Progressive Significance Map and Its Application to Error Resilient Image Transmission

Yang Hu, William A. Pearlman*, and Xin Li

***Abstract*—Set partition coding (SPC) has shown tremendous success in image compression. Despite its popularity, the lack of error resilience remains a significant challenge to the transmission of images in error-prone environments. In this paper, we propose a novel data representation called progressive significance map (prog-sig-map) for error-resilient SPC. It structures the significance map (sig-map) into two parts: a high-level summation significance map (sum-sig-map) and a low-level complementary significance map (comp-sig-map). Such a structured representation of the significance map allows us to improve its error resilient property at the price of only a slight sacrifice in compression efficiency. For example, we have found a fixed length coding of the comp-sig-map in the prog-sig-map renders $64\%$of the coded bit-stream insensitive to bit errors, compared to 40% with the conventional significance map. Simulation results have shown that the prog-sig-map can achieve highly competitive rate-distortion performance for binary symmetric channels, while maintaining low computational complexity. Moreover, we note that prog-sig-map is complementary to existing independent-packetization and channel-coding based error-resilient approaches and readily lends itself to other source coding applications such as distributed video coding.**

***Index Terms*—Error resilience, significance map, set partition coding (SPC), SPIHT, image transmission.**

I. Introduction

Set partition coding (SPC) has been widely studied in image and video compression. From Shapiro's pioneering work on embedded zerotree wavelet (EZW) coding [1] to the JPEG 2000 [2], [3] image compression standard and the new 2D lossy-to-lossless compression algorithm recently standardized by the Consultative Committee for Space Data Systems (CCSDS) [4], [5], SPC has served as a key component to the coding efficiency of wavelet-based approaches. SPC has also been successfully applied into several other well-known image coding algorithms including SPIHT [6], SPECK [7], SWEET [8], SBHP [9] and EZBC [10]. As an effective way of representing image data in the wavelet domain, SPC (sometimes combined with other entropy coding methods, such as arithmetic coding) takes advantage of the characteristics of wavelet transforms. Additionally, SPC can easily support desirable features such as precise rate control and progressive transmission. A comprehensive tutorial on SPC and its usage in wavelet coding systems can be found in [11], [12].

Y. Hu and W. A. Pearlman are with the Department of Electrical, Computer, and Systems Engineering, Rensselaer Polytechnic Institute, Troy, NY 12180, USA; Tel: 518-276-6082; Fax: 518-276-8715; E-mail: yanghuyh@gmail.com, pearlw@ecse.rpi.edu.

X. Li is with the Lane Department of Computer Science and Electrical Engineering, West Virginia University, Morgantown, WV 26506-6109, USA; E-mail: Xin.Li@mail.wvu.edu.

The salient feature of SPC is that its compressed bitstream divides naturally into two components: a significance map that conveys location information; and a value bitstream that conveys intensity information of signs and lower order bits of wavelet coefficients. Despite the nice properties of SPC, a widely known weakness of this technique is its sensitivity to channel errors. A single bit error in the significance map can cause a catastrophe in image reconstruction due to error propagation. To overcome this weakness, a number of error resilient coding techniques have been proposed in the literature. Forward error correction (FEC), initially proposed by Sherwood and Zeger [13], [14], is among the first joint source/channel coding schemes using concatenated code, where an outer cyclic redundancy check (CRC) code detects errors and an inner rate compatible punctured convolutional (RCPC) code corrects as many errors as possible. The FEC achieves good performance for a class of binary symmetric channels (BSC) with known error rates. Then it is developed for burst-error-prone channels with more powerful channel codes, such as in [15] with Reed-Solomon code and in [16] with Turbo code. Further improved error resilience is obtained through combining FEC with unequal error protection (UEP) techniques (e.g., in [17], [18]), where more important data are protected by stronger channel codes.

All of the aforementioned techniques employing FEC produce error resilience at the expense of substantial increase in channel transmission rate. Another class of techniques attempts to build natural error resilience into the compressed bitstream without resorting to FEC. Such techniques, which have been adopted by JPEG2000 and the new CCSDS image compression standard, partition the data into groups and encode each group into packets that can be decoded independently [19], [20], [21], [22]. Thanks to independent packetization, error propagation is limited to the packets within which channel errors occur. Error resilient entropy coding (EREC) [23] can be used to reorganize these variable-length packets into fixed-length data slots before multiplexing and transmission. Therefore, the synchronization of the start of each packet can be automatically obtained at the receiver without synchronization words. Self-synchronizing variable length coding [24] is a mechanism that limits the effects of an error to a small portion of the data. Typically, the natural error resilience techniques manage to stop the error propagation at the expense of small loss in coding efficiency. There exist hybrid schemes that protect naturally error resilient bitstreams with FEC. For example, [25], [26] apply channel codes to independently coded packets to further improve the robustness against both random error and packet erasure.

A fundamental issue related to error resilience of SPC is the synchronization at the decoder. Since two levels of uncertainty (location vs. intensity) are sequentially resolved by SPC as described in Sec. II, it is highly desirable to extend the strategy of SPC such that random errors associated with location decoding and intensity decoding do not propagate. In this paper, we propose a novel extension of SPC called *progressive significance map* (prog-sig-map) with the desirable natural error-resilient property. It is a new data representation consisting of a *summation significance map* (sum-sig-map) and a *complementary significance map* (comp-sig-map). The sum-sig-map conveys the high-level information of significance tests, which dictates coding/decoding of the comp-sig-map at the lower level. Unlike the conventional significance maps (sig-maps), such progressive representation allows us to code the comp-sig-map using fixed-length-codes; consequently, the decoding errors (in terms of bit flips) associated with comp-sig-map would not propagate within the comp-sig-map or to the decoding of sum-sig-map. To the best of our knowledge, this is the first time that error resilience is achieved within the core framework of SPC itself and without noticeable coding efficiency loss (typically about 2% to 3%). With proper design, the proposed prog-sig-map can be jointly used with other error resilience techniques (such as independent packetization, FEC, UEP, EREC, etc.) to further improve the robustness.

We note that Meany and Martens presented an error resilient entropy coding method in [27], bearing some resemblance to ours, where they separated the individual codewords into two fields: a greater protected variable-length prefix and an error-resilient fixed-length suffix. Given a correct prefix, the associated suffix is error-resilient. However, this so-called split field coding is built on some parametric coding methods, for which the codewords include fixed length suffixes (such as Rice coding [28] or start-step coding [29]). In contrast, our method can build its own variable length prefix (the sum-sig-map) and error-resilient fixed-length suffix (the comp-sig-map). In image compression, split field coding is applied to codewords for coefficient values, but not to positional codewords (such as the significance map), while our method is aimed at more error-sensitive positional information.

The rest of the paper is organized as follows. We briefly review conventional significance map and introduce a new conception of the significance test in terms of (c, w)-test in Sec. II. Based on the new conception, we present the prog-sig-map in Sec. III and illustrate it with a coding example. We analyze two classes of error propagation patterns which help explain the improved error resilience of prog-sig-map. We also provide a simple proof that the entropy of the prog-sig-map equals that of the conventional sig-map. Simulation results on coding efficiency, error performance, and computational complexity are given in Sec. IV. In Sec. V, we discuss some possible applications of the proposed coding technique. Sec. VI concludes the paper.

II. Set Partition Coding

In this section, we provide a brief overview of SPC. SPC starts from the highest bit-plane indexed by $n = n_{max} = \lfloor \log_2(\max_l |X^l|) \rfloor$, where X^l is a coefficient at location l, and descends in unit increments to successively lower bit-planes. When the magnitudes of all coefficients in a set are less than the threshold $T = 2^n$ of the n-th bit-plane, the set is deemed *insignificant* and indicated by bit 0; otherwise, the set is declared *significant* and indicated by bit 1. This comparison operation is called significance test or sig-test. When SPC is applied to the wavelet transform of an image, all sets are initially treated as insignificant and stored in an ordered list LIS (list of insignificant sets). Starting with the n_{max}-th bit-plane, the sig-test is performed on every set of the LIS. If a set is found to be significant, it is partitioned into several subsets according to the pre-defined set partitioning rule. These subsets are further tested and multi-element sets found to be significant are again partitioned until the significant coefficients are isolated, as illustrated by the quadtree example in Fig. 1. For each located significant coefficient, the sign is coded and sent out. Processing all LIS elements in this way at a given threshold completes the *sorting pass*. Upon completion of a sorting pass, a *refinement pass* typically follows to gather magnitude bits of significant coefficients[1]. Proceeding to lower thresholds (smaller bit-plane indices n), SPC alternates between sorting pass and refinement pass [6] to progressively resolve the location and intensity uncertainty of significant coefficients.

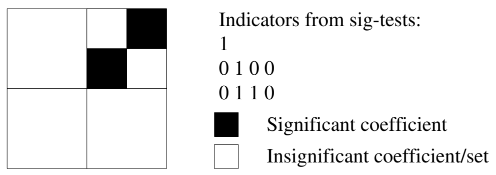


Fig. 1. The quadtree partitioning example of a 4×4 coefficient block. The sig-test of the whole 4×4 block outputs indicator 1 (the block is significant), and then partition the block into four 2×2 sub-blocks. The sig-test of the four 2×2 blocks outputs indicators 0100 according to the zig-zag scanning order (the top-right 2×2 block is significant), and the significant 2×2 block is further partitioned into four single-element sub-blocks. Finally, the sig-test of the four single-element blocks outputs indicators 0110, and the two significant coefficients (shaded) are located. The raw (uncoded) conventional sig-map consists of the indicators directly.

The sig-map associated with the sig-tests often occupies the majority of the coded bitstream. For example, in typical lossy image coding, arithmetic-coded conventional sig-map takes about 60% of the whole bitstream at 30 dB or higher recovered PSNR [30]. The sig-map is unfortunately sensitive to channel errors. It should be noted that error resilience and coding efficiency represent two conflicting goals. For example, fixed-length codes often have better error-resilience property than variable-length codes, but are worse on coding efficiency. The challenge lies in how to achieve a good tradeoff between these two such that the end-to-end rate-distortion performance can be improved. The motivation behind this work is similar to that of data partitioning - think of groups instead of coefficients

[1]A refinement pass in each bit-plane is not really part of the SPC process. Alternatively, when a significant coefficient is located, its sign and all its lower order bits may be coded and sent out immediately.

- but we opt to implement it by a novel data structure within the context of SPC. The subsets generated by set partitioning are called a *set group*. Assume that a set group has c_n sets and w_n of them are found significant in the sig-test of bit-plane n. In the sig-test of the next lower $(n-1)$ bit-plane, it follows from the SPC rule that

$$c_{n-1} = c_n - w_n, \tag{1}$$

because only the remaining insignificant sets are tested for significance again in the next lower bit-plane.

The above observation motivates us to introduce a new conception of the significant test, which we call the (c, w)-test throughout this paper. The c denotes the number of *candidates*, defined as the sets in the set group to be sig-tested; while w denotes the number of *winners*, defined as the newly found significant sets in the set group. For example, the sig-test of the four subgroups from a quadtree partition is called a $(4, w)$ test, where the w is the number of significant subgroups. It is not difficult to see that the conventional sig-map contains the results of a series of correlated (c, w) tests. More specifically, if we assign a binary indicator I, called *winner indicator*, to signal whether or not a candidate is a winner,

$$I = \begin{cases} 1, & \text{if the candidate is a winner;} \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

then a (c, w)-test generates c indicators $\{I_1, I_2, ..., I_c\}$ satisfying

$$w = \sum_{i=1}^{c} I_i. \tag{3}$$

Intuitively, such (c, w)-test can be seen as a group transformation of the sig-map to facilitate error-resilient coding, as we will elaborate next.

III. Progressive Significance Map

Based on the newly-defined (c, w)-test, we present the novel structure of prog-sig-map and its improved error resilient property. The key idea is to protect the synchronization of encoded bit stream by structuring the significance information into two parts: a high-level sum-sig-map and a low-level comp-sig-map.

A. Sum-Sig-Map

For a sig-test with c candidates, the sum-sig-map conveys how many of the c candidates are winners. The number of winners w is actually the sum of the c winner indicators in the conventional sig-map as in (3). If $w = 0$ or $w = c$, the c winner indicators must be all 0's or all 1's, respectively. If $0 < w < c$, there is ambiguity, because $\binom{c}{w}$ different choices exist for the w winners out of the c candidates. Therefore additional information (comp-sig-map) is needed to recover the c winner indicators correctly, which will be addressed in the next subsection.

When a group of sets is initially generated from set partitioning, the candidate number c is decided by the specific algorithm and is known by the decoder without error. For the set group in the following sorting passes, the candidate number c is inferred from the previous sig-test of the group according to (1). So the c values depend on the w values only. The sum-sig-map therefore conveys the coding architecture, comprising three parts:

P-1) The number w of partitioning operations needed after each (c, w) test of a group of multi-element sets.
P-2) The number w of significant coefficients located after each (c, w) test of a group of single-element sets.
P-3) The candidate number c for the sig-tests in the next sorting pass.

Correct P-1 and P-3 maintain the map synchronization, and correct P-2 keeps the value synchronization. Details on map and value synchronizations are described in Sec. III-C.

B. Comp-Sig-Map

The comp-sig-map assists in unambiguous recovery of the c winner indicators of a sig-test, given the known c and w satisfying $0 < w < c$. For $c \in \{1, 2, 3, 4\}$[2], there are six qualified (c, w) pairs: $(2, 1)$, $(3, 1)$, $(3, 2)$, $(4, 1)$, $(4, 2)$, and $(4, 3)$. To clarify the ambiguity left by the sum-sig-map, the comp-sig-map need to indicate which one of the $\binom{c}{w}$ indicator combinations is the correct one. This process is progressive, because each complementary bit precludes some impossible indicator combinations. Different expressions can be used for the complementary information.

1) Variable-Length Comp-Sig-Map: Assume that the $\binom{c}{w}$ indicator combinations of a (c, w) test approximately have equal probabilities, which is true for most sig-tests of the popular set partition methods[3]. The Huffman codes are adopted in the variable-length comp-sig-map, by means of Huffman coding a set of $\binom{c}{w}$ equally probable symbols. The $(2, 1)$, $(4, 1)$, and $(4, 3)$ sig-tests have 2, 4, and 4 equally probable symbols respectively. Their Huffman codes are of length 1, 2, and 2 bits respectively, which agree with the entropy (1, 2, and 2 bits per symbol) of these sig-tests. Each of $(3, 1)$ and $(3, 2)$ tests has 3 equally probable symbols, having entropy $\log_2 3$ bits per symbol. The corresponding Huffman codes include one 1-bit codeword and two 2-bit codewords. The $(4, 2)$ test has 6 equally probable symbols, having entropy $\log_2 6$ bits per symbol. Its Huffman codes include two 2-bit codewords and four 3-bit codewords. Arithmetic coding (AC) [32] can be applied to reduce the compression bit rate further.

To facilitate understanding, we map the Huffman codewords (i.e., the complementary bits or comp-bits) and the indicator combinations according to Fig. 2. For example, for the $(3, 1)$ test in Fig. 2(b), the first comp-bit is just the first indicator. If it is 1, the three indicators must be "100" and, thus, no more comp-bits are needed. If it is 0, the second indicator serves as the second comp-bit, and then all three indicators can be inferred unambiguously. A 1 indicates 0, while a 0 indicates

[2]There is a maximum of four sig-test candidates in most SPC methods for 2D image coding. However, more candidates can exist in some other scenarios. For example, in 3-D SPIHT [31] for video coding, the maximum candidate number is eight. To present the new sig-map without being redundant, we will consider $c \in \{1, 2, 3, 4\}$, but it can be easily extended to the cases with more candidates.

[3]For octave band partition, the three $\mathcal{S}$ subsets typically have higher significance probability than the $\mathcal{I}$ subset as analyzed in [7].

1 for the third symbol. Actually, after the first comp-bit 0, the decision for the last two indicators is the same as the $(2,1)$ test in Fig. 2(a). For $(4,1)$ and $(4,3)$ tests, the first comp-bit is the XOR (exclusive or) result of the first two indicators, and the second comp-bit is decided as in the $(2,1)$ test.

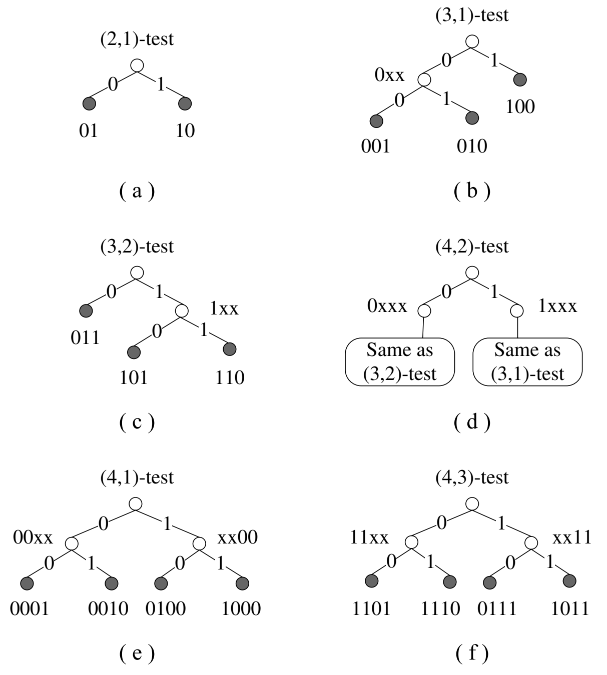


Fig. 2. Huffman codewords (comp-bits) generating for variable-length comp-sig-map (comp-bits on edges, indicator combinations on nodes).

2) Fixed-Length Comp-Sig-Map: Under fixed length coding, the number of comp-bits for a (c,w) test is fixed to $\lceil \log_2 M \rceil$, where $M = \binom{c}{w}$. For $(3,1)$, $(3,2)$, and $(4,2)$ tests. Additional bits are appended to the shorter codewords to make them the same length as the longer ones. We still take the $(3,1)$ test as an example. For the indicator combination "100", one bit is added after its comp-bit "1", making its comp-sig-map the same length of 2 bits as those of the other two indicator combinations. The appended bit could be 0 or 1 based on the agreement between the encoder and the decoder; or the appended bits could be utilized by other modules of a coding system, such as FEC, error-check, etc.

C. Information Theoretic Interpretation and Error Sensitivity Analysis

To gain deeper insight into the proposed progressive significance map, we will offer an information theoretic interpretation and analyze its error sensitivity in this subsection. Let $H(T)$ denote the entropy of a sig-test T. The random variables C, W, and I_i denote the candidate number, winner number, and winner indicator, respectively, satisfying $W = \sum_{i=1}^{C} I_i$. Because C is inferrable from previous sig-tests (according to Eq. (1)), we have

$$\begin{aligned} H(T) &= H(I_1, I_2, ..., I_C|C) && (4)\\ &= H(I_1, I_2, ..., I_C, W|C) \\ &= H(W|C) + H(I_1, I_2, ..., I_C|C, W) && (5) \end{aligned}$$

The expression in (4) corresponds to the conventional sig-map, while that in (5) is associated with the prog-sig-map. The $H(W|C)$ and $H(I_1, I_2, ..., I_C|C, W)$ are the entropies of the sum-sig-map and comp-sig-map, respectively. Therefore, in theory, the layered structure of the prog-sig-map introduces no penalty on coding efficiency compared with the conventional one.

What makes the prog-sig-map more attractive is its improved error resilience property. Depending on the nature of the errors, we can classify them into two types: globally damaging and locally damaging. They cover all possible error patterns caused by bit flips.

1) Globally Damaging Errors: In the first type, channel errors change the number of "1"-indicators in a sig-test — i.e., the w value of a (c,w) test is changed. This error pattern results in serious collapse at the decoder. The simplest case is a single bit error in the sig-test of a group of single-element sets (that cannot be further partitioned). Assuming that an indicator "0" is erroneously decoded to be "1", as in the example of Fig. 3(a), the (c,w) test becomes $(c,w+1)$-test. The decoder will read in and recover one more significant coefficient, which originally should be read in and recovered after some later sig-test. Thus every following significant value/coefficient will be recovered to a wrong location. This situation reflects the loss of "value synchronization". In the next sorting pass, there originally should be $c-w$ (2 in the example) candidates in the group (the top right 2×2 coefficients in the example) to be sig-tested, jointly or separately depending on the specific algorithm, but only $c-(w+1)$ candidates left due to the previous bit error. The indicators originally belonging to the sig-test of this set group will go to some later sig-tests, and consequently every following indicator will be interpreted to a wrong candidate. This phenomenon is referred to as loss of "map synchronization". Therefore, the global-damaging error on the sig-test of single-element sets will result in the loss of map synchronization from the next sorting pass while the loss of value synchronization happens immediately.

Another case is a single bit error on the sig-test of a group of multi-element sets (that can be further partitioned if significant). Still assuming that an indicator "0" is erroneously decoded to be "1", as in the example of Fig. 3(b), the (c,w) test becomes $(c,w+1)$-test. In this case, the decoder needs to read in more indicators (four more indicators in the example), which should originally be read in for the later tests. As a consequence, every following sig-test will use indicators originally belonging to later tests, so map synchronization is lost immediately. The value synchronization will be lost when the recursively performed sig-test comes to the sets with single elements. In short, the global-damaging error on the sig-test of multi-element sets will result in loss of map synchronization immediately and value synchronization soon thereafter.

2) Locally Damaging Errors: In the next type, the bitstream has changed due to channel errors, but the number of "1"-indicators in the sig-test is kept the same. For example, as shown in Fig. 4(a), the correct number ($w = 2$ in the example) of "1"-indicators is maintained. Significant coefficients may be recovered at the wrong places, but the errors are localized within the current tested group (the top right 2×2 block in

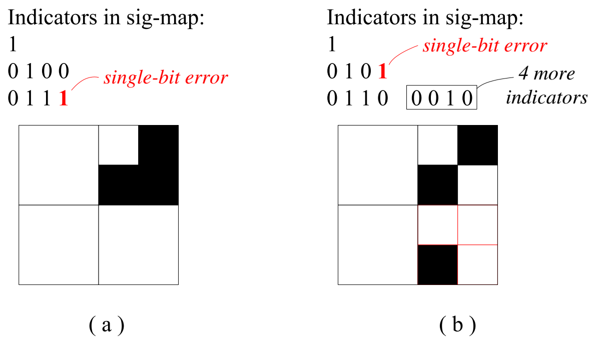


Fig. 3. The quadtree partitioning example with globally damaging error on the sig-test of (a) single-element sets and (b) multi-element sets.

the example) and do not propagate to the next coding-pass.

If this error pattern happens in the sig-test of a group of multi-element sets, as illustrated in Fig. 4(b), the correct number ($w = 1$ in the example) of "1" indicators is maintained. Although partitioning will be performed on some wrong set, the erroneous partitioning will be limited to the current tested group (limited to the four 2×2 blocks in the example) and the correct number of indicators will be read in for the following sig-tests. Still correct $c - w$ candidates will be left for the next sorting pass and the correct number of significant values will be recovered although to wrong places within the current tested sets (the four 2×2 blocks in the example). In summary, locally damaging errors are less catastrophic because error propagation is avoided by preserving synchronization.

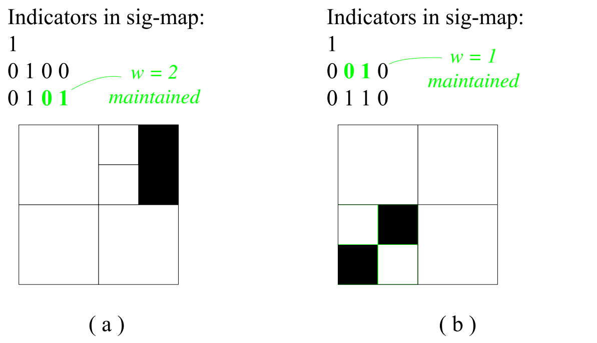


Fig. 4. The quadtree partitioning example with locally damaging error on the sig-test of (a) single-element sets and (b) multi-element sets.

In the proposed sig-map, a correct sum-sig-map can prohibit the occurrence of globally damaging errors. The reason is that if the sum-sig-map is correct, it will produce the correct winner numbers w. In the decoding of a sig-test, the indicators are recovered under the constraint of the correct (c, w) and indicated by comp-bits. Assuming error(s) in the comp-sig-map, erroneous indicators will be resulted. However, the number of '1'-indicators will still equal the correct w. Such indicators will lead to local damage only.

D. A Coding Example

To facilitate our understanding, we shall illustrate the coding procedure of the prog-sig-map through the toy example shown in Fig. 1. The coding begins with a $(1, 1)$ test, followed by a $(4, 1)$ test and a $(4, 2)$ test. The raw (uncoded) conventional sig-map carries the indicators directly as

1	...	indicator from the (1,1) test
0100	...	indicators from the (4,1) test
0110	...	indicators from the (4,2) test

While the prog-sig-map is formed as below.

1	...	w value of the (1,1) test;
		compose the sum-sig-map;
		no comp-sig-map needed.
1	...	w value of the (4,1) test;
		compose the sum-sig-map.
10	...	comp-bits of the (4,1) test (Fig. 2(e));
		compose the comp-sig-map;
2	...	w value of the (4,2) test;
		compose the sum-sig-map.
011	...	comp-bits of the (4,2) test (Fig. 2(d));
		compose the comp-sig-map;

With the prog-sig-map, the decoder infers the c number from previous sig-tests and receives the w number (assumed error free) from sum-sig-map. On knowing c and w, the decoder will recover the c indicators based on the later received comp-bits, according to the mapping rules in Fig. 2 (or its fixed length version).

For the example in Fig. 1, if the comp-sig-map of the $(4, 1)$ test has bit error(s), the received comp-sig-map could be 00, 01, or 11. On knowing $c = 4$ from the previous sig-tests and $w = 1$ from the received sum-sig-map, the decoder will recover the 4 indicators according to 2(e). The recovered indicator combination could be 0001, 0010, or 1000, corresponded to the 00, 01, and 11 comp-sig-map, respectively. All of them satisfy the locally-damaging error pattern, because the number of "1" indicators remains unchanged. Thus coding synchronization is still maintained and only local damage (within the four 2×2 blocks) results.

Assume a fixed length comp-sig-map is used. If the comp-sig-map of the (4,2) test has bit error(s), there are seven ($2^3 - 1 = 7$) possible received comp-sig-maps. Among them five (001, 010, 100, 101, and 110) can be mapped back to some indicator combination satisfying locally-damaging error pattern, i.e., with unchanged number of "1" indicators. The other two (000 and 111) possible comp-sig-maps are *illegal* comp-bit sequences. On receiving them, the error can be detected immediately and further steps (retransmission, error concealment, etc.) can be adopted. Since these steps are out of the scope of this paper, we just randomly assign a *legal* sequence for the comp-sig-map whenever the decoder receives an illegal one in our simulations.

In a word, under the constraint of a correct sum-sig-map (or equivalently w), the globally damaging errors can be prevented. Errors in the comp-sig-map will result in local damage only, which is less catastrophic. It should be noted that bit errors in the raw (uncoded) value data – signs and refinement bits – impact only the values of the specific coefficients corresponding to the error bits, so the damage is even milder.

IV. Simulation Results

To evaluate the performance of the proposed prog-sig-map, we have incorporated it into the well-known SPIHT [6] codec in our simulations[4]. We have chosen the reversible 5/3 filters [33] to implement the wavelet transform with three levels of decomposition (other popular filters such as Daubechies' 9/7 were also tested and similar conclusions can be drawn)[5]. Two standard 8-bit greyscale test images: $Lena$ and $Baboon$ size 512×512 and two 8-bit greyscale JPEG2000 test images: $Woman$ and $Cafe$ size 2048×2560 were used in our coding experiments. The coding bit rate is measured by bits per pixel (bpp); the objective image quality is measured with PSNR in dB, defined as PSNR $= 10\log_{10}(\frac{255^2}{MSE})$ for 8-bit images, with MSE denoting mean squared error. There are three sig-map strategies in the simulations.

- Conventional (AC) sig-map: arithmetic-coded conventional sig-map.
- Progressive (AC) sig-map: prog-sig-map consisting of arithmetic-coded sum-sig-map and arithmetic-coded variable-length comp-sig-map.
- Progressive (fixed) sig-map: prog-sig-map consisting of arithmetic-coded sum-sig-map and uncoded fixed-length comp-sig-map.

A. Performance on Coding Efficiency

We first report the comparison of coding efficiency between the prog-sig-map and the conventional one. The codec with conventional (AC) sig-map is used as the reference, and the length of the sig-map (or the whole bitstream) in other coding strategies is normalized with respect to this benchmark. In Fig. 5 we plot the normalized lengths of the arithmetic-coded sum-sig-map versus recovery PSNR for these four test images. We can see that the size of the sum-sig-map is about 60% of that of the conventional (AC) sig-map.

Fig. 6 demonstrates the whole prog-sig-map normalized by the conventional (AC) sig-map. The lower four curves (solid-line) demonstrate the progressive (AC) sig-map. Its size is similar to that of the conventional (AC) sig-map for recovery PSNR above 30 dB. For lower recovery PSNR of $Lena$, the prog-sig-map performs even better. This verifies that the new and the conventional sig-maps convey the same information and thus have similar sizes after properly designed entropy coding. The higher four curves (dashed-line) demonstrate the progressive (fixed) sig-map. It has a 3% to 5% penalty for moderate to high recovery PSNR, mainly due to the appended bits in the fixed-length comp-sig-map. This progressive (fixed)

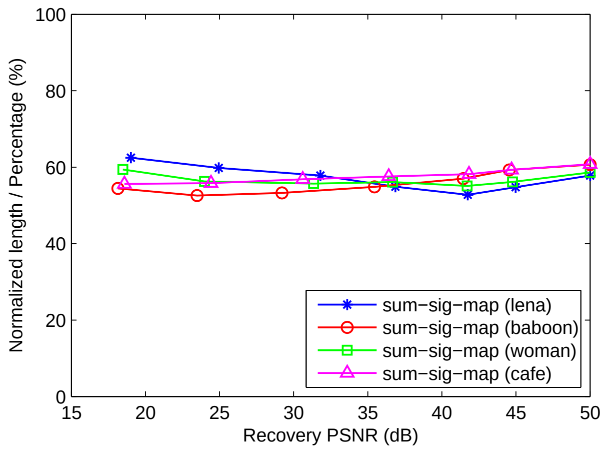


Fig. 5. The length of the arithmetic-coded sum-sig-map normalized by the length of the conventional (AC) sig-map.

sig-map is the map structure that has error resilience property. When considering the whole bitstream (including signs and refinement data), as shown in Fig. 7, the penalty introduced by fixed-length comp-sig-map is about 2% to 3% of the conventional SPIHT bitstream.

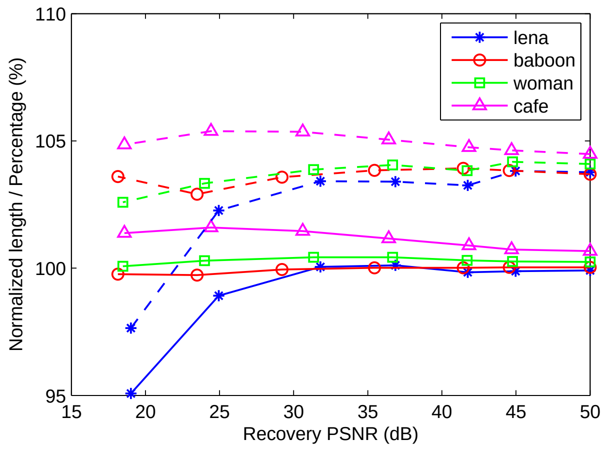


Fig. 6. The length of the prog-sig-map normalized by that of the conventional (AC) sig-map. Solid-line curves: normalized progressive (AC) sig-map; dashed-line curves: normalized progressive (fixed) sig-map.

B. Error Resilience Performance

In order to demonstrate error resilience of the prog-sig-map, we introduce bit errors randomly only into the comp-sig-map, where errors were shown to cause only local damage, and leave the preceding sum-sig-map and following value bit stream error free. In this way, we can evaluate the effects of comp-sig-map errors alone. We want to compare it to bit errors inserted randomly into the conventional sig-map at a point corresponding to the beginning of the comp-sig-map in the prog-sig-map. Since the conventional and progressive sig-maps have different lengths and there is no possible registration of bits between them, we leave error free the same proportion of

[4]Both C and MATLAB implementations of SPIHT are available online: http:// qccpack.sourceforge.net; http://www.cipr.rpi.edu/research/SPIHT/ .

[5]Degradation in PSNR due to channel errors does not depend on the type of filters used for the wavelet transform.

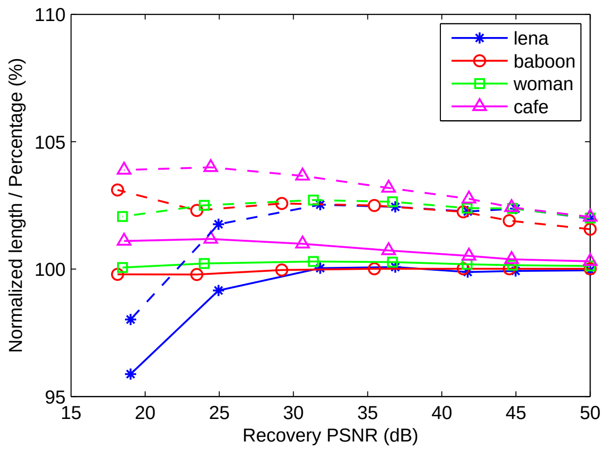


Fig. 7. The length of the whole bitstream with prog-sig-map normalized by that of the whole bitstream with conventional (AC) sig-map. Solid-line curves: normalized bitstream with progressive (AC) sig-map; dashed-line curves: normalized bitstream with progressive (fixed) sig-map.

the conventional sig-map as the prog-sig-map (approximately 60%) and insert bit errors randomly in the rest. This would not be a realistic scenario in practice, say for a UEP (unequal error protection) application, since we can never completely guarantee error-free reception even with strong channel codes and/or ARQ (automatic request repeat). Furthermore, in the UEP scenario, it implies strong protection of the sum-sig-map in all bit planes and the conventional sig-map only in the highest, most important bit planes. We shall show later that the redundancy bits of the channel code are wasted in protection of the lower bit planes. However, these comparative simulations will serve to validate the claim of robustness of the prog-sig-map. That is their only purpose.

We have adopted a memoryless binary symmetric channel characterized by its bit error rate (BER), or known as crossover probability. Experimental comparisons between the progressive and the conventional sig-maps are made for the four test images at the BERs of 0, 0.001, 0.002, 0.005, 0.008, and 0.01. Similar to SPIHT, we vary the coded bit rate by adjusting the threshold of final bit-plane (four were tested here: 32, 16, 8, and 4); all reported coding results are averaged over 20 simulations.

The plots of PSNR versus bit rates for different images and BERs are shown in Fig. 8. (We have omitted the curves of BERs 0.002 and 0.010 from this figure.) We have plotted only one red curve for all tested non-zero BERs in Fig. 8 for the conventional sig-map, because the graphs of PSNR versus rate almost completely overlap for different tested BERs. (The recovered PSNR is largely determined by the location of the first bit error.) The following observations may be gleaned from these plots.

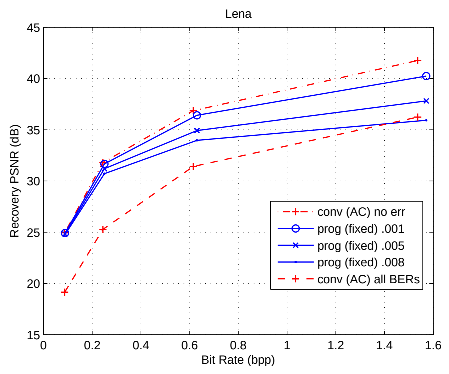
1) The conventional sig-map shows significant degradation in performance in the presence of random bit errors. Comparing the red/dashed-line curves of error-free and error-present transmission of the conventional sig-map, we can clearly observe significant PSNR loss (often $> 5dB$) in the presence of random errors regardless of BER levels.
2) The progressive sig-map offers a graceful PSNR degradation as the BER increases. For example, for the PSNR of about 35 dB, the prog-sig-map is 5 to 9 dB higher than the conventional sig-map for all test images when BER = 0.001. As the BER increases, the gain decreases as expected.
3) The progressive sig-map is more effective for low to medium bit-rates than high bit-rates. Since random bit errors are added from the very beginning of the comp-sig-map, errors are introduced from the highest bit-plane, causing the most severe damage locally. As the bit rate increases, the extra bits are used to code lower bit-planes, which introduces minor increase in recovery quality, because local damage from errors in the higher bit-planes is still present.

In summary, we conclude that prog-sig-map offers an appealing error-resilient coding tool primarily for low and medium bit rates and BERs smaller than 0.01 under our simulation condition, by which the "whole" sum-sig-map is error-free and random bit errors are added from the "beginning" of the comp-sig-map. This condition was chosen for simplicity and serves to reveal the error resilient property of the proposed prog-sig-map.

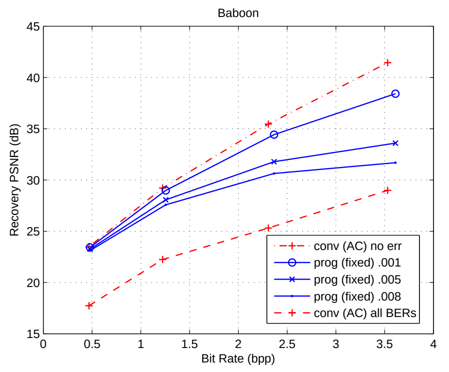
To facilitate subjective evaluation, recovered $Lena$ images from conventional (AC) and progressive (fixed) sig-maps, both at the BER of 0.001, are shown in Fig. 9. Comparing Fig. 9(a) and 9(b), or comparing Fig. 9(c) and 9(d), we can see that the erroneous prog-sig-map provides much better recovery quality than the erroneous conventional sig-map at a similar coding bit rate, both objectively (in terms of the PSNR) and subjectively. The bit rates of prog-sig-map are only slightly higher, due to the raw fixed-length comp-sig-map. Figs. 9(b) and 9(c) have similar PSNR results, but the former has much lower bit rates and even better subjective quality. One reason is that the erroneous prog-sig-map (more specifically, comp-sig-map) degrades the recovered image locally. As evinced earlier in [34], the reconstructed images with localized distortions tend to be preferred over the ones with global distortions in terms of subjective quality.

C. Computational Complexity

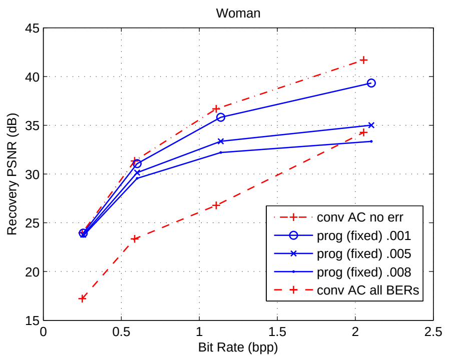
Another salient feature of the proposed sig-map is that it adds little complexity to the already low complexity of SPIHT. To justify this claim, we compare the run times for lossless encoding of four test images. These run-time numbers are reported for a PC with Intel Core 2 Duo CPU and 2.95 GB RAM. As for the SPIHT encoding operations only (excluding file I/O, transformation, and arithmetic coding), provided in upper half of Table I, the progressive (fixed) sig-map increases the complexity slightly, due to the summation operations to produce w for the sum-sig-map and the additional operations to get the comp-bits. However, the low complexity property of SPC is still maintained (processing speed is approximately in the order of 10^6 pixels per second). In a realistic system, where AC is applied to the conventional sig-map and the sum-sig-map, the running times (including SPIHT encoding, AC, and



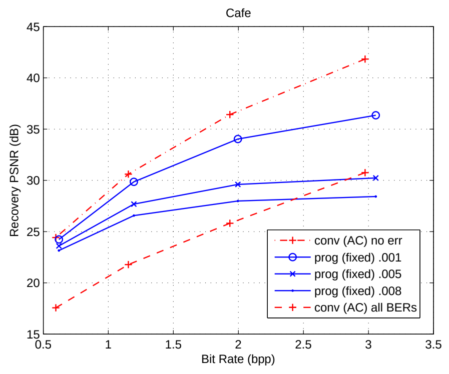
(a) Lena



(b) Baboon

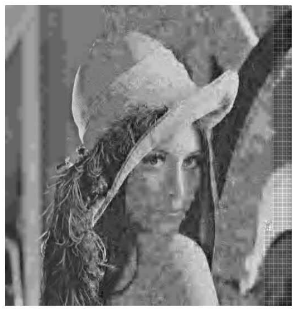


(c) Woman

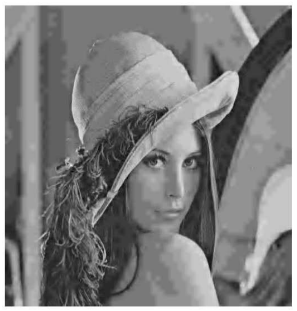


(d) Cafe

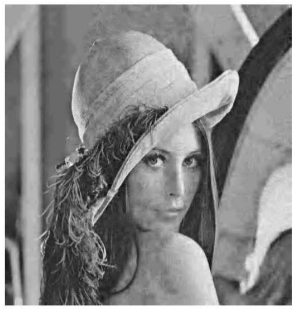
Fig. 8. Rate-distortion performance of conventional (AC) and progressive (fixed) sig-maps when BER = 0.001, 0.005, and 0.008



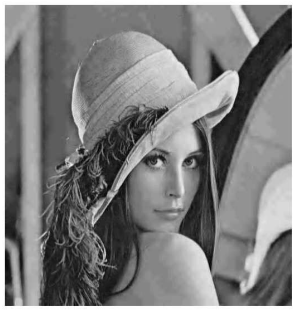
(a) Recovered from conventional (AC) sig-map (0.244 bpp, 24.12 dB)



(b) Recovered from progressive (fixed) sig-map (0.250 bpp, 31.61 dB)



(c) Recovered from conventional (AC) sig-map (0.615 bpp, 31.59 dB)



(d) Recovered from progressive (fixed) sig-map (0.630 bpp, 36.49 dB)

Fig. 9. Reconstructed images for Lena when BER = 0.001.

file output operations[6]) are provided in the lower half of Table I. The progressive (fixed) sig-map is even computationally simpler than the conventional (AC) sig-map, mainly because the fixed-length comp-sig-map need not AC.

	Lena	**Baboon**	**Woman**	**Cafe**
SPIHT encoding only:				
conv. (AC) sig-map	0.0460	0.0520	1.0640	1.1215
prog. (fixed) sig-map	0.0479	0.0541	1.1044	1.1679
SPIHT encoding, arithmetic coding, and file output:				
conv. (AC) sig-map	0.1442	0.1873	3.1554	3.7022
prog. (fixed) sig-map	0.1276	0.1740	2.8723	3.4647

TABLE I
RUNNING TIME (IN SECOND) FOR DIFFERENT SIG-MAPS.

V. APPLICATIONS

One promising application of the prog-sig-map is image transmission in error prone environments. Errors occurring in the comp-sig-map and value bitstream produce only local damage, while errors in the sum-sig-map, which comprises about 60% of the progressive sig-map and 36% of the entire bitstream for recovery with moderate to high quality, propagate and cause global damage. Therefore, the sum-sig-map needs stronger protection from bit errors than the other segments of the bitstream (the UEP technique).

We can assess the potential of the prog-sig-map in a UEP scenario with the following experiment. We mentioned previously that it was wasteful of channel code bits to protect strongly the lower bit planes as we do in our simulations to prove error resiliency. It is advantageous to shift protection of lower bit-planes of the sum-sig-map to higher bit-planes of the comp-sig-map. Suppose the sum-sig-map is L bits long. We still assume L bits of the proposed sig-map error-free (implemented with strong channel codes or other techniques), but assign 0.9L to the sum-sig-map (from the beginning) and 0.1L to the comp-sig-map (from the beginning). Under this arrangement and BER 0.008, the rate-distortion performance for Lena is 39.2 dB @ 1.571 bpp, instead of the 36.1 dB @ 1.571 bpp in Fig. 8. This significant PSNR gain of 3.1 dB, achieved with an arbitrarily chosen 90% and 10% division of error protection, shows the potential of using the prog-sig-map in a UEP scenario. Thus the simulation assumption for Fig. 8 that protects the entire sum-sig-map and leaves the following comp-sig-map unprotected is just for simplicity to reveal the error resilient property of the proposed prog-sig-map. It is far from its optimal performance in a UEP application.

Distributed source coding (DSC) is another possible application. In the literature, [35], [36], [37], [38] to name a few, only the value information can be Slepian-Wolf (SW) coded [39]. The sig-map conventionally is difficult to be SW coded. On one hand, it is hard to generate an effective side information sig-map (of the same length and bit-wise correlated with the original one). On the other hand, the conventional sig-map is error sensitive, while decoding errors are unavoidable for SW coding even if the error rate is very low. The prog-sig-map alleviates these difficulties. With the intra coded sum-sig-map, both the comp-sig-map and the value information can be further SW coded with the commonly used channel-code-based (such as Turbo or LDPC codes) SW codec. An implementation of this framework is described and simulated for distributed video coding in [40].

Another DSC related application is to take advantage of the progressive feature of the comp-sig-map. More specifically, the comp-sig-map could be used as rate-adaptive SW data, to send out initially a portion of the comp-bits according to the SW boundary, $H(X|Y)$. Upon failed decoding, more progressive comp-bits are requested and transmitted until correct decoding is achieved. When all comp-bits are transmitted, a successful decoding is guaranteed. In this DSC model, the ambiguity, which is generated at SW encoding and clarified at SW decoding assisted by side information, is introduced by an ambiguous sig-map. To the best our knowledge, this would be a novel approach of ambiguity generation in DSC applications. The two main existing ambiguity generation methods are based on channel coding as in [41] [42] [43] [44] or arithmetic coding as in [45] [46].

[6]The image input and transformation operations are excluded in this comparison because they are exactly the same for different tested sig-maps.

VI. CONCLUSION

To improve the error-resilient property of SPC, we propose a novel data representation called progressive significance map in this paper. It separates the significance map of SPC into two layers: sum-sig-map (number of significant coefficients in a set) and comp-sig-map (locations of significant points in the set). Just as the uncoded value data – signs and refinement bits – are less sensitive than the significance map, the fixed-length coded comp-sig-map has better error-resilient property than the variable-length coded sum-sig-map. The simulation results show that the proposed method achieves much better error resilience performance in BSC, with about 2% to 3% penalty on coding efficiency, while maintaining low computational complexity. To the best of our knowledge, this is the first time that error resilience is achieved within the core framework of SPC itself. Designed as a tool supplementary to other existing error resilience coding techniques, the progressive significance map can be jointly used with FEC, UEP, independent packetization, EREC and so on, to achieve higher robustness in a variety of error prone environments. The proposed prog-sig-map also lends itself to other applications such as distributed video coding.

REFERENCES

[1] J. M. Shapiro, “Embedded image coding using zerotrees of wavelet coefficients,” *IEEE Trans. Signal Processing*, vol. 41, no. 12, pp. 3445–3462, 1993.

[2] ISO/IEC, “Information technology—JPEG 2000 image coding system, Part-1: Core coding system,” *ISO/IEC 15444-1*, 2000.

[3] ISO/IEC, “Information technology—JPEG 2000 image coding system, Part-2: Core coding system,” *ISO/IEC 15444-2*, 2001.

[4] CCSDS, “Image data compression,” *CCSDS-122.0-B-1 Blue Book*, 2005.

[5] P. S. Yeh, P. Armbruster, A. Kiely, B. Masschelein, G. Moury, C. Schaefer, and C. Thiebaut, “The new CCSDS image compression recommendation,” in *Proc. IEEE Aerospace Conf.*, pp. 4138–4145, 2005.

[6] A. Said and W. A. Pearlman, "A new, fast, and efficient image codec based on set partitioning in hierarchical trees," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 6, no. 3, pp. 243–250, 1996.

[7] W. A. Pearlman, A. Islam, N. Nagaraj, and A. Said, "Efficient, low-complexity image coding with a set-partitioning embedded block coder," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 14, no. 11, pp. 1219–1235, 2004.

[8] J. Andrew, "A simple and efficient hierarchical image coder," in *Proc. Int. Conf. on Image Processing (ICIP)*, vol. 3, pp. 658–661, 1997.

[9] C. Chrysafis, A. Said, A. Drukarev, A. Islam, and W. A. Pearlman, "SBHP–a low complexity wavelet coder," in *IEEE Int. Conf. on Acoust., Speech, Signal Processing (ICASSP)*, vol. 6, pp. 2035–2038, 2000.

[10] S. T. Hsiang, *Highly scalable subband/wavelet image and video coding*. PhD thesis, Rensselaer Polytechnic Institute, 2002.

[11] W. A. Pearlman and A. Said, "Set partition coding: Part I of set partition coding and image wavelet coding systems," *Foundations and Trends in Signal Processing*, vol. 2, no. 2, pp. 95–180, 2008.

[12] W. A. Pearlman and A. Said, "Image wavelet coding systems: Part II of set partition coding and image wavelet coding systems," *Foundations and Trends in Signal Processing*, vol. 2, no. 3, pp. 181–246, 2008.

[13] P. G. Sherwood and K. Zeger, "Progressive image coding on noisy channels," in *Proc. Data Compression Conf. (DCC)*, pp. 72–81, 1997.

[14] P. G. Sherwood and K. Zeger, "Progressive image coding for noisy channels," *IEEE Signal Processing Letters*, vol. 4, no. 7, pp. 189–191, 1997.

[15] P. G. Sherwood and K. Zeger, "Error protection for progressive image transmission over memoryless and fading channels," *IEEE Trans. Communications*, vol. 46, pp. 1555–1559, 1998.

[16] N. Thomos, N. V. Boulgouris, and M. G. Strintzis, "Wireless image transmission using Turbo codes and optimal unequal error protection," *IEEE Trans. Image Processing*, vol. 14, no. 11, pp. 1890–1901, 2005.

[17] A. A. Alatan, M. Zhao, and A. N. Akansu, "Unequal error protection of SPIHT encoded image bit streams," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 6, pp. 814–818, 2000.

[18] R. Hamzaoui, V. Stankovic, and Z. Xiong, "Optimized error protection of scalable image bit streams: Advances in joint source-channel coding for images," *IEEE Signal Processing Magazine*, vol. 22, no. 6, pp. 91–107, 2005.

[19] C. D. Creusere, "A new method of robust image compression based on the embedded zerotree wavelet algorithm," *IEEE Trans. Image Processing*, vol. 6, no. 10, pp. 1436–1442, 1997.

[20] J. K. Rogers and P. C. Cosman, "Wavelet zerotree image compression with packetization," *IEEE Signal Processing Letters*, vol. 5, no. 5, pp. 105–107, 1998.

[21] T. Kim, S. Choi, R. E. Van Dyck, and N. K. Bose, "Classified zerotree wavelet image coding and adaptive packetization for low-bit-rate transport," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 11, no. 9, pp. 1022–1034, 2001.

[22] S. H. Yang and T. C. Cheng, "Error-resilient SPIHT image coding," *Electronics Letters*, vol. 36, no. 3, pp. 208–210, 2000.

[23] D. W. Redmill and N. G. Kingsbury, "The EREC: An error-resilient technique for coding variable-length blocks of data," *IEEE Trans. Image Processing*, vol. 5, no. 4, pp. 565–574, 1996.

[24] W. M. Lam and A. Reibman, "Self-synchronizing variable-length codes for image transmission," in *IEEE Int. Conf. on Acoust., Speech, Signal Processing (ICASSP)*, pp. 477–480, 1992.

[25] P. C. Cosman, J. K. Rogers, P. G. Sherwood, and K. Zeger, "Combined forward error control and packetized zerotree wavelet encoding for transmission of images over varying channels," *IEEE Trans. Image Processing*, vol. 9, no. 6, pp. 982–993, 2000.

[26] S. Cho and W. A. Pearlman, "Multilayered protection of embedded video bitstreams over binary symmetric and packet erasure channels," *Journal of Visual Communication and Image Representation*, vol. 16, no. 3, pp. 359–378, 2005.

[27] J. J. Meany and C. J. Martens, "Split field coding: Low complexity error-resilient entropy coding for image compression," in *Applications of Digital Image Processing XXXI, Proc. SPIE*, vol. 7073, 2008.

[28] R. F. Rice, "Some practical universal noiseless coding techniques," in *Jet Propulsion Laboratory Publication 79-22*, 1979.

[29] E. R. Fiala and D. H. Greene, "Data compression with finite windows," *Communications of the ACM*, vol. 32, no. 4, pp. 490–505, 1989.

[30] Y. Hu and W. A. Pearlman, "Motion differential set partition coding for image sequence and video compression," *J. Vis. Comm. Image Represent.*, vol. 23, pp. 634–647, May 2012.

[31] B. J. Kim and W. A. Pearlman, "An embedded wavelet video coder using three-dimensional set partitioning in hierarchical trees (SPIHT)," in *Proc. Data Compression Conf. (DCC)*, pp. 251–260, 1997.

[32] I. H. Witten, R. M. Neal, and J. G. Cleary, "Arithmetic coding for data compression," *Communications of the ACM*, vol. 30, no. 6, pp. 520–540, 1987.

[33] A. R. Calderbank, I. Daubechies, W. Sweldens, and B. L. Yeo, "Lossless image compression using integer to integer wavelet transforms," in *Proc. Int. Conf. on Image Processing (ICIP)*, vol. 1, pp. 596–599, 1997.

[34] N. Serrano, D. Schilling, and P. C. Cosman, "Quality evaluation for robust wavelet zerotree image coders," in *Proc. 2nd Annual UCSD Conf. Wireless Communications*, pp. 128–134, 1999.

[35] N. M. Cheung, C. Tang, A. Ortega, and C. S. Raghavendra, "Efficient wavelet-based predictive Slepian-Wolf coding for hyperspectral imagery," *EURASIP Journal on Signal Processing, Special Section on Distributed Source Coding*, vol. 86, no. 11, pp. 3180–3195, 2006.

[36] V. Thirumalai, I. Tosic, and P. Frossard, "Balanced distributed coding of omnidirectional images," in *Proc. SPIE VCIP*, vol. 6822, 2008.

[37] X. Guo, Y. Lu, F. Wu, and W. Gao, "Distributed video coding using wavelet," in *Proc. IEEE Int. Symposium on Circuits and Systems (ISCAS)*, pp. 5427–5430, 2006.

[38] Y. Zheng, A. Wang, and Z. Li, "Wavelet-domain distributed video coding using optimized TCQ," in *Proc. Int. Conf. on Innovative Computing, Information and Control (ICICIC)*, pp. 361–364, 2009.

[39] D. Slepian and J. Wolf, "Noiseless coding of correlated information sources," *IEEE Trans. Information Theory*, vol. 19, no. 4, pp. 471–480, 1973.

[40] Y. Hu and W. A. Pearlman, "Distributed video coding with progressive significance map," in *Visual Information Processing and Communication III, Proc. SPIE*, vol. 8305, 2012.

[41] S. S. Pradhan and K. Ramchandran, "Distributed source coding using syndromes (DISCUS): Design and construction," *IEEE Trans. Information Theory*, vol. 49, no. 3, pp. 626–643, 2003.

[42] R. Puri and K. Ramchandran, "PRISM: A new robust video coding architecture based on distributed compression principles," in *Proc. Allerton Conf. on Cummunication Control and Computing*, vol. 40, pp. 586–595, 2002.

[43] Z. Xiong, A. D. Liveris, and S. Cheng, "Distributed source coding for sensor networks," *IEEE Signal Processing Magazine*, vol. 21, no. 5, pp. 80–94, 2004.

[44] B. Girod, A. M. Aaron, S. Rane, and D. Rebollo-Monedero, "Distributed video coding," *Proc. IEEE Special Issue on Advances in Video Coding and Delivery*, vol. 93, no. 1, pp. 71–83, 2005.

[45] M. Grangetto, E. Magli, and G. Olmo, "Distributed arithmetic coding," *IEEE Communications Letters*, vol. 11, no. 11, pp. 883–885, 2007.

[46] M. Grangetto, E. Magli, and G. Olmo, "Distributed arithmetic coding for the Slepian–Wolf problem," *IEEE Trans. Signal Processing*, vol. 57, no. 6, pp. 2245–2257, 2009.