our impatient and frenzied world. A fast, low complexity decoder has always been a crucial requirement, but it becomes increasingly difficult with state-of-the-art efficiency for large images. Now there is a call for real-time encoding for cameras and digital cinema. That requires fast and simple compression, which again is difficult to attain with state-of-the-art efficiency for large images. We shall concentrate our attention on low-complexity compression techniques that meet these requirements and possess the features itemized above.

3 EMBEDDED CODING ALGORITHMS

An algorithm is said to provide an embedded bit stream if one can extract a smaller file with a given characteristic that would have the same characteristic if produced directly by the same algorithm. For example, a file from a rate-embedded algorithm contains all smaller rate files with the same bits as would have been produced by the algorithm directly. Pure rate-embedded algorithms have granularity at the bit level. A rate scalable algorithm refers to similar properties, regardless of the granuarity of the embedding. A rate embedded coder allows progessive encoding from lossy to purely lossless reconstruction, when the filters map integers from the image domain to integers in the transform domain. In order to realize embedded coding, one requires either the full transform or the full compressed bit stream in

memory. Clearly this is unfeasible for large images or long image sequences. We shall now describe some methods of embedded image coding.

3.1 JPEG-2000 Embedded Coding

The emerging JPEG-2000 still image compression standard uses a memory-saving line-based wavelet transform and bit plane entropy coding of subblocks of wavelet subbands. The method of entropy coding of the subblocks is called "Embedded Block Coding with Optimized Truncation (EBCOT)" [14]. Prior to entropy coding, the image is transformed into subbands by a wavelet (or wavelet packet) decomposition. The subbands are then subdivided into as many subblocks, nominally of dimensions 64x64 or 32x32, as possible. Generally there will be partial-size subblocks for subband dimensions not divisible by 32 or 64 or smaller than 32 or 64. They will be treated similarly to the square subblocks with the appropriate modifications. Entropy coding progresses from lowest to highest scale among subbands and in raster order within subbands. Every subblock is encoded independently via context-based bit plane arithmetic coding. The wavelet coefficients are first finely quantized to a sign and magnitude representation of the bin indices. Encoding of the binary expansion of the bin index magnitudes starts from the highest order non-all-zero bit plane in each subblock and proceeds in order through the lower bit planes until the target bit budget is reached for the subblock. Three passes through each bit plane are made to gather contexts of bits in three categories for adaptive arithmetic coding. There is also passage through the sign plane for context-based adaptive arithmetic coding prior to the magnitude passes. The decoder must be informed of the number of leading all-zero bit planes for each subblock. The array of such numbers, one for each subblock, is encoded by a quadtree technique and sent as overhead information.

In order to encode the image to a given target bit rate, the encoding algorithm calculates, for each subblock, several actual points of the rate-distortion curve and uses them to assign the optimal number of bits to each subblock. This requires encoding each subblock to a high rate, usually 3 bits per pixel, and truncating each subblock's bit stream to the point determined by the rate assignment calculation. One can determine a set of optimal truncation points for each subblock corresponding to a set of target bit rates. Once the subblock bitstreams have been truncated for a certain target rate, they are merged and interleaved by common order bit planes to form an embedded composite bit stream. Note that for efficient implementation without disk swapping, the bitstreams of all subblocks must be held in memory to achieve this embedded compressed bit stream.

4 SET PARTITIONING CODERS

The JPEG-2000 encoding algorithm is a fairly complex and computationally intense procedure. It supports all the desirable features discussed previously. However, there has been concern as to its suitability for so-called embedded hardware applications. There are encoding algorithms of low complexity that possess all the desirable features and sacrifice very little, if at all, in performance compared to JPEG-2000. These algorithms have in common the partitioning of the wavelet transform into sets according to their significance as defined by a magnitude threshold on the maximum magnitude of coefficients within the set. We shall describe two such algorithms, called Set Partioning In Hierarchical Trees (SPIHT) [13], and Set Partioning Embedded bloCK (SPECK) coder[9]. Since SPIHT is by now well known, we shall start with SPECK and then explain their common attributes.

4.1 SPECK Codng

The SPECK encoding algorithm can directly replace the JPEG-2000 entropy coding of the wavelet subband subblocks. First we define a significance test for coefficients $c_{i,j}$ in a set of coefficients $\mathcal{T}$.

$$\mathcal{S}_n(\mathcal{T}) = \begin{cases} 1, & \text{if } 2^n \leq \max\limits_{(i,j)\,\in\,\mathcal{T}} |c_{i,j}| < 2^{n+1} \\ 0, & \text{else} \end{cases} \tag{1}$$

When the outcome of the test is "1", we say the set is *significant* for bit plane n, otherwise it is *insignificant*.

Starting with S as the full subblock and n at the highest non-all-zero bit plane n_{max}, S is significant for n_{max}, so we spit the subblock into four quadrant sets, collectively denoted $\mathcal{O}(S)$, as shown in Figure 1. Now we apply the significance test for the same n to each of these sets and split into four again only if significant. Significant sets continue to be recursively split until all there are four pixels, whereupon, the significants ones are found and appended to a list of significant pixels, called the LSP. Figure 2 depicts this recursive splitting.

Insignificant pixels and sets are listed in a list of insignificant sets, called the LIS, in order of increasing size from top to bottom. Then the bit plane order n is decremented by 1 and the sets in the LIS are tested from top to bottom (smallest to largest) in the same way as before, finding pixels significant for $n-1$ and moving them to the end of the LSP and finding pixels and sets insignificant for $n-1$ and putting their identifiers at the bottom of the LIS. The outcome of every test, "1" or "0", is put into the the code bit stream. When a single pixel is found significant, a "1" and a sign bit are put into the bit stream. Also, once the LSP is

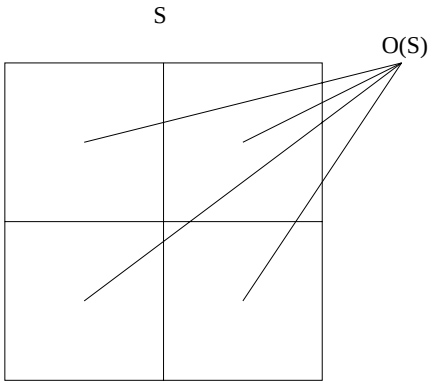

Figure 1. Partitioning of set $\mathcal{S}$.

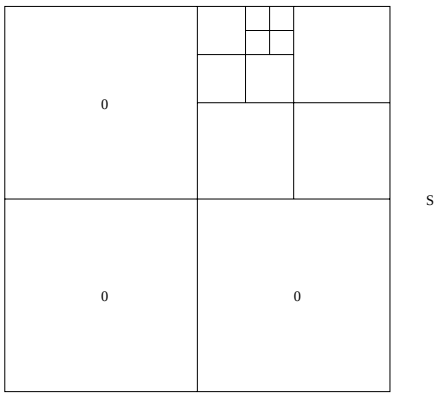

Figure 2. Recursive Partitioning of set $\mathcal{S}$

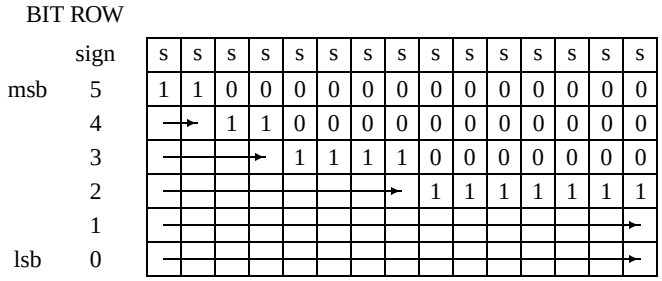

Figure 3. Binary representation and progressive transmission of the magnitude-ordered significant coefficients.

complete for $n-1$, the $n-1$ bits of all coefficients in the LSP found significant at all higher thresholds (the *refinement* bits) are put into the bitstream. The bit plane order n continues to be decremented until the bit budget of the subblock has been satisfied. An illustration of this progressive bit plane transmission is shown in Figure 3. Note that the highest "1" bit (signalling "significant") never increases as more coefficients are put into the LSP.

The code stream bits can be sent raw, meaning without further entropy coding, and still be efficient. However, the significance bits for four quadrants produced by recursive splitting can be Huffman or arithmetic coded to obtain a small improvement in efficiency. Sign and refinement bits need not be further encoded, as gains appear to be negligible. So when entropy coding is enacted in SPECK, it is far simpler than that of JPEG-2000. The SPECK calculations for coding are just finding highest order bits in a set which can be implemented by a bitwise OR of all its coefficients. So nothing more complicated than bitwise operations is needed for SPECK coding.

4.2 SPIHT Coding

The SPIHT algorithm, which preceded SPECK, differs from SPECK in the way the wavelet coefficients are grouped and partitioned. Otherwise, it is fairly similar, aside from some implementation details. In SPIHT, the coefficients of the wavelet transform are grouped into spatial orientation trees, that is, linked according to spatial orientation in subbands across scales. See Figure 4 for the llustration of these linkages. Since the subband dimensions increase by factors of two from coarse to fine scale, a coefficient at a coarser scale is linked to a 2x2 block of coefficiens at a finer scale. The exception is the roots of the trees that are in the low-low subband at the coarsest scale. Here the upper left coefficient of a 2x2 block has no linkages (descendants) and the other three are each linked to a 2x2 block in a subband at the same scale in the same spatial

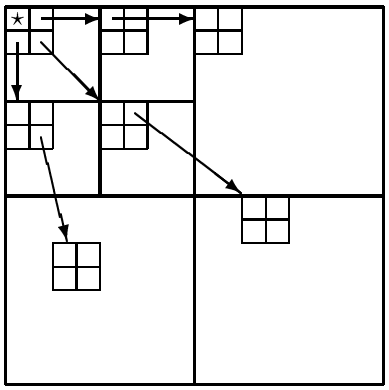

Figure 4. Examples of parent-offspring dependencies in the spatial-orientation tree.

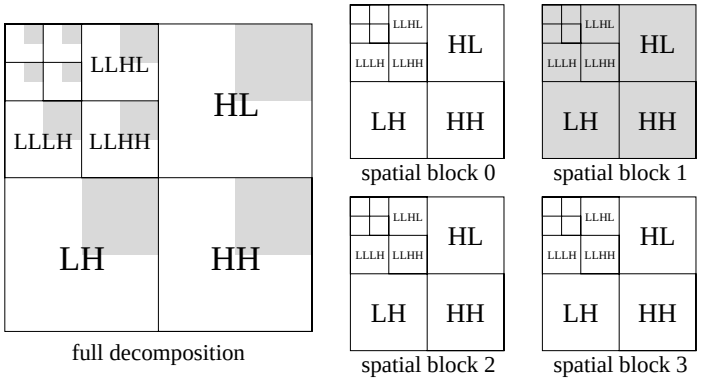

Figure 5. Grouping the spatial orientation trees of the wavelet transform. The shaded subbands belong to group or spatial block "1".

orientation.

The coordinates of coefficients in a tree are grouped into the following different sets:

- $\mathcal{O}(i,j)$: set of coordinates of all offspring of node (i,j);
- $\mathcal{D}(i,j)$: set of coordinates of all descendants of the node (i,j);
- $\mathcal{H}$: set of coordinates of all spatial orientation tree roots (nodes in the highest pyramid level);
- $\mathcal{L}(i,j) = \mathcal{D}(i,j) - \mathcal{O}(i,j)$.

For instance, except at the highest and lowest pyramid levels, we have

$$\mathcal{O}(i,j) = \{(2i,2j), (2i,2j+1), (2i+1,2j), (2i+1,2j+1)\}. \quad (2)$$

With these sets, we can test their coefficients for significance just as we did with SPECK. The set partitioning rules are simply:

1. the initial partition is formed with the sets $\{(i,j)\}$ and $\mathcal{D}(i,j)$, for all $(i,j) \in \mathcal{H}$;
2. if $\mathcal{D}(i,j)$ is significant then it is partitioned into $\mathcal{L}(i,j)$ plus the four single-element sets with $(k,l) \in \mathcal{O}(i,j)$.
3. if $\mathcal{L}(i,j)$ is significant then it is partitioned into the four sets $\mathcal{D}(k,l)$, with $(k,l) \in \mathcal{O}(i,j)$.

In SPIHT we maintain a separate list, the LIP, to store the coordinates of insignificant single coefficients, in addition to the LIP, and LSP. The LIP now stores all insignificant sets with the exception of the singleton ones. The LIP in SPIHT is always visited first for testing at a given bit plane order n, similar to the singleton sets at the top of the LIS in SPECK. This order of testing sets is necessary to find significant individual coefficients using the fewest number of significance decision bits.

As with SPECK, further entropy coding is optional. The significance test outcomes can be arithmetic or Huffman encoded in 2x2 blocks with a small benefit. Sign and refinement bits can be sent raw without further coding. Clearly, arithmetic or Huffman coding overhead of complexity is much lower than that of JPEG-2000. Again, only bitwise operations are needed for the main coding algorithm.

In SPECK, we described a small memory coding of subband subblocks, whereas here with SPIHT we described coding of the full wavelet transform. Spatial orientation trees with adjacent roots can be grouped and coded together apart from other groups. For example, consider the three level image wavelet transform shown in Figure 5 with the roots in the coarsest (low-low) subband separated into 4 groups, labelled 0,1,2,3. If we follow the trees from these groups, we see that they also manifest as 4 groups at the corresponding spatial orientations. Putting the like labelled groups together results in 4 groups of trees corresponds to 4 quadrants of the image with some overlap. Hence we call these groups "spatial blocks". Clearly, this paradigm extends to more levels of wavelet decomposition and a larger number of tree groups or spatial blocks. Each group can be encoded independently with SPIHT to a high rate and truncated according to a bit assignment procedure, such as the one alluded to previously for JPEG-2000 and SPECK. Then the individual bit streams can be reorganized as before to form an embedded composite bit stream.

4.3 Is Embedding Necessary?

Many applications do not require or can not afford an embedded bit stream. An embedded bit stream requires either the image or the compressed bit stream in memory. Clearly, this is not feasible for large images. Most large images are coded in tiles or stripes, where it is desirable to maintain constant quality across these coding units. One can encode to constant quality with any of the embedded systems mentioned by sending bits down to a given bit plane or pre-quantizing all the coefficients in every unit to the same quantization interval and encoding the signed bin numbers losslessly. With the last scenario, the bit plane transmission slows the execution and may give no added value to the application. In JPEG-2000, there is no choice but to pass through all the bit planes (several times). In SPIHT or SPECK, one can skip these refinement passes through the bit planes of significant coefficients (represented by bin numbers) and send all the lower order bits immediately upon finding a significant coefficient. There is still an approximate embedding by value as coefficients with larger most significant bits always precede those with smaller most significant bits. So, in this non-embedded scenario, SPECK and SPIHT have an even greater advantage in simplicity over JPEG-2000.

5 Complexity Comparisons

We conducted performance and time comparisons among SPECK, SPIHT and VM3.2A and VM4.2 versions of JPEG-2000. Later versions of JPEG-2000 (Part I) eliminated one of the original four passes per bit plane and made some other modifications to speed it up by a small amount and thereby sacrifice a little in performance. So the performance figures you will see for JPEG-2000 in these comparisons are better and the time figures are somewhat smaller than what is obtained with the final version. Nonetheless, these figures are good indicators of what can be obtained.

Both SPIHT and SPECK were implemented in the JPEG-2000 framework, replacing the entropy coding[15, 5]. There was no further entropy coding in SPIHT, as only raw uncoded sign, decision, and refinement bits were sent to the bit stream. With SPECK, the 4 bit binary masks resulting from significance test of quadrants of a divided block were encoded with a simple, fixed Huffman code of 15 symbols, through table lookup. Generally speaking, SPIHT performed about 1.0 - 1.5 dB below VM3.2A in PSNR (peak signal-to-noise-ratio) for a set of large photographic images. This particular implementation of SPECK, called SBHP for Subband Block Hierarchical Partitioning, performed only about 0.4-0.5 dB below VM4.2 in PSNR on the average over sets of photographic, medical, and satellite images.

The real story is in the reduction of coding times over JPEG-2000. Decoding time comparisons of algorithm are considered to be good indicators of relative speed and complexity. A non-embedded form of SPIHT was 10 times faster in decoding than VM3.2A. For SPECK, the embedded form was 8 times faster and the non-embedded form was 11 times faster than VM4.2. These numbers show very significant complexity reductions of SPECK and SPIHT over JPEG-2000. The consequent degradation in performance is nearly insignificant, especially with SBHP.

6 IMAGE SEQUENCE CODING

Three-dimensional versions of SPIHT and SPECK have been very effective in encoding video[11, 10, 8]. Even without motion compensation, SPIHT looks and measures better than MPEG2 for the same bit rate. SPECK, in a the particular implementation EZBC, has achieved some of the best results yet reported for SIF sequences. For SPIHT, the data structure is 3-dimensional (3-D) spatio-temporal orientation trees of a 3-D wavelet transform. For SPECK, the data strucure is a 2-dimensional (2-D) block of a 3-D wavelet tranform. Here, the temporal transform is taken along the estimated motion trajectories of pixels in the image frames. SPECK and SPIHT have also been utilized for coding of non-video image sequences, where there is no underlying motion model. Examples of such sequences are tomographic images, medical, material or geological, and multi-component images, such as multi-spectral or hyperspectral images.

There has been a growing trend to compress data associated with points of a two or three dimensional grid. Image compression techniques can be adapted for such uses. One example is compression of digital terrain elevation data, for which SPIHT has already been successfully utilized[6]. Another would be atmospheric data of temperature, pressure, moisture, or wind velocity at a grid of points in a volume of the atmosphere. For data, one might have to adopt an error criterion different than the usual mean squared error, such as bounded error magnitude. For the latter, algorithms entirely different than the ones mentioned here might have to be employed.

6.1 Coding Results with Volume Images

The undertaking of 3-D transforms and/or 3-D coding, considering the increase in memory and complexity required, may not result in a worthwhile payoff. To settle this question, we present some coding results with medical image sequences, comparing 2-D and 3-D techniques. The test sequences are an MR Chest volume, comprised of 64 256x256, 8 bpp frames and a CT Skull volume, comprised of 128 256x256, 8 bpp frames. First we compare the techniques for purely lossless (perfectly reversible) compres-

Method	GOF	Filters	Bit Rate (bits per pel)	
			MR Chest	CT SKull
3-D/3-D SPIHT	16	I(4,2)	1.78	2.04
3-D/2-D SPECK	16	I(5,3)	2.176	2.47
2-D SPIHT	1	I(4,2)	2.85	2.69
JPEG-LS	1	none	2.93	2.85

Table 1. Lossless Volume Image Compression

	SPIHT at 0.1 bpp			SPECK at 0.25 bpp		
	GOF		Δ	GOF		Δ
	1	16	(dB)	1	16	(dB)
MR Chest	36.38	42.98	6.60	39.46	45.60	6.14
CT SKull	26.38	33.98	7.60	29.66	36.51	6.85

Table 2. 3-D to 2-D Coding Improvement for Lossy Volume Image Compression

sion. The techniques are SPIHT with 3-D transform and 3-D coding, SPECK with 3-D transform and 2-D coding, 2-D SPIHT, and JPEG-LS, the new lossless and near-lssless still image compression standard. The results are shown in Table 1. The best compression for both volumes is achieved by 3-D/3-D SPIHT. Averaged over both volumes, 3-D/2-D SPECK requires 22% larger file size, while the purely 2-D techniques, SPIHT and JPEG-LS require 46% and 52% larger file sizes, respectively. Clearly, the 3-D transform alone gives fairly impressive gains in coding efficiency, but when combined with 3-D coding, the gains are even more impressive. However, these gains may occur only for this kind of image data, but they are encouraging.

For lossy coding of these image volumes, we compared SPIHT and SPECK between their 2-D and 3-D modes and compiled some results in Table 2. With SPIHT at 0.1 bpp, the average gain in PSNR between 2-D and 3-D was 7.10 dB. With SPECK at 0.25 bpp, the analogous average gain was 6.49 dB. The lower gain of SPECK is attributed to the 2-D coding mode versus the 3-D coding mode of SPIHT. These PSNR gains are really quite significant and make a substantial visual impact.

7 CONCLUSIONS

The demand for image compression seems to be accelerating rapidly along with the venues and set of features for its application. The trend is toward higher resolution, larger 2D and 3D images. The volume medical images above benefit greatly from 3D transform and coding, but they are relatively small. Although coding and decoding times are adequately fast for smaller images and image volumes, they might not be so for the larger images contemplated for future applications. For example, there is a call for proposals by MPEG for a new standard for digital cinema. The requirements are real-time, visually lossless, compression and decompression of 1920x1080 pixel, 6 byte per pixel frames, at 24 frames per second. It appears that only a hardware solution using very fast, low complexity algorithms is feasible. The set partitioning coders presented here are good candidates for meeting these requirements while achieving state-of-the-art compression efficiency. There are other, set partitioning coders that use more adaptive set partitioning schemes to obtain slightly better performance with a small increase in complexity[3, 12, 7]. We believe that the set partitioning paradigm is essential to keep the computational cost and complexity resonably low and still achieve superior compression efficiency.

References

[1] M. D. Adams and F. Kossentini. Performance evaluation of the spatially segmented wavelet transform in the jpeg-2000 baseline system. *ISO/IEC JTC 1/SC 29/WG 1 N868*, June 1998.

[2] Motorola Australia Research Center. A memory saving method of calculating the discrete wavelet transform,. *Project Report*, December 1997.

[3] B.-B. Chai, J. Vass, and X. Zhuang. Significance-linked connected component analysis for wavelet image coding. *IEEE Trans. on Image Processing*, 8:774–784, June 1999.

[4] C. Chrysafis and A. Ortega. Line based, reduced memory, wavelet image compression. *IEEE Trans. on Image Processing*, 9:378–389, March 2000.

[5] C. Chrysafis, A. Said, A. Drukarev, A. Islam, and W. A. Pearlman. Sbhp-a low complexity wavelet coder. *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP 2000)*, 4:2035–2038, 2000.

[6] W. R. Franklin and A. Said. Lossy compression of elevation data. *Proceedings of Seventh International Symposium on Spatial Data Handling*, Aug. 1996.

[7] E. S. Hong and R. E. Ladner. Group testing for image compression. *Proceedings of Data Compression Conference (DCC 2000)*, pages 3–12, Mar. 2000.

[8] S.-T. Hsiang and J. W. Woods. Embedded video coding using invertible motion compensated 3-d subband/wavelet filter banks. *Proceedings of Packet Video Workshop (PV 2000)*, May 2000.

[9] A. Islam and William A. Pearlman. An embedded and efficient low-complexity hierarchical image coder. *Visual Communication and Image Processing '99, Proc. SPIE Vol. 3653*, pages 294–305, Jan. 1999.

[10] B.-J. Kim and W. A. Pearlman. An embedded wavelet video coder using three-dimensional set partitioning in hierarchical trees. *Proceedings of Data Compression Conference (DCC 97)*, pages 251–260, Mar. 1997.

[11] B.-J. Kim, Z. Xiong, and W. A. Pearlman. Low bit-rate scalable video coding with 3d set partitioning in hierarchical trees (3d spiht). *IEEE Trans. Circuits and Systems for Video Technology*, 10:1374–1387,, December 2000.

[12] D. Marpe and H. L. Cycon. Very low bit-rate video coding using wavelet-based techniques. *IEEE Trans. on Circuits and Systems for Video Technology*, 9:85–94, Feb. 1999.

[13] Amir Said and William 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*, 6:243–250, June 1996.

[14] D. Taubman. High performance scalable image compression with ebcot. *IEEE Trans. on Image Processing*, 9:1158–1170, July 2000.

[15] F. W. Wheeler and W. A. Pearlman. Low-memory packetized spiht image compression. *Conference Record of Thirty-Third Annual Asilomar Conference on Signals, Systems, and Computers*, 2:1193–1197, Oct. 1999.

[16] Z. Xiong, O. Guleryuz, and M. T. Orchard. A dct-based embedded image coder. *IEEE Signal Processing Letters*, 3:289–290, November 1996.