Highly Scalable, Low-Complexity Image Coding Using Zeroblocks of Wavelet Coefficients

Gui Xie, *Student Member, IEEE,* and Hong Shen

***Abstract*—We propose a new highly scalable wavelet transform-based image coder, called S-SPECK, on the extension of a well-known zero-block image coder SPECK, by achieving not only distortion scalability, resolution scalability, and region of interest (ROI) retrievability, but also excellent compression performance with very low computational complexity. Though new features have been introduced into S-SPECK, our coder is quite competitive with SPECK on compression performance (peak signal-to-noise ratio) and computational complexity (encoding and decoding times) at various bit rates for standard test images. A novel quality layer formatting method is implemented in S-SPECK, which is much simpler and faster than PCRD used in JPEG2000. Extensive experiments have verified all our claims for S-SPECK.**

***Index Terms*—Distortion scalability, quality layer, resolution scalability, region of interest (ROI) retrievability, S-SPECK, wavelet transform, zeroblock.**

I. 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 region of interest (ROI) retrievability. In general, the above features are incompatible for a coder to achieve high compression performance. The SPECK coder proposed by Pearlman *et al.* [1] is a distortion scalable coder, which achieves excellent coding performance with very low computational complexity. Comparative run-times, as reported in [2], show that SPECK is 4.6 to 15.7 times faster than JPEG2000's VM 3.2 A (Verification Model, version 3.2 A), which is essentially the EBCOT coder [3], 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, and 2.0 bits per pixel. Meanwhile the reduction of PSNR from that of VM 3.2 A ranges only from a minimum of 0.48 dB for entropy-coded versions to a maximum of 0.85 dB for nonentropy-coded versions. However, SPECK does not support resolution scalability and ROI retrievability. Here we addresse this problem and successfully extend 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 new coder, S-SPECK, is related in various degrees to some earlier work on scalable image compression, such as Shapiro's EZW [4], Said and Pearlman's SPIHT [5], and Taubman's EBCOT [3]. SPIHT is a successful extension and improvement of Shapiro's EZW [4] algorithm based on set partitioning in hierarchical trees, and has been a standard benchmark in image compression. EBCOT, as the core algorithm of the new still image compression standard JPEG2000 [6], [7], is a landmark in the field of image compression achieving high compression performance while retaining distortion scalability, resolution scalability, and ROI retrievability. To incorporate all the above rich features in one frame, EBCOT codes groups of wavelet coefficients, called codeblocks, independently, and utilizes time-consuming method PCRD [3] to find the optimal truncation points. A complex arithmetic coder is also introduced into EBCOT to guarantee high compression performance. The SPECK algorithm is much less complex than EBCOT because it uses a simple and efficient structure, zeroblock, to exploit the redundancy of the clustered wavelet coefficients without introducing any other complex procedures. The new proposed coder, S-SPECK, also utilizes zeroblocks to code wavelet coefficients bitplane by bitplane to guarantee simplicity and compression performance, while incorporating some strategies of organizing the wavelet coefficients and the bits in the coded bitstream to support distortion, resolution scalability, and ROI retrievability. Our experiments have proved that S-SPECK's compression performance and computational complexity are similar to those of SPECK, while all the rich features previously provided only by EBCOT are also supported. We believe that this new coder would be a better alternative to JPEG2000 for some applications.

The S-SPECK coder contains the following main features.

- It uses a discrete biorthogonal wavelet transform to decompose the original image I, which provides a multiresolution representation of I.
- It is a fast codec, whose computational complexity is similar to that of SPECK.
- It generates a resolution scalable bitstream which contains distinct subsets B_r representing the samples from all the necessary subbands at each successive resolution level r. The number of the resolution levels to be held in the final coded bitstream can be set by the users.

Manuscript received August 2, 2003; revised March 28, 2004. This work is supported by Japan Society for the Promotion of Science (JSPS) Grant-in-Aid for Scientific Research (B) under Grant 14380139. This paper was recommended by Associate Editor H. Sun.

G. Xie is with the Graduate School of Information Science, Japan Advanced Institute of Science and Technology, Ishikawa 923-1292, Japan (e-mail: g-xie@jaist.ac.jp).

H. Shen is with the Graduate School of Information Science, Japan Advanced Institute of Science and Technology, Ishikawa 923-1292, Japan, and also with the Department of Computer Science, University of Science and Technology, Hefei 230026, China (e-mail: shen@jaist.ac.jp).

Digital Object Identifier 10.1109/TCSVT.2005.848311

- It generates an embedded distortion scalable bitstream which contains distinct subsets, B_q, such that $\bigcup_{k=0}^{q} B_k$ together represents the samples from all subbands at some reconstruction quality level, q. The embedded bitstream output from S-SPECK can be transmitted progressively and truncated at any point to get an optimal or suboptimal representation of the original image I. S-SPECK runs sequentially and can stop whenever a target bit rate or a target distortion is met.
- It generates an ROI-retrievable bitstream which contains distinct subsets B_i representing all the necessary samples required to reconstruct a region i inside the original image I. Due to the short filters with linear phase property used in image decomposition and reconstruction, the quality of an ROI reconstructed by a subset is good enough for viewer perception.

This paper is organized as follows. The next section, Section II, describes the strategies for grouping the wavelet coefficients into codeunits, which is the basis of implementing a scalable coder. In Section III, we explain the principles of zeroblock coding and how we incorporate this technique into our coder. A new simple and efficient method for formatting the optimal quality layers, very different from PCRD used in EBCOT, is described in Section IV. In Section V, S-SPECK is presented in detail in a pseudocode language. Section VI discusses entropy coding and computational complexity of S-SPECK. In Section VII, we describe rate, distortion, and execution time results obtained by operating S-SPECK on some standard test images. The advantages of the new coder S-SPECK are verified by the numerical data. The conclusion of the paper is in the last section.

II. Wavelet Coefficients Grouping

S-SPECK uses a biorthogonal wavelet transform to decompose the original image into different subbands, which is identical to a hierarchical octave-band decomposition. K-level decomposition results in $3K+1$ subbands. The subbands at each decomposition level are related to some resolutions.

The quad-tree structure and organization of subbands into resolution levels are shown in Fig. 1. The lowest resolution level, R_0, consists only of the lowest frequency subband, LL_K. The next lowest resolution level, R_1, contains the additional three horizontal, vertical, and diagonal high frequency subbands required to reconstruct LL_{K-1}. In general, if we interpret the original image as LL_0, levels R_0 through R_r together contain the subbands required to synthesize the reduced resolution image LL_{K-r} of size $(M)/(2^{K-r}) \times (N)/(2^{K-r})$. We say a coded bitstream is resolution scalable if the compressed representation of LL_{K-r} may be obtained by simply discarding the elements corresponding to resolution R_{r+1} through R_K. Therefore, to get resolution scalability, it is reasonable to group the coefficients within subbands into distinct subsets according to the resolution levels and then code the subsets independently.

Moreover, the wavelet coefficients in the pyramid subband system are highly spatially correlated with respect to some regions of the original image. The parent–offspring dependencies

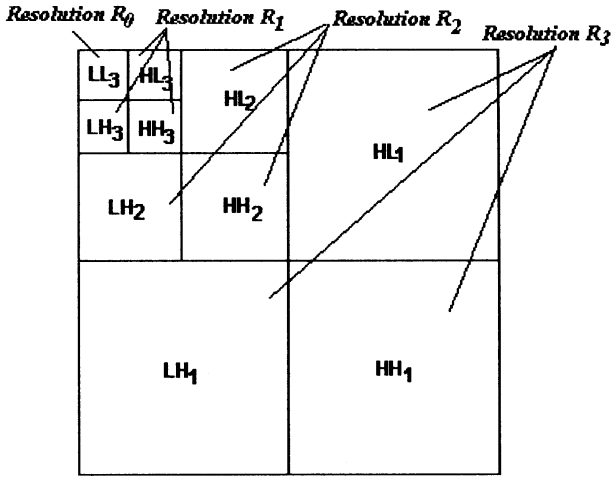

Fig. 1. Resolution levels within a dyadic quadtree-structured subband decomposition with depth $K = 3$.

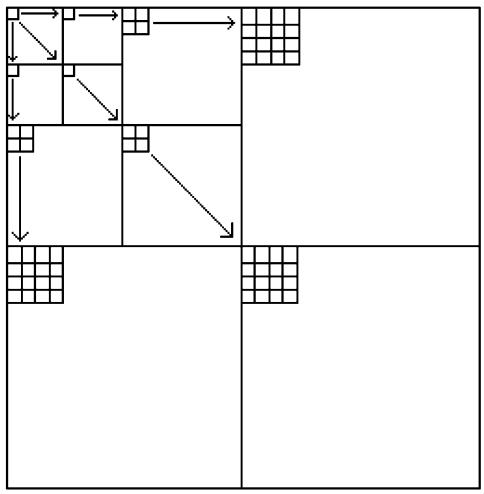

Fig. 2. Parent–offspring dependencies in the spatial orientation tree.

in the spatial orientation tree are shown in Fig. 2. All the coefficients are organized by trees with the roots located inside the lowest frequency subband. The tree structure is similar to that in [4], 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 $(2i+1,2j+1)$. A square region can be independently reconstructed by the coefficients of a tree if the filters used to decompose the original image are short enough, even though the reconstruction is lossy around the border of that region.

Suppose that an image I of size $M \times N$ is decomposed at level K. The number of trees we can organize, denoted N_T, equals the number of coefficients inside the lowest frequency subband. We have

$$N_T = \frac{M}{2^K} \times \frac{N}{2^K} \tag{1}$$

so N_T regions of size $2^K \times 2^K$ can be retrieved. If we denote the coordinate of a coefficient in the lowest frequency subband by

(a) (b)

Fig. 3. Retrieving regions by trees in a 7-level decomposition pyramid of the Lena image. (a) The original 512 × 512 grayscale Lena image. (b) Retrieved regions.

(i, j) and the coordinate of the top left corner of the region corresponding to the tree with the root located at (i, j) by (i^*, j^*), we have the following relationship

$$\begin{cases} i^* = i \times 2^K \\ j^* = j \times 2^K \end{cases}. \tag{2}$$

A good example is shown in Fig. 3, where 16 square regions in Fig. 3(b) are retrieved independently by the organized trees in the 7-level pyramid decomposition of the Lena image in Fig. 3(a).

Therefore, to retain ROI retrievability, it is reasonable to group coefficients into trees and code them independently.

To retain both resolution scalability and ROI retrievability, S-SPECK first organizes the wavelet coefficients into trees with the roots located inside the lowest frequency subband and then further groups the coefficients in each tree into subgroups according to the resolution levels we want the coded bitstream to hold. These subgroups, called codeunits in this paper, are the coding units of S-SPECK, which are compressed independently based on a modified SPECK coder. A good example of the grouping procedure is depicted in Fig. 4. Four ROIs can be retrieved and three resolution levels are held in the coded bitstream. It is worthy to note that the tree has a pyramid structure similar to that of the entire wavelet coefficients matrix, so the coefficients in each tree can be put together to construct a square block of the same size as its corresponding ROI. Thereby, we can apply SPECK coders independently for the various codeunits. For example, as shown in Fig. 4, the block ROI_0 consists of the coefficients in tree ROI_0 (the first tree in Fig. 4) and has the same pyramid structure as depicted in Fig. 1.

III. Zeroblock Coding

SPECK is a typical zeroblock coder [8] (as opposed to zerotree coders), employing a hierarchical quadtree decomposition algorithm to recursively divide a region into homogeneous subregions whenever the set of the coefficients inside that region test as significant. Zeroblock is a set containing all the insignificant coefficients with respect to a given threshold. Because zeroblock coding is conceptually simple and very efficient, it has been successfully applied in wavelet bitplane coding [1], [9]–[11].

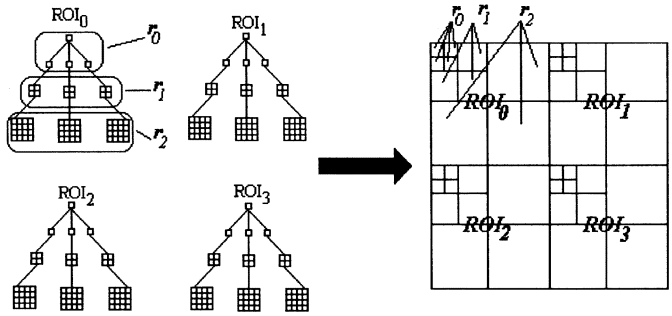

Fig. 4. Grouping the wavelet coefficients by trees and resolution levels.

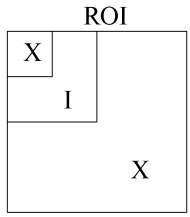

Fig. 5. Codeunits in an ROI block with three resolutions supported.

Following the ideas of SPECK, a significance test function on a set T of wavelet coefficients is defined as

$$\Gamma_n(T) = \begin{cases} 1, & 2^n \le \max_{c_{i,j} \in T} |c_{i,j}| < 2^{n+1} \\ 0, & \text{else} \end{cases} \tag{3}$$

where $c_{i,j}$ is the magnitude of the wavelet coefficient located at (i, j). We say T is significant if $\Gamma_n(T) = 1$, otherwise it is insignificant.

In our coder, the wavelet coefficients are grouped together to ROI blocks and then partitioned further into codeunits to be coded independently. In Fig. 5, a ROI block has been partitioned into several codeunits according to the resolution levels we want the coded bitstream to hold. There are three different type sets among these codeunits (see Fig. 5).

X−Type A codeunit is an X-type set if it is a square region located at the top left corner of a ROI block.

I−Type A codeunit is an I-type set if it is obtained by chopping off a small square region from the top left portion of a larger square region.

S−Type A codeunit is an S-type set if it is a square region generated by partitioning a X-type or I-type set.

These codeunits are independently processed by zeroblock coders similar to SPECK based on three partitioning rules depicted in Fig. 6 according to their set types. The last two rules for I-type and S-type sets are the same as those used in SPECK. The first rule partitions an X-type set into two sets: one is an S-type set holding only one coefficient (dc component) located at the top left corner of a ROI block and the other an I-type set containing all the other coefficients in that X-type set.

Let N_R be the number of resolutions we want the final coded bitstream to hold. Each ROI block is initially partitioned into $(N_R - 1)$ I-type codeunits and one X-type codeunit, as shown in Fig. 5, where $N_R = 3$. For each codeunit, two linked lists: list of insignificant sets (LIS) and list of significant pixels (LSP) are maintained. The former contains sets of varying sizes which have not been found significant against a threshold n, while the

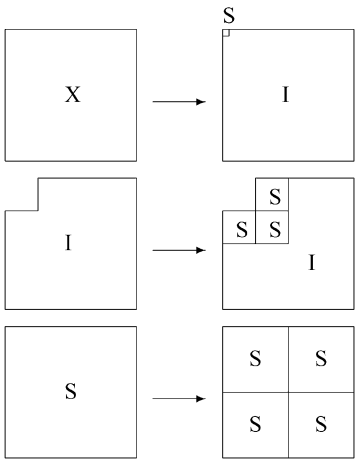

Fig. 6. Three rules for partitioning different type sets.

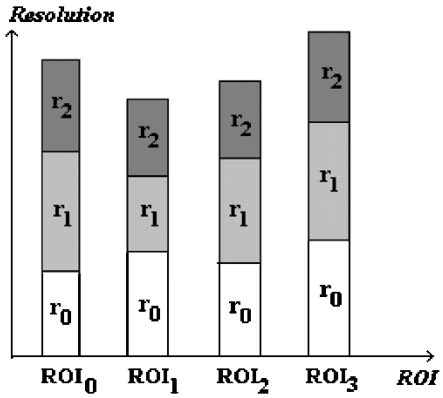

Fig. 7. Codestream structure after codeunits generation and coding.

latter holds those coefficients that have been found significant. For a given threshold n which is successively halved, all the elements in LIS are tested and partitioned until the significant coefficients against n are identified. There are then added to LSP. The elements of LIS are visited in order of size from smallest single coefficient sets first to largest sets last, as suggested in SPECK. A refinement pass is executed on LSP after the significance test procedure. Then each codeunit in an ROI block can generate an embedded bitstream independently. Fig. 7 gives the structure of the codestream generated by coding 12 codeunits, which contains four ROI blocks and three resolution levels.

IV. Optimal Quality Layer Formation

If we decompose an image of the size $M \times N$ at K-level and set the number of resolutions to be held in the final compressed bit stream to be N_R, then $M/2^K \times N/2^K \times N_R$ different bit streams c_i will be generated in the S-SPECK coder according to the codeunits described in Section III. The straightforward method of constructing the overall compressed bit stream 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.

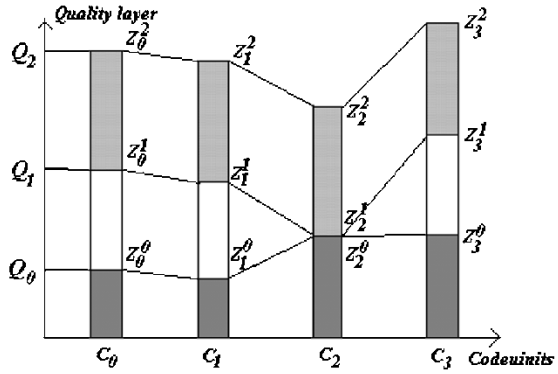

Fig. 8. Quality layer in S-SPECK. Z_i^j denote the truncated points of the bitstream c_i for quality layer Q_j.

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 [3], is used here, as illustrated in Fig. 8, where four codeunits are shown. The wavelet coefficient magnitude distribution will vary among the codeunits, so each will contribute a different number of bits to a quality layer in order to minimize the distortion for a given overall target bit rate. As shown in Fig. 8, the truncation points Z_i^j identify the different contributions from each codeunit to quality layer Q_j. EBCOT utilizes a one-pass bit-rate control method known as PCRD [3] 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. The SPECK variant SBHP [11] simplifies this computation by predicting and estimating the rate-distortion function. 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 in the above section: each codeunit is compressed independently by maintaining its own LIS and LSP. In fact, all the individual LISs and LSPs 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 encountered 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, or 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 the bits output from the encoder into their correspondent codeunit positions in a quality layer during the encoding process. As illustrated in Fig. 9, quality layer Q_j is formatted 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

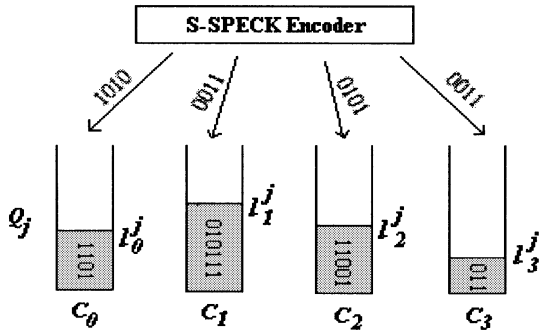

Fig. 9. Bits distribution for a quality layer. l_i^j is the length of bits distributed from the encoder to the bitstream of codeunit c_i in quality layer Q_j.

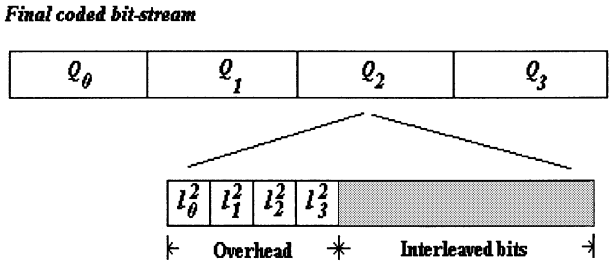

Fig. 10. Structure of interleaved bits in a quality layer.

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.

To delineate different components in a quality layer, some overhead bits are needed for holding the length of the distributed bits in a codeunit, for example l_0^j, l_1^j as shown in Fig. 9. Let H denote the number of overhead bits for each codeunit. The total number of overhead bits for a quality layer, denoted QH, is computed as

$$QH = \frac{M}{2^K} \times \frac{N}{2^K} \times N_R \times H \tag{4}$$

where the size of the original image is $M \times N$, K is the decomposition level, and R is the number of resolution levels. In general, the overhead is negligible and has little effect on the compression performance of S-SPECK. Experiments show that for a typical 512×512 image to be coded into four quality layers at bit rates 0.125, 0.25, 0.5, 1.0, and 2.0, it is enough to set $H = 16$.

For each quality layer, the bits from different codeunits are interleaved one by one to support distortion scalability inside that quality layer. A good example is given in Fig. 10, where quality layer Q_2 consists of the attributions from four codeunits and $l_i^2 (i = 0, \ldots, 3)$ denotes the length of the ith codeunit's attribution. As illustrated in Fig. 10, Q_2 can be truncated at any point as long as the overhead is preserved.

V. Pseudocode of S-SPECK Algorithm

Having described the principles used in the S-SPECK coding method, we are now in a position to understand the actual algorithm in a pseudocode language. The main body of the S-SPECK coding algorithm is presented in pseudocode in Fig. 11. The function *SIGTest(t)* called by the main body of the algorithm is given in Fig. 12. It is worth noting that a I-type set is recursively partitioned by an octave band partitioning scheme, so at some point it will be broken down into three S-type sets, but there will be no new reduced I-type sets. Also, for the nonentropy-coded S-SPECK, as for SPECK, we can save some overhead in bit budget by using the fact that if a set S or I has been found significant and its first three subsets insignificant, then this ensures that the fourth subset is significant and we don't need to send the significance test result of the last subset.

1) **Initialization**
   - Group wavelet coefficients into ROI blocks (Fig. 4).
   - Partition each ROI block into several initial sets (X-type and I-type) according to their resolution levels (Fig. 5), and then add these sets into LIS_c.
   - output $n = \lfloor \log_2(max|c_{i,j}|) \rfloor$
   - set $LSP_c = \varnothing$ and set the index of the current quality layer $q = 0$.
2) **Sorting pass**
   for each element t in LIS_c do
   SIGTest(t)
3) **Refinement pass**
   for each element s in LIP_c except those included in the last sorting pass do
   - distribute the n-th MSB of s to the current quality layer Q_q.
   - if the maximal length of Q_q has been met, then increase q by 1 and prepare to fill next quality layer.
4) **Stepsize updating**
   - decrease n by 1
   - if $n \geq 0$, then goto 2.

Fig. 11. S-SPECK algorithm.

Procedure *SIGTest(t)*

1) Output $\Gamma_n(t)$ to the current quality layer Q_q (Fig. 9).
2) If the maximal length of Q_q has been met, then increase the index q by 1 and prepares to fill next quality layer.
3) If $\Gamma_n(t) = 1$, then
   - Remove t from LIS_c.
   - if t is a S-type set, then
     - if t is a pixel, then
       - Output sign of t to the current quality layer Q_q (Fig. 9) and add t to LSP_c.
       - If the maximal length of Q_q has been met, then increase the index q by 1 and prepares to fill next quality layer.
     - else
       - partition t into four S-type sets t_1, t_2, t_3, t_4 (Fig. 6) and add them to LIS_c.
       - *SIGTest(t_1)*; *SIGTest(t_2)*; *SIGTest(t_3)*; *SIGTest(t_4)*;
   - if t is a X-type set, then
     - partition t into a one-pixel S-type set t_1 and a I-type set t_2 (Fig. 6), then add them to LIS_c.
     - *SIGTest(t_1)*; *SIGTest(t_2)*;
   - if t is a I-type set, then
     - partition t into three S-type sets t_1, t_2, t_3 and a possible empty I-type set t_4 (Fig. 6), then add them to LIS_c.
     - *SIGTest(t_1)*; *SIGTest(t_2)*; *SIGTest(t_3)*; *SIGTest(t_4)*;

Fig. 12. Function SIGTest used by the S-SPECK algorithm.

VI. Entropy-Coding and Complexity Analysis

During the S-SPECK encoding process the significance map can be compressed losslessly using arithmetic coding with simple context-based models, as suggested in [1]. The significance maps are the binary decisions created by the recursive partitioning process. The SPECK variant, EZBC [8], has chosen a complicated context model to obtain more coding gains with increase in complexity. A fast fixed Huffman code is utilized in SBHP [11] for the significance map quadtree coding. A more time-consuming projection technique is proposed in [12] to compress the sign bits. In general, the more complex the entropy coder is, the more coding gains it can achieve, at the cost of substantial increase in complexity. The application will dictate whether the increase in coding performance is worth the added complexity.

In order to strike the best compromise between the complexity and coding performance, in our S-SPECK coder, each codeunit utilizes a simple first-order adaptive arithmetic coder [13] with three independent conditional context models for the sign, refinement, and significance test bits. The first two models use binary alphabets with two 1-bit symbols, and the last one groups the significance test results of the four subsets of set S and I (see Fig. 6) to the four-bit symbols and codes them together. Because the fine scalability properties require the independent arithmetic coding of each codeunit, the frequency of symbols in different context models for that codeunit's arithmetic coder does not converge rapidly due to inadequate samples, which potentially affects the coding efficiency of adaptive entropy coding. We can compensate this negative effect by sharing samples between codeunits in the same ROI block. Consider the fact that, at the decoding side, if a codeunit $C_{r_i}^{\mathrm{ROI}_k}$ in a ROI block ROI_k (see Fig. 5) with the resolution index i is decoded, then all the codeunits $C_{r_j}^{\mathrm{ROI}_k}$ with the resolution indexes $j < i$ in the same ROI block must be decoded, since the reconstructed image should be meaningful for the practical applications. Therefore, at the encoding side, whenever the context models of the arithmetic coders for $C_{r_j}^{\mathrm{ROI}_k}(j < i)$ are updated, the corresponding context models of the arithmetic coder for $C_i^{\mathrm{ROI}_k}$ are also updated using the same samples. Thus, the decoder can duplicate the updating process for $C_{r_i}^{\mathrm{ROI}_k}$ using the available samples of $C_{r_j}^{\mathrm{ROI}_k}(j < i)$. With this sample sharing technique, the context models will have more data to speed up the convergence process. Moreover, most of the redundancy in the bits output from the nonentropy S-SPECK exists locally, i.e., the samples in different ROI blocks are approximately independent. For this reason, the sample sharing technique designed here can dramatically alleviate the negative effect of the inadequate sample problem, which is verified in Table I that compares the compression ratios of the arithmetic coding in SPECK and S-SPECK at various target compression sizes. We can see in Table I that the average loss in the arithmetic compression performance of S-SPECK in terms of the compression ratio is only 0.69%, compared with SPECK. Note that the overheads have been excluded from the target bit sizes.

The expected increase of the computational complexity of the highly scalable coder, S-SPECK, is the result of two new introduced procedures: wavelet coefficients grouping and quality layer formation. The wavelet coefficients grouping process only needs to visit the coefficients one time by the tree structure and can be integrated into the wavelet transform function. As for the quality layer formation procedure that is embedded during the encoding process and does not need the time-consuming computation of the rate-distortion function. It has little negative effect on S-SPECK since it does the formatting work simply by distributing the bits according to their corresponding codeunits.

TABLE I
Comparison of the Arithmetic Coding Performance in S-SPECK and SPECK at Various Target Compression Bit Sizes Using Common Test Images

Test Image	Compression Ratio (%)			
	31.16 kb	63.160 kb	127.93 kb	258.23 kb
S-SPECK				
Bit Rate	0.125 bpp	0.25 bpp	0.5 bpp	1 bpp
Lena	93.90	93.77	94.08	94.31
Barb	95.30	95.71	95.23	95.68
Goldhill	93.71	92.41	93.41	94.41
SPECK				
Bit Rate	0.119 bpp	0.24 bpp	0.49 bpp	0.99 bpp
Lena	92.96	92.96	93.49	93.77
Barb	94.17	95.38	94.97	95.49
Goldhill	92.70	91.38	92.59	93.83

VII. Numerical Results

The following results were obtained with three standard monochrome, 8 bpp, 512×512 images, Lena, Barbara, and Goldhill. We used 7-level pyramids constructed with 9/7-tap filters of [14], and using a "reflection" extension [15] at the image edges. Each image here is coded by S-SPECK into a final coded bitstream containing five quality layers at bit rates 0.125, 0.25, 0.5, 1.0, and 2.0 bpp, and three resolution levels: 512×512, 256×256, and 128×128. Our experiments were conducted for the nonentropy-coded and entropy-coded versions of these image coders. SPECK and S-SPECK were implemented in VC++6.0, and the QccPackSPIHT [16] was used directly. The distortion is measured by the peak signal to noise ratio (PSNR)

$$\mathrm{PSNR} = 10 \log_{10}\left(\frac{255^2}{\mathrm{MSE}}\right) \text{ dB} \tag{5}$$

where MSE denotes the mean squared-error between the original and reconstructed images.

Table II shows the comparison of SPECK and S-SPECK's reconstructed image PSNR performance at rates 0.125, 0.25, 0.5, 1.0, and 2 bpp. The results of nonentropy-coded and entropy-coded versions of these coders are listed in the "N" and "E" columns respectively. The Δ rows give the percentage of PSNR loss of the new coder, compared with SPECK. Table II demonstrates that S-SPECK is quite competitive with SPECK on compression performance. Note that the negative effect of the overheads needed in S-SPECK on its compression performance is a little more pronounced in entropy coding than that in the nonentropy coding.

Tables III and IV compare SPECK and S-SPECK's encoding and decoding times at various bit rates, when they run on a 2-GHz Pentium-4 processor. To get an objective evaluation,

TABLE II
COMPARISON OF SPECK AND S-SPECK'S RECONSTRUCTED IMAGE PSNR AT VARIOUS BIT RATES USING COMMON TEST IMAGES

Coder	PSNR(db)									
	0.125 bpp		0.25 bpp		0.5 bpp		1 bpp		2 bpp	
	N	E	N	E	N	E	N	E	N	E
Lena (512×512)										
SPECK	30.80	31.00	33.77	34.02	36.86	37.08	39.90	40.12	44.09	44.35
S-SPECK	30.65	30.73	33.64	33.80	36.76	36.96	39.85	40.02	44.05	44.25
$\Delta(\%)$	-0.48	-0.87	-0.38	-0.65	-0.27	-0.32	-0.13	-0.25	-0.09	-0.23
Barb (512×512)										
SPECK	24.89	24.98	27.64	27.75	31.32	31.54	36.21	36.47	41.96	42.16
S-SPECK	24.85	24.87	27.48	27.58	31.23	31.38	36.09	36.33	41.91	42.09
$\Delta(\%)$	-0.16	-0.44	-0.58	-0.61	-0.29	-0.51	-0.33	-0.38	-0.12	-0.17
Goldhill (512×512)										
SPECK	28.28	28.43	30.27	30.48	32.74	32.96	36.00	36.26	40.85	41.20
S-SPECK	28.19	28.25	30.17	30.32	32.65	32.83	35.93	36.13	40.78	41.06
$\Delta(\%)$	-0.32	-0.63	-0.33	-0.53	-0.27	-0.39	-0.19	-0.36	-0.17	-0.35

TABLE III
COMPARISON OF SPECK AND S-SPECK'S ENCODING TIMES AT VARIOUS BIT RATES USING COMMON TEST IMAGES

Coder	Encoding Speed(μs per pixel)									
	0.125 bpp		0.25 bpp		0.5 bpp		1 bpp		2 bpp	
	N	E	N	E	N	E	N	E	N	E
Lena (512×512)										
SPECK	0.18	0.24	0.30	0.32	0.48	0.54	0.72	1.04	1.31	1.79
S-SPECK	0.24	0.30	0.35	0.48	0.54	0.72	0.84	1.19	1.49	2.09
$\times$	1.33	1.67	1.17	1.50	1.13	1.33	1.17	1.14	1.14	1.17
Barb (512×512)										
SPECK	0.24	0.24	0.30	0.36	0.42	0.60	0.72	1.01	1.31	1.91
S-SPECK	0.30	0.30	0.36	0.48	0.52	0.72	0.83	1.13	1.49	2.09
$\times$	1.67	1.67	1.20	1.33	1.24	1.20	1.15	1.12	1.14	1.09
Goldhill (512×512)										
SPECK	0.24	0.30	0.30	0.42	0.42	0.65	0.77	1.07	1.31	1.85
S-SPECK	0.30	0.35	0.36	0.48	0.53	0.77	0.84	1.19	1.43	2.03
$\times$	1.67	1.17	1.20	1.14	1.26	1.18	1.09	1.11	1.09	1.10

TABLE IV
COMPARISON OF SPECK AND S-SPECK'S DECODING TIMES AT VARIOUS BIT RATES USING COMMON TEST IMAGES

Coder	Decoding Speed(μs per pixel)									
	0.125 bpp		0.25 bpp		0.5 bpp		1 bpp		2 bpp	
	N	E	N	E	N	E	N	E	N	E
Lena (512×512)										
SPECK	0.12	0.12	0.18	0.18	0.30	0.36	0.60	0.72	1.25	1.43
S-SPECK	0.18	0.18	0.24	0.35	0.42	0.60	0.71	1.07	1.43	2.03
$\times$	1.50	1.50	1.33	1.94	1.40	1.67	1.18	1.42	1.14	1.42
Barb (512×512)										
SPECK	0.12	0.12	0.18	0.18	0.36	0.35	0.60	0.71	1.20	1.43
S-SPECK	0.18	0.18	0.24	0.35	0.42	0.60	0.71	1.01	1.43	2.09
$\times$	1.50	1.50	1.33	1.94	1.17	1.71	1.18	1.42	1.19	1.46
Goldhill (512×512)										
SPECK	0.12	0.12	0.18	0.18	0.36	0.36	0.66	0.77	1.24	1.49
S-SPECK	0.18	0.24	0.24	0.35	0.48	0.60	0.77	1.13	1.49	2.08
$\times$	1.50	2.00	1.33	1.94	1.33	1.67	1.17	1.47	1.20	1.40

we tried 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 Tables III and IV. From these ratios, we can see that both of S-SPECK's encoding and decoding speeds are close to those of SPECK.

Comparative evaluations of the new coder S-SPECK in respect to the benchmark coder SPIHT are illustrated in Fig. 13(a) (nonentropy-coded) and Fig. 13(b) (entropy-coded). Clearly, at different bit rates for the standard test images, S-SPECK is quite competitive with SPIHT, particularly in the case of nonentropy coding. Fig. 14 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 rate 0.5 bpp.

Fig. 15 illustrates an example of S-SPECK's ROI retrievability scalability, in which the same region of the Lena image

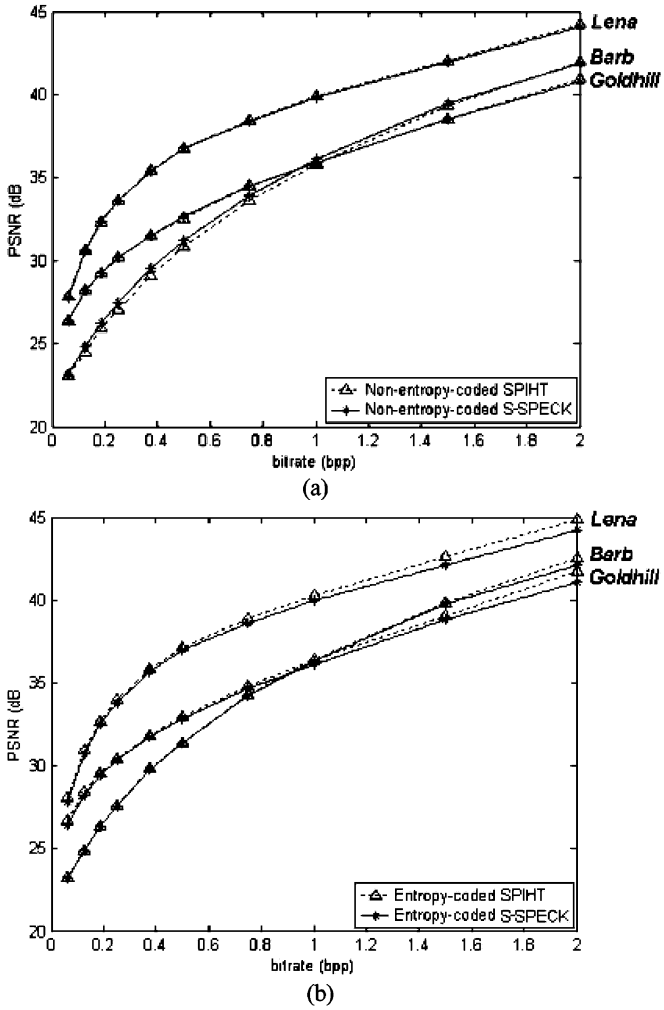

Fig. 13. Comparison of S-SPECK and SPIHT's PSNR performance at 0.0625, 0.125, 0.1875, 0.25, 0.375, 0.5, 0.75, 1.0, 1.5, and 2.0 bpp. (a) Non-entropy coding. (b) Entropy coding.

Fig. 14. Reconstructed images of three resolution levels at target bit rate 0.5 bpp.

is retrieved from the coded bitstream at bit rates of 0.125, 0.25, 0.5, and 1.0 bpp.

VIII. Conclusion

We described a new wavelet-transform-based image coder, S-SPECK, which extends the original coder, SPECK, successfully to a highly scalable scheme. S-SPECK not only supports distortion scalability, resolution scalability, and retrievability,

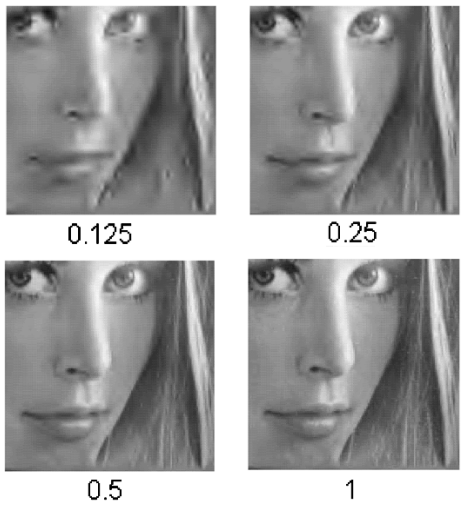

Fig. 15. Region retrieved from various bit-rate coded bitstreams at 0.125, 0.25, 0.5, and 1.0 bpp.

but also achieves excellent compression performance with very low computational complexity. In S-SPECK, wavelet coefficients are grouped to codeunits according to their relationship with ROIs and resolution levels. A zeroblock coder similar to SPECK is then incorporated to code these codeunits based on a combined LIS and LSP. Each coded bit output from the zeroblock encoder is distributed to a quality layer during the encoding process. This quality layer formatting method is simple and efficient, compared with the time-consuming PCRD method used in JPEG2000. Extensive experiments showed that the loss of S-SPECK's compression performance and computational speed is negligible compared with SPECK. It will be advantageous to apply S-SPECK to modern multimedia applications on the Internet.

References

[1] 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.*, no. 11, pp. 1219–1235, Nov. 2004.

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

[3] D. Taubman, "High performance scalable image compression with ebcot," *IEEE Trans. Image Process.*, vol. 9, no. 7, pp. 1158–1170, Jul. 2000.

[4] J. Shapiro, "Embedded image coding using zerotrees of wavelet coefficients," *IEEE Trans. Signal Process.*, vol. 41, no. 12, pp. 3445–3462, Dec. 1993.

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

[6] D. Taubman and M. Marcellin, *JPEG2000: Image Compression Fundamentals, Standards, and Practice*, 2nd ed. Norwell, MA: Kluwer, 2002.

[7] C. C. A. Skodras and T. Ebrahimi, "The JPEG2000 still image compression standard," *IEEE Signal Process. Mag.*, vol. 18, no. 9, pp. 36–58, Sep. 2001.

[8] Highly scalable subband/wavelet image and video coding, S.-T. Hsiang. (2002, Jan.). [Online]. Available: http://www.cipr.rpi.edu/hsiang/

[9] Trellis source coding and memory constrained image coding, F. Wheeler. (2000, Dec.). [Online]. Available: http://www.cipr.rpi.edu/wheeler/

[10] H. Man, F. Kossentini, and M. Smith, "A family of efficient and channel error resilient wavelet/subband image codecs," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 9, no. 2, pp. 95–108, Feb. 1999.

[11] C. Chrysafis, A. Said, A. Drukarev, A. Islam, and W. Pearlman, "SBHP—A low complexity wavelet coder," in *Proc. IEEE Int. Conf. Accoustics, Speech, and Signal Processing*, Jun. 2000, pp. 2034–2038.
[12] T. A. Deever and S. H. Sheila, "Efficient sign coding and estimation of zero-quantized coefficients in embedded wavelet image codecs," *IEEE Trans. Image Process.*, vol. 12, no. 4, pp. 421–431, Apr. 2003.
[13] A. Moffat, R. Neal, and I. Witten, "Arithmetic coding revisited," in *ACM trans. Inf. Syst.*, vol. 16, Jul. 1998, pp. 256–294.
[14] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies, "Image coding using wavelet transform," *IEEE Trans. Image Process.*, vol. 1, no. 4, pp. 205–220, Apr. 1992.
[15] B. Usevitch, "A tutorial on modern lossy wavelet image compression: Foundations of jpeg2000," *IEEE Signal Process. Mag.*, vol. 18, no. 9, pp. 22–35, Sep. 2001.
[16] (2004) The SPIHT coder in the Qccpack library. [Online]. Available: http://qccpack.sourceforge.net/

Gui Xie (S'04) received the B.Eng. and M.Eng. degrees from the School of Computer Science, Wuhan University, Wuhan, China, in 1997 and 2000. He is currently working toward the Ph.D. degree in the Graduate School of Information Science, Japan Advanced Institute of Science and Technology, Ishikawa, Japan.

His research interests mainly focus on image coding and watermarking.

Hong Shen received the B.Eng. degree from Beijing University of Science and Technology, Beijing, China, the M.Eng. degree from the University of Science and Technology of China, Hefie, China, and the Ph.Lic. and Ph.D. degrees from Abo Akademi University, Turku, Finland, all in computer science.

He is currently a Full Professor in the Graduate School of Information Science, Japan Advanced Institute of Science and Technology, Ishikawa, Japan. Previously, he was a Professor at Griffith University, Brisbane, Australia. He has published over 200 technical papers on algorithms, parallel, and distributed computing, interconnection networks, parallel databases and data mining, multimedia systems, and networking.

Dr. Shen has served as an Editor of *Parallel and Distributed Computing Practice*, Associate Editor of the *International Journal of Parallel and Distributed Systems and Networks*, a member of the editorial boards of *Parallel Algorithms and Applications, International Journal of Computer Mathematics*, and the *Journal of Supercomputing*, and chaired various international conferences. He is a recipient of the 1991 National Education Commission Science and Technology Progress Award and the 1992 Sinica Academia Natural Sciences Award.