A HIGHLY SCALABLE SPECK IMAGE CODER

Gui Xie and Hong Shen

Graduate School of Information Science
Japan Advanced Institute of Science and Technology
Tatsunokuchi, Ishikawa, 923-1292, Japan

ABSTRACT

Pearlman's SPECK image coding algorithm is a distortion scalable coder which achieves excellent compression performance with very low computational complexity. However, SPECK has no resolution and ROI (region of interest) salabilities. In this paper, we extend it to a novel highly scalable image coder named S-SPECK through an efficient strategy of grouping the wavelet coefficients according to their resolution levels and relationship with ROIs. The proposed new coder not only retains all the advantages of SPECK, but also implements resolution and ROI scalabilities, which, we believe, is a better alternative than JPEG2000 for some applications. Extensive experiments have been conducted to verify all our claims.

1. INTRODUCTION

For modern multimedia applications, particularly in the Internet environment, it is desirable to implement a high compression image coder that supports rich features, such as low computational complexity, distortion scalability, resolution scalability and ROI (region of interest) retrievability. In general, the above features are controversial for a coder to achieve high compression performance. Pearlman's SPECK [2] is a distortion scalable coder, which achieves excellent coding performance with very low computational complexity. As reported by the data on comparative run-times with JPEG 2000's VM 3.2A (Verification Model, version 3.2A) in [3], which is essentially the EBCOT coder [1], SPECK is 4.6 to 15.7 faster than VM 3.2A in encoding and 8.1 to 12.1 faster in decoding on the average over a set of four images and set of four rates, 0.25, 0.50, 1.0, 2.0 bits per pixel, while the reduction of PSNR from that of VM 3.2A ranges only from a minimum of 0.48 dB for entropy-coded versions to a maximum of 0.85 db for non-entropy-coded versions. However, SPECK doesn't support resolution scalability and ROI retrievability. This paper addresses this problem and successfully extends SPECK to a new image coder called S-SPECK, which not only achieves excellent compression performance with very low complexity, but also retains distortion scalability, resolution scalability and ROI retrievability. The acronym S-SPECK is derived from the description "scalable set-partitioning embedded block coder" which identifies some of the major characteristics of the proposed image coder.

This work is supported by Japan Society for the Promotion of Science (JSPS) Grant-in-Aid for Scientific Research (B) under Grant No.14380139.

2. CODEUNIT CONSTRUCTION

S-SPECK use a biorthogonal wavelet transform to decompose the original image into different subbands. K-level decomposition results in $3K + 1$ subbands, by which the original image can be reconstructed at up to $K + 1$ resolution levels. Therefore, in order to achieve resolution scalability, it is reasonable to group the wavelet coefficients into distinct subsets in terms of their correspondent resolution levels and then code these subsets independently.

Moreover, the wavelet coefficients in the hierarchical octave-band decomposition system are highly spatially correlated with respect to some square regions of the original image. A tree structure suggested in [5] can be used to organize the wavelet coefficients, by which except that at the highest and lowest pyramid levels, each coefficient located at (i, j) has four children located at $(2i, 2j)$, $(2i, 2j + 1)$, $(2i+1, 2j)$ and $(2i+1, 2j+1)$. The number of trees equals the number of coefficients located in the lowest frequency approximation subband. A square region within the original image can be reconstructed by the coefficients of a tree independently if the filters used to decompose the original image are short enough, even though the reconstruction is lossy around the border of that region.

S-SPECK therefore constructs the codeunits based on the above two facts by first organizing the wavelet coefficients into spacial trees and then grouping the coefficients of each tree into distinct subsets, i.e., the codeuinits, according to the resolution levels we want the final coded bitstream to hold. Each codeunit constructed here is compressed independently. Note that the coefficients in a tree are also a hierarchical structure same as that of the decomposed image, which is the basis of applying SPECK algorithm on these

0-7803-8554-3/04/$20.00 ©2004 IEEE.

codeunits independently. Fig. 1 illustrates this grouping process, where three codeunits according to tree resolution levels are constructed in each tree.

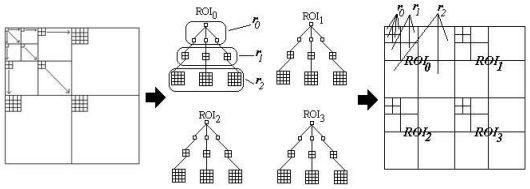

Fig. 1. Grouping the wavelet coefficients into codeunits by trees and resolution levels. Three codeunits in each tree according to three resolution levels are constructed.

3. OPTIMAL QUALITY LAYER FORMATION

After codeunits are constructed, SPECK algorithm is applied to compress them independently. SPECK is a typical zeroblock coder [6] (as opposed to zerotree coders), which employs a hierarchical quadtree decomposition algorithm to recursively divide a region into homogeneous sub-regions whenever the set of the coefficients inside that region test significant. Zeroblock is a set containing all the insignificant coefficients with respect to a given threshold.

In our S-SPECK coder, each codeunit is compressed by SPECK algorithm into a coded bitstream denoted by c_i. The straightforward method of construction of the overall compressed bitstream is to concatenate all suitable truncated versions of c_i. Such a bit-stream is resolution scalable, because all information representing individual codeunits is retained and hence the subbands and resolution levels are clearly delineated. Also the bit-stream possesses ROI scalability, because all the necessary codeunits required to reconstruct a region of interest are easily identified.

However this simple concatenated bit-stream is not distortion scalable, even though its individual codeunits are compressed in an embedded fashion. To solve this problem, a quality layer structure introduced in the EBCOT algorithm [1], is used here, but constructed by a different method. The principle of constructing quality layers is to minimize the distortion of a given overall target bit rate by distributing a different number of bits of c_i to those layers, which are identified by the truncation points. EBCOT utilizes a one-pass bit-rate control method known as PCRD [1] to compute the truncated points, which requires computing the increases in bit rate and the decrease in distortion for each bit-plane coding pass. Though the computation of the number of bits is straightforward, the computation of decrease in distortion is time-consuming because it requires computation of square values for each coded pixel. In S-SPECK, a very different, but much faster method to format the optimal quality layers is proposed, which is executed during the encoding process of S-SPECK instead of applying PCRD after finishing encoding as EBCOT does.

We have described the main principle of the S-SPECK coder that each codeunit is compressed independently by maintaining its own LIS and LSP. In fact, all the individual LIS's and LSP's can be combined together to one LIS_c and LSP_c respectively. The same zeroblock coder is operated on these two combined lists. For each element of LIS_c or LSP_c appearing during the encoding process, it is easy to identify the codeunit where it is located using the coordinates of that element. During the encoding process, each output bit out of the encoder results from operating on some element of LIS_c or LSP_c, for example, significance testing on a S-Type set in the sorting pass, and outputting the current most significant bit of a significant coefficient in a refinement pass. So each output bit is related to a codeunit. Using this relationship, we can distribute bits output from the encoder into their correspondent positions in a quality layer during the encoding process. As illustrated in Fig. 2, A quality layer Q_j is formed by the bits distributed from the encoder according to their related codeunits. When the maximal length of the quality layer Q_j is met, a new empty quality layer Q_{j+1} replaces the current quality layer Q_j and waits for the bits distributed from the encoder. This formation method guarantees that the quality layers in S-SPECK are optimal in the sense that a quality layer contains as many significant coefficients as possible.

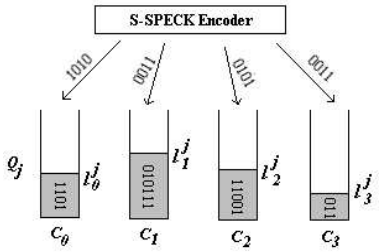

Fig. 2. Bits distribution for a quality layer. l_i^j is the length of bits distributed from the encoder to the bitstream of codeunit c_i in quality layer Q_j.

4. ENTROPY-CODING AND COMPLEXITY ANALYSIS

The significance map during the S-SPECK encoding process can be compressed losslessly using arithmetic coding with simple context-based models as suggested in [2]. The significance maps are the binary decisions created by the recursive partitioning process. We have chosen not to introduce this entropy-coding procedure into S-SPECK in the

following experiments because we want to test the impact of our modifications on SPECK. It's straightforward to add an entropy coding procedure into S-SPECK, which can improve S-SPECK's compression performance to the same degree as it did in SPECK.

The expected increase of the computational complexity of the highly scalable coder S-SPECK owns to two new introduced procedures: wavelet coefficients grouping and quality layer formation. Wavelet coefficients grouping only needs to visit the coefficients one time by the tree structure, which is a fast operation. As for the quality layer formation procedure that is embedded during the encoding process, it has little negative effect on S-SPECK because it does the formatting work simply by distributing the bits according to their correspondent codeunits.

5. NUMERICAL RESULTS

The following results were obtained with three standard, $8bpp$, 512×512 images, Lena, Barbara, and Goldhill. We used 7-level pyramids constructed with 9/7-tap filters of [4], and using a "reflection" extension at the image edges. Each image here is coded by S-SPECK into a final coded bitstream containing four quality layers at bit rates 0.125 bpp, 0.25 bpp, 0.5 bpp and 1.0 bpp and three resolution levels 512×512, 256×256 and 128×128.

Table 1 shows the comparison of the reconstructed image PSNR performance at the rates 0.125, 0.25, 0.5 and 1.0 for non-entropy-coded versions of SPECK and S-SPECK. Results in Table 1 are obtained without entropy-coding the significance decision bits because it's enough to verify that the compression performance of S-SPECK is competitive to SPECK. The Δ rows give the percentage of PSNR loss of the new coder compared with SPECK.

Tables 2 and 3 give the comparison of the encoding and decoding times between SPECK and S-SPECK at different bit rates, when they run on a 2GHz Pentium-4 processor. To get an objective evaluation, we try to keep the testing conditions similar for both of the coders, such as programming language, data structure and platform. Because the absolute speed depends a lot on the testing environment, a relative measurement—the ratios of S-SPECK's running times to SPECK's running times—are given in the "$\times$" rows of Table 2 and 3. From these ratios, we can see, both of the S-SPECK's encoding and decoding speeds are almost same as those of SPECK. The negligible speed loss owns to the introduction of wavelet coefficients grouping process in S-SPECK coder.

Comparative evaluation of the new coder S-SPECK in respect to the benchmark coder SPIHT is illustrated in Fig. 3. We can see, at different bit rates for the standard test images, S-SPECK is quite competitive to SPIHT.

Fig. 4(a) illustrates an example of S-SPECK's resolution scalability, where three resolution levels of the same original Goldhill image (512×512, 256×256 and 128×128) are reconstructed from the coded bitstream holding three quality layers at bit rates 0.125 bpp, 0.25 bpp, and 0.5 bpp. Fig. 4(b) illustrates an example of S-SPECK's ROI retrievability scalability, where the same region of the Lena image is retrieved from the coded bitstream at different bit rates 0.125 bpp, 0.25 bpp, 0.5 bpp and 1.0 bpp.

Table 1. Comparison of reconstructed image PSNR at different bit rates between non-entropy-coded versions of SPECK and S-SPECK using common test images

Coder	PSNR(dB)			
	0.125 bpp	0.25 bpp	0.5 bpp	1.0 bpp
Lena image (512×512)				
SPECK	30.80	33.77	36.86	39.90
S-SPECK	30.65	33.63	36.76	39.85
Δ	-0.49%	-0.41%	-0.27%	-0.13%
Barbara image (512×512)				
SPECK	25.40	28.21	32.07	37.16
S-SPECK	25.23	28.10	31.93	37.06
Δ	-0.67%	-0.39%	-0.44%	-0.27%
Goldhill image (512×512)				
SPECK	28.29	30.27	32.74	36.01
S-SPECK	28.19	30.16	32.65	35.93
Δ	-0.35%	-0.36%	-0.27%	-0.22%

Table 2. Comparison of encoding times at different bit rates between non-entropy-coded versions of SPECK and S-SPECK using common test images.

Coder	Encoding Speed(μs per pixel)			
	0.125 bpp	0.25 bpp	0.5 bpp	1.0 bpp
Lena image (512×512)				
SPECK	1.01	1.37	1.93	3.02
S-SPECK	1.14	1.50	2.08	3.19
$\times$	1.13	1.09	1.08	1.07
Barbara image (512×512)				
SPECK	0.88	1.17	1.75	2.84
S-SPECK	1.12	1.44	1.99	3.09
$\times$	1.27	1.23	1.13	1.09
Goldhill image (512×512)				
SPECK	1.03	1.43	1.96	3.06
S-SPECK	1.19	1.61	2.13	3.22
$\times$	1.16	1.13	1.09	1.05

Table 3. Comparison of decoding times at different bit rates between non-entropy-coded versions of SPECK and S-SPECK coder using common test images.

Coder	Decoding Speed(μs per pixel)			
	0.125 bpp	0.25 bpp	0.5 bpp	1.0 bpp
Lena image (512×512)				
SPECK	0.22	0.41	0.77	1.61
S-SPECK	0.30	0.51	0.97	1.84
$\times$	1.36	1.24	1.26	1.14
Barbara image (512×512)				
SPECK	0.24	0.42	0.81	1.63
S-SPECK	0.30	0.55	0.98	1.87
$\times$	1.25	1.31	1.21	1.15
Goldhill image (512×512)				
SPECK	0.17	0.42	0.85	1.68
S-SPECK	0.28	0.53	0.99	1.91
$\times$	1.64	1.26	1.16	1.14

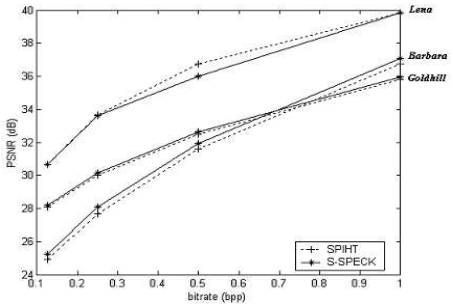

Fig. 3. Comparison of the PSNR performance at 0.125 bpp, 0.25 bpp, 0.5 bpp and 1.0 bpp between non-entropy-coded versions of S-SPECK and SPIHT. The dotted line corresponds the SPIHT coder.

(a) Reconstructed images of three resolution levels at target bit rate 0.5 bpp.

(b) A region retrieved from different bit-rate coded bitstream at 0.125 bpp, 0.25 bpp, 0.5 bpp and 1.0 bpp

Fig. 4. S-SPECK's scalabilities

6. CONCLUSION

We described a new wavelet-transform-based image coder S-SPECK, which extends the original coder SPECK successfully to a highly scalable scheme. Extensive experiments showed that the loss of S-SPECK's compression performance and computational speed is negligible compared with SPECK. It's desirable to apply S-SPECK into modern multimedia applications in the Internet environments. Future work could be on implementing the entropy-coded version of S-SPECK and extend it to a video coding algorithm.

7. REFERENCES

[1] D. Taubman, "High performance scalable image compression with ebcot," *IEEE Trans. Image Processing*, vol. 9, pp. 1158–1170, July 2000.

[2] 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 Syst. Video Technol.*, In Press.

[3] W. Pearlman, "Presentation on core experiment codeff 08: Set partitioned embedded block coding (speck)," ISO/IEC/JTC1,SC29, WG1 N1245, 1999.

[4] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies, "Image coding using wavelet transform," *IEEE Trans. Image Processing*, vol. 1, pp. 205–220, Apr. 1992.

[5] A. Said and W. A. Pearlman, "New fast and efficient image codec based on set partitioning in hierarchical trees," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, pp. 243–250, June 1996.

[6] S.-T. Hsiang, "Highly scalable subband/wavelet image and video coding," Ph.D. dissertation, Rensselaer Polytechnic Institute, Troy, NY 12180, USA, Jan. 2002.