ZM-SPECK: A Fast and Memoryless Image Coder for Multimedia Sensor Networks

Naimur Rahman Kidwai, Ekram Khan, *Senior Member, IEEE*, and Martin Reisslein, *Fellow, IEEE*

***Abstract*— The set partitioned embedded block (SPECK) algorithm is an efficient block-based image coder to encode wavelet transformed images. SPECK uses linked lists to track significant/insignificant coefficients and block sets, thereby having a large memory requirement that increases with the encoding rate. Furthermore, multiple memory read/write operations and list management slow down the algorithm. In addition, the implementation of the traditional discrete wavelet transform (DWT) is memory intensive and time-consuming. Therefore, it is difficult to implement image coding using the traditional DWT and SPECK algorithm on low-cost visual sensor nodes. Most of the existing studies on low-memory implementations of the SPECK algorithm attempt to replace the dynamic memory of linked lists by a static memory in the form of fixed-length state tables/markers. In this paper, a fast and memoryless image coder is proposed, which uses the fractional wavelet filter to calculate the DWT coefficients of the image and a zero-memory listless SPECK algorithm for quantization and coding of the DWT coefficients. The proposed algorithm, referred as zero-memory SPECK (ZM-SPECK), completely eliminates the linked lists and only uses a few registers to perform some low-level arithmetic/logical operations. The elimination of linked lists also reduces the memory access time, thereby making ZM-SPECK faster than the original SPECK algorithm. Simulation results show that the proposed ZM-SPECK coder outperforms the contemporary state-of-the-art wavelet image coders in terms of memory requirement and computational complexity, while retaining their coding efficiency. The proposed ZM-SPECK image coder is thus very well suited for image communication in visual sensor networks.**

***Index Terms*— Fractional wavelet filter, memory efficient image codec, SPECK, visual sensors, wireless multimedia sensor networks.**

I. Introduction

A. Motivation

WIDESPREAD use of wireless networks (e.g., Wi-Fi and cellular networks), has led to an exponential growth of image communications through handheld portable multimedia devices (e.g. digital cameras, smart phones, and tablets). Also, developments in micro-electromechanical systems, and wireless communication together with low-cost imaging devices, have led to the development of wireless networks of visual sensors [1]–[4], called visual sensor networks (VSNs) or wireless multimedia sensor networks (WMSNs). The low-cost sensor nodes are severely constrained in terms of resources. The transmission of real-time images to a more resourceful hub or sink node over a band-limited wireless channel is a challenging task. On-chip random access memory (RAM) available on low-cost sensor nodes is limited, and has become a major constraint for the processing of large images on the sensor nodes [5]. The on-chip RAM available on most of the low-cost sensor nodes is of the order of 10 kB [6]. For sensor nodes equipped with a visual sensor (or camera) to capture images and transmit them over WSNs, there is a need for memory-efficient and low-complexity image codecs [7]–[12].

For efficient compression of images, a number of image coding algorithms [13]–[26] have been developed. The contemporary image coding methods, such as JPEG2000 [13], embedded zero tree of wavelet coefficients (EZW) [15], set partitioning in hierarchical trees (SPIHT) [16], virtual SPIHT (VSPIHT) [18], set partitioned embedded block coding (SPECK) [20], wavelet block tree coding (WBTC) [25], and wavelet lower tree (LTW) [26] support a wide range of functionalities. However, they have high computational complexity, high memory requirement, or generate non-embedded bit-streams.

Most wavelet-based image coding algorithms achieve compression by aggregating a large number of insignificant coefficients either in spatial trees (zero-trees) [14]–[18], or in spatial blocks (zero-blocks) [19]–[24], or spatial block-trees [25]. Among them, SPIHT [16] and SPECK [21] are the most popular coding algorithms due to their superior compression and low computational complexity (high energy efficiency) [27], [28]. A common feature of these algorithms is that they use multiple linked lists to track the locations of significant/insignificant coefficients or sets (blocks or trees). The SPECK [21] algorithm uses only two lists instead of three in SPIHT [16]. However, the continuously (dynamically) growing lists in these codecs not only result in variable and data-dependent memory requirements, but also necessitate memory management as the list nodes are added, deleted, sorted, or moved among the lists. This significantly increases the encoding/decoding times due to multiple memory accesses and increased memory read/write operations. These problems become more severe for encoding/decoding of high-resolution images using portable devices with limited resources.

Manuscript received September 29, 2015; revised January 15, 2016; accepted January 16, 2016. Date of publication January 19, 2016; date of current version February 24, 2016. The associate editor coordinating the review of this paper and approving it for publication was Dr. Anna G. Mignani.

N. R. Kidwai is with the Department of Electronics and Communication Engineering, Integral University, Lucknow 226026, India (e-mail: naimkidwai@gmail.com).

E. Khan is with the Electronics Engineering Department, Aligarh Muslim University, Aligarh 202002, India (e-mail: ekhan.el@amu.ac.in).

M. Reisslein is with the Goldwater Center, School of Electrical, Computer, and Energy Engineering, Arizona State University, Tempe, AZ 85287-5706 USA (e-mail: reisslein@asu.edu).

Digital Object Identifier 10.1109/JSEN.2016.2519600

B. Related Work

Over the years, significant efforts have been made to reduce the memory requirements of wavelet-based coders [29]–[47]. The memory requirements for these coders can be divided into two parts: transform memory (memory required for wavelet and inverse wavelet transforms) and coding memory (memory requirement for encoding/decoding of wavelet coefficients). The overall codec memory is either the sum of two memories or the maximum of the two, depending on whether the two stages run in parallel or sequentially. Traditional implementations of the discrete wavelet transform (DWT) of images generally require a large memory. A number of efforts have been made to reduce the transform memory [29]–[36]. The line-based implementations of the wavelet transform with or without the lifting scheme [29], [30] are among the earliest efforts in this direction. The line based approach requires about 26 kB of on-board memory (or RAM) for a six level transform of a 512×512 gray scale image [6]. Another approach to reduce memory is to apply the DWT on a block-by-block basis, rather than on the entire image [31], [32]. However, the block-based approach requires almost the same memory as the line-based approach [6]. Strip-based low memory wavelet transform architectures have been proposed in [33]–[35]. Recently, the Fractional Wavelet Filter (FrWF) has been proposed [36], which requires only a few kB of memory to compute the DWT of images and is therefore suitable for implementation in memory-constrained portable multimedia devices and sensor nodes.

In order to reduce the coding memory (for encoding/ decoding of wavelet coefficients) of zero tree and zero block coding algorithms, generally one of the following three approaches is used: reducing the number of elements in lists [25], reducing the number of lists [37]–[39], or replacing lists with state-tables or markers (listless approach) [40]–[47]. The listless approach uses fixed-size static memory in the form of state-tables/markers, in place of lists. For example, No-list SPIHT (NLS) [41] uses 4 bits/coefficient marker, while the implementation in [42] uses 3 bits/coefficient marker. Motivated by the NLS algorithm, a number of listless SPECK algorithms have been developed. Among them, the listless SPECK (LSK) algorithm [43] uses 2 bits/coefficient marker, the no-list implementation of SPECK proposed in [44] uses 0.75 bits/ coefficient marker, while the no-list implementation of SPECK [47] uses 0.25 bits/coefficient marker to keep track of blocks and coefficients to be significance tested.

Although, the listless implementations of state-of-the-art wavelet image coders achieved significant memory reductions, the overall memory requirement is still very high for low-cost sensor nodes. To the best of our knowledge, only limited efforts have been made to minimize the overall memory requirements of image coders. The wavelet image two line coder (Wi2l) [48] which combines FrWF with line-based BCWT [49], [50], has made significant progress towards overall low-memory image coding. However, the Wi2l coder produces a non-embedded bit-stream, which is a major drawback since embedded bit-streams are highly desirable for scalable image transmission over heterogeneous networks [15]. The low-memory block-tree coder (LMBTC) [51] is a low memory version of WBTC [25] and uses only a few markers. Combining LMBTC with FrWF drastically reduces the overall memory requirement, while retaining the embedding property. However, the memory requirement of the FrWF-based LMBTC coder depends on the number of wavelet decomposition levels and the image size, and the use of markers in LMBTC increases its computational complexity. Thus, there is still a need to design an efficient, feature-rich, and low-memory image coder for portable multimedia devices and visual sensor nodes.

C. Contribution

In this paper, we propose a fast memory-less SPECK algorithm combined with FrWF that significantly reduces the overall memory requirement of an image coder. The memory requirement at the transform stage is reduced due to FrWF and the memory requirement of the encoding stage is reduced by the proposed listless SPECK algorithm. The algorithm encodes the wavelet coefficients of an image with the partitioning rules of SPECK. However, the proposed algorithm does not require any lists/ state-tables/ markers, and therefore requires no static or dynamic memory. For this reason, the proposed coder is named the **Z**ero-**M**emory **SPECK** (ZM-SPECK) image coder. The heart of the coding algorithm is the exploitation of linear indexing features. The proposed coder uses only a few registers to perform arithmetic and logical operations and does not require memory for encoding/ decoding of coefficients. It generates an embedded bit-stream, has reduced computational complexity, and has coding efficiency comparable to that of the original SPECK algorithm.

A summary of a preliminary form of the ZM-SPECK algorithm was presented in [52]. The preliminary form of the ZM-SPECK algorithm was implemented to encode conventional DWT coefficients and a limited set of initial performance results was presented in [52]. In contrast, this paper gives a complete presentation of a refined form of the ZM-SPECK algorithm. Moreover, we combine the ZM-SPECK algorithm with the FrWF-based DWT to build a very low memory image coder. The use of FrWF reduces the transform stage memory drastically (compared to conventional DWT), which in turn reduces the overall memory of the proposed image coder. We present a comprehensive performance evaluation of the image coder built from the FrWF-based DWT and the ZM-SPECK coding algorithm in this paper.

Recent advances in low-memory image coding, such as LMBTC [51], have reduced the memory requirements of the coding stage to less than the memory requirements of the transform stage. For sequential transform and coding stages, the overall memory requirement is the maximum of the memory required at the transform and coding stages. One may thus question the need of further reductions of the coding memory requirements below the memory required for the transform stage as they presently do not reduce the overall memory required for the image coder. However, we believe that through ZM-SPECK we are pushing the coding memory requirements

to their absolute minimum in this study. We expect that future research will reduce the memory requirements of the transform stage. Future low-memory transform approaches can then be combined with our low-memory coding approach to achieve reductions of the overall image coder memory requirement.

The rest of the paper is organized as follows. Section II presents brief background on the FrWF, SPECK, and listless SPECK (LSK) algorithms. Section III first explains the properties of linearly indexed wavelet coefficients that are exploited in the proposed ZM-SPECK algorithm and then introduces the proposed ZM-SPECK algorithm. The simulation results and related discussions are presented in Section IV, and the paper is concluded in Section V.

II. Background

In this section we briefly review the FrWF scheme to compute the DWT of images as well as the SPECK and listless SPECK (LSK) algorithms.

A. Fractional Wavelet Filter (FrWF)

One of the major difficulties in applying the conventional 2D-DWT to images on a resource-constrained platform is the high amount of required working memory. 2D-DWT implementations generally require the original and/or transformed images to be placed in the processor's on-board memory, so that low-pass and high-pass filtering can be separately applied. On personal computers (PCs), which are equipped with very large RAM, this is not a major concern. However, applying the DWT to images on memory-constrained platforms, e.g., digital cameras and visual sensor nodes, is difficult due to limited on-board memory. To overcome this problem, the Fractional Wavelet Filter (FrWF) [36] has recently been proposed as an alternative to the DWT in resource-constrained environment.

A typical sensor node platform consists of a microcontroller extended with external flash memory (MMC or SD card). The image data and computed coefficients are stored on the SD card. For an image of size $N \times N$ pixels, the FrWF uses three buffers namely, $\boldsymbol{s}$, LL_HL, and LH_HH, each of size N bytes. The buffer $\boldsymbol{s}$ stores the current input line, whereas buffers LL_HL and LH_HH store the resulting LL/HL and LH/HH sub-bands destination lines, respectively [53]. For the first level of decomposition, the algorithm reads the image data line-by-line from the SD-card while it writes the subbands line-wise to a different destination on the SD-card. For each subsequent decomposition level, the LL subband of the previous level is considered as the input data and the same process is repeated. Note that the input samples for the first level are the image pixels, each of size one byte, whereas the input samples for the higher levels are either of four bytes (floating point filter) or two bytes (fixed point filter). The filtering does not use in-place memory, rather for each level, a new destination matrix is allocated on the SD-card. However, as the SD-card has plenty of memory, it does not affect the sensor's resources. The memory (in bytes) required for the wavelet transform of an $N \times N$ image with L-level decomposition using FrWF with floating-point and fixed-point filters are $9N$ and $5N$ bytes, respectively, for the first decomposition level ($L = 1$), and $\frac{12N}{2^{(L-1)}}$ and $\frac{6N}{2^{(L-1)}}$ bytes, respectively, for higher decomposition levels ($L \geq 1$) [6].

Although, the FrWF significantly reduces the transform stage memory, there is scope of further memory reduction in the implementation of FrWF. From the above discussions, it is clear that the memory consumption to implement the first level ($L = 1$) of wavelet transform using FrWF is the highest and the memory requirement decreases for higher transform levels. Since the higher levels of transform are performed after the first level, we propose that the memory consumed for the first decomposition level can be reused for the subsequent higher decomposition levels of the wavelet transform. Therefore, the memory required for the computation of the wavelet transform of an image using FrWF will be the same as that required for the first transform level.

B. SPECK Image Coding Algorithm

SPECK [21] is a block-based algorithm to quantize and encode the wavelet coefficients by exploiting similarities among coefficients within a subband. The algorithm consists of three stages: initialization, sorting, and refinement passes, and uses two linked lists: the list of insignificant sets (LIS) and the list of significant pixels (LSP).

The algorithm is initialized by setting threshold $T = 2^b$, where $b = \lfloor \log_2(\max_{(\forall(i,j))}\{\boldsymbol{C}(i, j)\})\rfloor$ is the most significant bit-plane number and $\{\boldsymbol{C}(i, j)\}$ represents the set of wavelet coefficients. The transformed image is initially partitioned into two sets: a root set S (consisting of a block of LL-band coefficients), and a set I (set of remaining coefficients). The LIS is then initialized with the coefficients of the S block and the list LSP is initially left empty.

The sorting pass tests the significance of each set (block of coefficients) of types S and I of the LIS against the current threshold. The significance of S blocks is checked first. If a set S is found to be significant, it is partitioned using quad partitioning, resulting in four equal sized subsets which are added to the LIS. A subset of size 1×1 corresponds to a single coefficient, and when a coefficient is found significant for the first time, its sign bit is also coded and the coefficient is sent to the LSP. If an I-block is found to be significant, it is partitioned using octave-band partitioning, resulting in four sets: three sets of type S and one of type I. The size of each of these three S sets is equal to the size of the chopped portion of set I. The three S sets of octave band partitioning are processed in the regular image-scanning order, after which the newly formed reduced I set is processed. Once all sets in the LIS have been processed for a particular threshold, the refinement pass is initiated, which refines the quantization of the pixels in the LSP (pixels tested significant during the previous passes). The threshold is then reduced by a factor of two and the sequence of sorting and refinement passes is repeated for sets in the LIS against this lower threshold. The sets in the LIS are processed in increasing order of their size. The entire process is repeated until the desired bit-rate is achieved.

Though SPECK is an efficient algorithm with low computational complexity, it uses linked lists to track the set

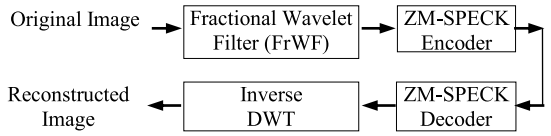


Fig. 1. Block diagram of of proposed image codec: The Fractional Wavelet Filter (FrWF) computes the wavelet transform coefficients, which are encoded with the novel zero-memory SPECK (ZM-SPECK) algorithm.

partitioning and refinement pixels. The use of linked lists in SPECK necessitates large run-time system memory, increases the coder's complexity due to multiple accesses of memory, and requires proper memory management, therefore limiting its applications in memory-constrained environments, such as handheld multimedia devices and WMSNs. To overcome this limitation, a listless SPECK (LSK) algorithm was proposed by Latte et al. [43], which is discussed in the next section.

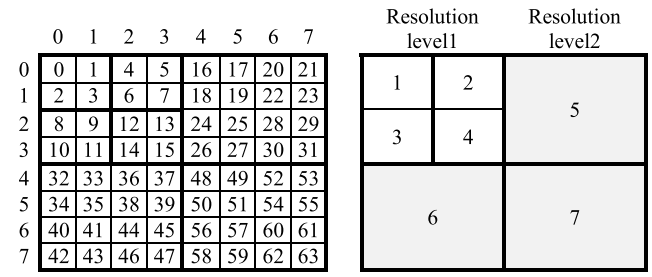
C. Listless SPECK (LSK)

LSK [43] is a listless implementation of the SPECK [21] algorithm. LSK follows the partitioning rules of SPECK without using any list. State information is kept in a fixed-size array that corresponds to the array of coefficient values, with two bits per coefficient to enable fast scanning of the bit planes. LSK uses special markers similar to those used in NLS [41], which are updated at partitioning. The skipping of blocks of insignificant coefficients is efficiently accomplished using a recursive Z-scanning scheme, also known as linear indexing. The linear indexing offers computational and organizational advantages, allowing the indexing of the array coefficients with one index, instead of two indices. The LSK algorithm uses static memory in the form of 2 bits/coefficient marker memory to track the significant/ insignificant pixels/sets.

Replacing dynamic memory by static memory is advantageous, but the memory requirement of the coder is still high, therefore limiting its use in memory-constrained environments. Our proposed ZM-SPECK algorithm, described below, is an attempt to develop an efficient image coder for such applications, as it does not require any static or dynamic memory for coding of wavelet coefficients. Furthermore, the memory required at the transform stage of the encoder is reduced by using the FrWF, instead of the forward DWT.

III. Proposed Memory-Efficient Image Coder

The low processing power and limited RAM of sensor nodes are the major constraints in the processing of images on wireless nodes. A low-complexity image coder with high coding efficiency and low memory requirement is needed for resource-constrained WSNs. The compression can also save energy within the network, as the coding energy is typically lower than the transmission energy [48]. We combine FrWF with the novel zero-memory SPECK (ZM-SPECK) algorithm to design a new image coder. The block diagram of the proposed codec is shown in Fig. 1. The input image is first transformed using the FrWF and then quantized and coded using the ZM-SPECK algorithm. The bit-stream thus generated is transmitted and the received bit-stream is decoded by the ZM-SPECK decoder. Finally, the image is reconstructed using the traditional inverse DWT.



Subband	1	2	3	4	5	6	7
Linear Index Range	0-3	4-7	8-11	12-15	16-31	32-47	48-63
Set Length	4	4	4	4	16	16	16

(c)

Fig. 2. Illustration of linear indexing for an 8×8 image with 2-level DWT: (a) linear index, (b) subbands at different resolution levels, and (c) linear index of each subband and size.

The proposed zero memory SPECK (ZM-SPECK) algorithm is a novel implementation of the SPECK algorithm without the use of any lists or markers. The objective is to reduce the dynamic memory required by the codec to encode/decode wavelet coefficients. In order to reduce the memory requirement at the transform stage, the proposed image codec uses the FrWF. The transformed image, which is usually stored in raster fashion, is then converted into a linear index array [54], described in Subsection III.A. The ZM-SPECK algorithm avoids the use of lists by exploiting the properties of linear indexing.

A. Linear Indexing

Linear indexing allows the addressing of a two dimensional (rows, columns) array with a single index. Let M and N be the numbers of rows and columns of a 2-D array $\boldsymbol{X}$, and let m and n be the row and column indices of a coefficient in that array. The linear index $\Im$ of the 2-D array $\boldsymbol{X}$, varying in the range of 0 to $MN-1$, can be obtained by interleaving the bits of the binary representations of m and n [54]. Fig. 2(a) illustrates the linear indexing of a 2-D array for an 8×8 image with two level dyadic wavelet transform.

In a dyadic wavelet transformed image, the resulting subbands form a pyramidal structure, in which low resolution subbands are at the top of the pyramid and higher resolution subbands are towards the bottom of the pyramid. At each resolution level (except the topmost), there are three subbands, one each in the three spatial resolutions, as illustrated in Fig. 2(b). The linear indexing arranges the wavelet transformed coefficients in Z-scan order by positioning the coefficients belonging to low resolution subbands earlier in the linear-indexed array, than the coefficients of higher resolution subbands. Further, the number of coefficients in any subband is related to the number of coefficients in the LL-subband (with some integer power of 4 as multiplicative factor), as evident from Fig. 2(c). The linear indexing can be utilized to support the operations on coefficient positions that are needed for tree- or block-based wavelet image coding algorithms. The symbols and set

TABLE I
SET STRUCTURES IN 2-D ARRAY (USED IN ORIGINAL SPECK [21]) AND LINEAR INDEX ARRAY (USED IN PROPOSED ZM-SPECK)

	L level dyadic wavelet transformed 2-D array $\mathbf{C}$ of size (M x N)	L level dyadic wavelet transformed linear array $\Im$ of length N_{pix}=MN
Root set	$S_{0,0}^{\frac{M}{2^L},\frac{N}{2^L}} = \left\{ C_{k,l} : 0 \le k < \frac{M}{2^L}, 0 \le l < \frac{N}{2^L} \right\}$	$S_0^{\lambda_{ROOT}} = \left\{ \Im_k : 0 \le k < \lambda_{ROOT}, \text{where} \lambda_{ROOT} = \frac{N_{pix}}{4^L} \right\}$
S Set	$S_{i,j}^{m,n} = \{ C_{k,l} : i \le k < i+m, j \le l < j+n \}$.	$S_i^{\lambda} = \{ \Im_k : i \le k < i+\lambda; i \text{ is the starting index}, \lambda \text{ is set length} \}$.
Quad Partition	$\text{Set of Quad}, O_{i,j}^{m,n} = \left\{ S_{i,j}^{\frac{m}{2},\frac{n}{2}}, S_{i+\frac{m}{2},j}^{\frac{m}{2},\frac{n}{2}}, S_{i,j+\frac{n}{2}}^{\frac{m}{2},\frac{n}{2}}, S_{i+\frac{m}{2},j+\frac{n}{2}}^{\frac{m}{2},\frac{n}{2}} \right\}$.	$\text{Set of Quad}, O_i^{\lambda} = \left\{ S_i^{\frac{\lambda}{4}}, S_{i+\frac{\lambda}{4}}^{\frac{\lambda}{4}}, S_{i+\frac{\lambda}{2}}^{\frac{\lambda}{4}}, S_{i+\frac{3\lambda}{4}}^{\frac{\lambda}{4}} \right\}$.
I set	$I^q = \left\{ C_{k,l} : \frac{M}{2^{L-q}} \le k < M, \frac{N}{2^{L-q}} \le l < N; q = 0,1,2,..\text{such that } q < L \right\}$	$I_i = \left\{ \Im_k : i \le k < N_{pix}; i = 4^q \lambda_{ROOT}, q = 0,1,2,...\text{such that } q < L \right\}$
Octave band partition	$I^q = \left\{ S_{0,\frac{N}{2^{L-q}}}^{\frac{M}{2^{L-q}},\frac{N}{2^{L-q}}}, S_{\frac{M}{2^{L-q}},0}^{\frac{M}{2^{L-q}},\frac{N}{2^{L-q}}}, S_{\frac{M}{2^{L-q}},\frac{N}{2^{L-q}}}^{\frac{M}{2^{L-q}},\frac{N}{2^{L-q}}}, I^{q+1} \right\}$.	$I_i = \left\{ S_i^i, S_{2i}^i, S_{3i}^i, I_{4i} \right\}$

structures used in the original SPECK algorithm, which uses 2D indexing, and the proposed algorithm, which is based on linear indexing, are defined in Table I.

In order to understand the usefulness of linear indexing in block-based wavelet image coding algorithms, such as SPECK, consider an L level wavelet transformed image of size $M \times N$. The number of subbands in the transformed image is $3L+1$. Let k ($k = 1, 2, 3, \ldots, 3L+1$) represent the index of the subbands, whereby $k = 1$ is the index of the LL-subband. Then, the number of coefficients (which we refer to as set length λ) in each subband set, at various resolution levels, is

$$\left.\begin{aligned} \lambda_{ROOT} &= \frac{MN}{4^L}, \text{ for } k = 1, 2, 3, 4 \text{ (resolution level } L) \\ \lambda_p &= 4^p \lambda_{ROOT}, \text{ where } p = \left\lfloor \frac{k-2}{3} \right\rfloor, \text{ for } k \ge 5. \end{aligned}\right\} \tag{1}$$

We have observed several interesting properties of the linear indexing for square size ($M = N = 2^p$, p is an integer) dyadic wavelet transformed images that are useful for tracking the wavelet coefficients [55]:

Property 1: If S_i^{λ} represents a subband set, then the set length λ is one of the factors of starting index i. In particular, the set length λ of a subband set is the highest integer power of 4 by which starting index i of the set is completely divisible. Thus, for a set S_i^{λ} with linear starting index i, the set length λ can be evaluated as.

$$\begin{aligned} \lambda &= \max[4^t], \quad \text{subject to } \frac{i}{4^t} \in I_+ \ \&\ t \in I_+, \\ I+ &\rightarrow \text{Set of positive integers.} \end{aligned} \tag{2}$$

Property 2: The initial index i and set length λ of a given set S_i^{λ} are sufficient to obtain the indexes and set lengths of the quad-partitioned subsets of the set S_i^{λ}.

In linear indexing, the first quad-partitioned subset is obtained by setting the set length to one fourth of the set length of the original set. For the remaining three subsets, the set lengths remain equal to that of the first subset (Table I) and can be evaluated from its starting index using Eqn. (2).

Property 3: The octave bands (set I in SPECK) can be identified as the set of coefficients with starting index i such that set length λ, obtained from i using Eqn. (2), is equal to the index i itself.

In the SPECK algorithm, an I set represents the set of coefficients of the transformed image that have not yet been processed as S sets. After processing the S sets, the I set is processed and if the I set is found significant, it is partitioned into three S sets and one I set [21]. With linear indexing, an I set can be identified using Property 3 and the starting addresses and set lengths of all sets generated due to octave band partitioning can be determined. The initial I set will be a set of coefficients starting from index λ_{ROOT} to the last coefficient of set $\Im$. That is, the initial I set can be defined as $I = \Im(i : MN-1) = \{S_i^i \cup S_{2i}^i \cup S_{3i}^i \cup \Im(4i : MN-1)\}$, where $i = \lambda_{ROOT}$ and $\Im(i : MN-1)$ denotes the set of contiguous elements starting from index i to index $MN-1$ in the linearly indexed array $\Im$. Thus, once an I set is identified, all its subsets due to octave band partitioning can be obtained without using any lists or markers.

Thus, in a linear-indexed dyadic-transformed image, the starting index of a subband set can be used to identify its length (number of elements in that subband). This property of linear indexing allows tracking the partitioned sets (octave and quad partitioned sets) without any lists or markers. These properties of linear indexing motivate us to implement the SPECK algorithm without LIS, which is one of the main contributions of this study.

B. Proposed ZM-SPECK Algorithm

Consider an image $\boldsymbol{X}$ of size $N \times N$ pixels which is L-level wavelet transformed using the FrWF. The transformed coefficients are stored in a linear array $\Im$, that has $N_{\text{pix}} = N^2$ elements. The transformed image $\Im$ exhibits a hierarchical pyramidal structure defined by the number of wavelet decomposition levels. The coefficients in the linear-indexed array are automatically arranged according to the scanning order depicted in Fig. 2(b). That is, the coefficients of lower resolution subbands (lower indexed subbands) are at the beginning of the array, whereas coefficients of higher resolution subbands are positioned towards the end of the array. The set length of the root set (LL-subband) $S_0^{\lambda_{ROOT}}$ is $\lambda_{\text{ROOT}} = N^2/4^L$. The encoding algorithm with the pseudo code given in Table II is performed for each bit plane, starting with the b^{th} bit-plane (b is the most significant bit-plane number) and decrementing b by 1 down to 0 or until a prescribed bit budget is achieved.

TABLE II
PSEUDO CODE OF PROPOSED ZM-SPECK ENCODER

```
(1) Initialization:
    Set number of elements in the array ℑ to N_pix, Root set length to λ_ROOT ,
    and bit plane number b = ⌊ log2 (max {|ℑ|}) ⌋
(2) Sorting Pass:
    Set starting index i of linear array ℑ to i = 0,
    threshold T=2^b, set length λ = λ_ROOT,
    while(i<N_pix)
    {
      ProcessS ( S_i^λ )                          /* S set */
      if (i=λ && i ≥ λ_ROOT)  ProcessI( I_i );    /* I set */
    }
(3) Quantization step update
    Decrement b by 1 and go to step 2

ProcessS ( ) /* Function for processing of block S_i^λ =ℑ(i : i+λ-1)*/
{
  output Γb( S_i^λ )                  /* significance of S_i^λ set, Eq. (3)*/
  if (Γb( S_i^λ )= 0)  i=i+λ;         /*skip insignificant set*/
  else                                /*significant set*/
  {
    if (λ>4) {λ=λ/4; return; }        /*Quad partition*/
    else  Pscan (i, T);               /*Processing of 2x2 block set */
  }
 EvalSL (i, λ)
}

ProcessI( ) /* Function for processing of block I_i= ℑ(i : N-1)*/
{
  output Γb(I_i )                     /*output significance of I set, Eq. (3)*/
  if (Γb(I_i )= 0)  i=N_pix;          /*skip insignificant I set*/
}

Pscan ( ) /* Function for processing of smallest block S_i^4 = ℑ(i : i+3)*/
{
  for j = 0 : 3
  {
    output Γb(ℑ(i +j));               /*output significance, Eq. (3)*/
    if (Γb(ℑ(i +j)) = 1)  output sign bit of ℑ(i +j);  /*New significant pixel*/
    else if( Γb(ℑ(i +j))=NULL) output b^th MSB of ℑ(i +j);
  }
  i=i+4                               /*Increment the index*/
}

EvalSL ( ) /* Function for evaluating set length λ from first index i*/
{
while (bitand (i, 3λ) =0)  λ = 4λ;
}
```

ZM-SPECK follows the set partitioning rules of SPECK, but does not use any lists, state tables, or markers to encode coefficients. The functionalities of the LIS of the original SPECK are performed in the proposed ZM-SPECK algorithm by exploiting the linear indexing properties. By using the linear indexing, the set length of the next set to be processed can be calculated from the starting index of the new set using Eqn. (2), which is highest power of 4 that completely divides the starting index.

As ZM-SPECK does not use any lists or markers, the pixels or set states in a coding pass (or bit-plane) are not preserved (and therefore not known in the subsequent passes). Therefore, in the ZM-SPECK algorithm, each pass begins with the starting index '0' and the set length as the length of the root set. The algorithm is initialized by choosing the root set (LL-subband) as S set. After an S set has been processed, the index is incremented by the set length, giving the starting index of the next S set. The set length of the new S set to be processed is identified with Property 1 of the linear indexing. During the process, a set may be found significant and needs to be quad-partitioned. The quad partitioned subsets should be processed first, before proceeding to the next set. This can be readily achieved in the linear indexed array without any reference to the original set, as explained in the following paragraph.

1) Processing of Quad Partitioned Subsets: Let sets S_i^λ, $S_{i+\lambda}^\lambda, \ldots$ have to be processed (in that order). Set S_i^λ is checked for its significance according to Eqn. (3). If set S_i^λ is found significant, then it is quad-partitioned resulting in four subsets $S_i^{\lambda/4}$, $S_{i+\lambda/4}^{\lambda/4}$, $S_{i+2\lambda/4}^{\lambda/4}$, and $S_{i+3\lambda/4}^{\lambda/4}$ which replace the original set S_i^λ. Now, the set $S_i^{\lambda/4}$ is processed first, followed by the processing of the three subsets $S_{i+\lambda/4}^{\lambda/4}$, $S_{i+2\lambda/4}^{\lambda/4}$, $S_{i+3\lambda/4}^{\lambda/4}$, and then $S_{i+\lambda}^\lambda$ (in order). The algorithm differentiates the quad-partitioned sets from set $S_{i+\lambda}^\lambda$, as the set length of set $S_{i+\lambda}^\lambda$ is four times the set length of the quad-partitioned sets. To perform this function, ZM-SPECK performs an additional bitwise AND operation of the initial index of the current set with three times the previous set length. If the output of the AND operation is non-zero, then the current set has the same length as the previously processed set. Otherwise (if the AND operation gives zero output), the current set has a set length four times longer than the previous set.

2) Processing of Octave Band Partitioned Subsets: Initially, the first I set used in SPECK is the set of the remaining coefficients of the linear-indexed array $\Im$, formed by chopping off $S_0^{\lambda_{ROOT}}$ from $\Im$ and is referred to as $I = \Im(\lambda_{ROOT}: N_{\text{pix}} - 1)$. In ZM-SPECK, the presence of an I set is identified by the fact that the starting index i and the set length λ computed from the starting index are equal for the first S_i^λ set resulting from octave band partitioning. Octave band partitioning can be identified by choosing the first set as S_i^i and the next two sets having the same length, followed by the next three sets of length $4i$, and the process continues, until the last three sets, each of set length $N_{\text{pix}}/4$.

3) Significance Test for Merged Refinement and Sorting Pass: The LSP used in SPECK is avoided in ZM-SPECK by merging the refinement pass into the sorting pass. This merging is achieved through a modified significance test function. Specifically, in ZM-SPECK, the significance $\Gamma_b(\cdot)$ of a set (S or I) or a coefficient against a threshold $T = 2^b$, where b is the bit-plane index, is determined as follows:

$$\Gamma_b(S) = \begin{cases} 0 & if \ \max(S) < T \\ 1 & if \ T \le \max(S) < 2T \\ NULL & if \ \max(S) \ge 2T. \end{cases} \quad (3)$$

According to Eqn. (3), a set is significant if it includes either a newly significant pixel, or a pixel requiring refinement, or both. However, the significant bit '1' will be generated only if it contains at least one newly significant coefficient (i.e., the maximum valued coefficient has a magnitude

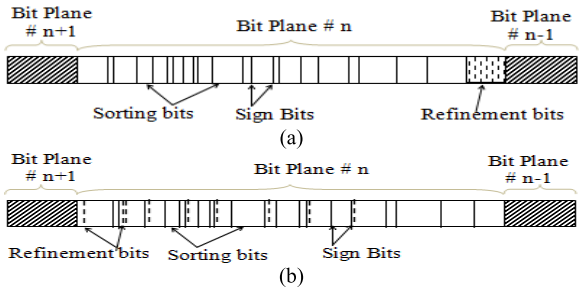


Fig. 3. Bit-stream in sorting pass of (a) SPECK and (b) ZM-SPECK.

between T and $2T$). A magnitude of the largest coefficient of the set greater than or equal to $2T$ implies that at least one coefficient has already been found significant in earlier passes and therefore needs to be refined in the current pass. For such cases, the set is treated as significant but no extra bit will be put out by the encoder. Thus, no coding overhead occurs in processing the sets having refinement pixels as no bit is generated during the significance test. A significant set is iteratively partitioned until smallest sets of length 4 are obtained. Once a smallest-size set is found significant, each coefficient of the set is tested for its significance/refinement and the sign bit is encoded for a newly significant coefficient. The decoder can readily identify the sets with pixels requiring the refinement using the significance test in Eqn. (3). That is, in a set with a refinement pixel, the maximum valued pixel will have magnitude greater than or equal to $2T$, and therefore the decoder can be synchronized with the encoder. These facts are exploited in ZM-SPECK to avoid the use of LSP.

The ZM-SPECK algorithm generates the bits in a different order than the SPECK algorithm, as illustrated in Fig. 3. The SPECK algorithm generates all refinement bits at the end of each (except the most significant) bit-plane, whereas the ZM-SPECK algorithm distributes the refinement bits over the bit-plane. Due to this re-organization of bits (compared to SPECK), ZM-SPECK may slightly degrade the decoded image quality (compared to the original SPECK), if the bit-budget is exhausted in the middle of a bit-plane. This is due to the fact that some of the bits are used in refinements of the coefficients. Therefore, the number of new significant coefficients (which are mainly responsible for increasing the decoded image quality) coded at that bit-budget may be reduced. However, if the bit-budget is exhausted at the end of a pass (or bit-plane), ZM-SPECK has the same performance as SPECK.

The ZM-SPECK algorithm performs all multiplication and division operations by integer powers of 2, which can be implemented by bit shifting operations. Further, the addition operations needed during set partitioning can be implemented with bitwise OR operations.

4) ZM-SPECK Decoder: The ZM-SPECK decoder is symmetric and follows the same algorithm as the encoder with using input instead of output, and sets the bits and signs of coefficients with an additional step to identify the sets that contain coefficients requiring refinement. The decoder performs mid-tread de-quantization for coefficients that are not fully decoded.

5) Summary: The ZM-SPECK algorithm generates an embedded bit-stream with progressive transmission and does not use any lists or markers, thereby reducing the memory requirement and computational cost involved in appending/sorting the dynamic memory or markers. Since the decoder uses similar significance tests for each set as the encoder, the decoder complexity is of the same order as the encoder complexity.

C. Memory Requirement

In order to estimate the memory requirement of the image coder shown in Fig. 1, suppose that the transform and coding stages are performed sequentially. Let M_{T} and M_{C} denote the memory requirements of the transform and coding stages, respectively. The total memory required for coding an image, M_{TOTAL} is

$$M_{\mathrm{Total}} = \max(M_T, M_C). \tag{4}$$

It may be noted that M_{T} depends on the image size and the implementation of the discrete wavelet transform (conventional dyadic transform, lifting-based DWT, or FrWF). For an image of size $N \times N$, the conventional DWT requires $2N^2$ memory elements of four byte each for floating point filter coefficients for low pass and high pass filtering of the coefficients, while the memory required in floating point FrWF is only $9N$ *bytes*.

The coding memory M_{C} depends on the implementation of the quantization and encoding algorithm used to encode the wavelet coefficients. The original SPECK algorithm implements the set-partitioning rules by maintaining two lists, namely LIS and LSP, which are responsible for the large memory consumption of the SPECK algorithm. For a 512×512 image, LIS and LSP require 18 bits to store the address of a set and a coefficient, respectively. Assuming that the numbers of entries in the LIS and LSP are approximately one-fourth of the number of pixels in the image, the total list memory required to implement the SPECK algorithm is approximately $512 \times 512 \times (18+18)/4 = 288$ KB. Similarly, the listless SPECK algorithm [43], which uses 2 bits per coefficient marker, needs 64 KB of memory to store static markers. On the other hand, the proposed ZM-SPECK algorithm does not use any lists or markers. Therefore, ZM-SPECK does not require any static or dynamic memory to store the state information. Thus, for ZM-SPECK-based wavelet image coding, the total memory requirement is equal to the memory required at the transform stage, which can be minimized by using the FrWF instead of the conventional or lifting-based DWT.

D. Computational Complexity

In the SPECK algorithm, every time a quad or octave partitioning of significant S or I sets is performed, four new sets are generated that result in appending four new entries to the LIS and deletion of one previous entry form the LIS. In addition, SPECK necessitates memory management and multiple accesses of linked lists, thereby increasing the time complexity of the algorithm. On the other hand, in the proposed ZM-SPECK coder, quad partitioning is emulated by using one-fourth of the original set length as the new set length.

Octave partitioning is emulated by using the starting index of the *I* set as the set length of the first *S* set. ZM-SPECK identifies a new set to be processed by computing the set-length from its starting index using a bit-wise AND operation. It may be noted that the cost of computation of the new set length in ZM-SPECK is much lower than the complexity of SPECK due to appending of memory, multiple accesses of memory, and complex memory management.

Thus, the ZM-SPECK coding algorithm not only achieves the absolute lowest memory (zero memory) requirement but also reduces the computational complexity compared to SPECK. The ZM-SPECK decoder has an additional step (in comparison to the SPECK decoder) of identifying sets with coefficients to be refined, thereby requiring significance testing of sets. Thus, the complexity of the ZM-SPECK decoder is almost of the same level as the ZM-SPECK encoder complexity.

IV. Simulation Results

In this section the rate-distortion (R-D) performance, memory requirement, and computational complexity of the image coder formed by combining the FrWF transform stage with the ZM-SPECK coding stage is evaluated and compared with other state-of-art coders on twelve standard gray scale (8 bit/pixel) images of dimensions 256×256, 512×512, and 1024×1024 pixels. The test images are taken from a standard image database (http://sipi.usc.edu/database). The images of dimension 256×256 are 'Bird', 'Cameraman', 'Clock', 'Goldhill', of dimension 512×512 are 'Lena', 'Barbara', 'Baboon', 'Cycle', and of dimension 1024×1024 are 'Airplane (U-2)', 'Airport', 'Man', 'Pentagon'. All images are wavelet transformed with five levels of decomposition using the Daubechies 9/7 filter, by either the fractional wavelet filter (FrWF) or the traditional DWT. Floating point transformed coefficients are quantized to the nearest integers, and read into a linear indexed array. These coefficients are then encoded using the SPIHT [16], SPECK [21], WBTC [25], NLS [41], LSK [43], LMBTC [51], and proposed ZM-SPECK algorithms. In sensor networks, decoding of the bit-stream is typically done at workstations which are not constrained in memory and processing power. Therefore, the reconstruction is done using the traditional inverse DWT. All image coding algorithms are implemented using MATLAB7.0 and are executed on a Windows 8.1 Netbook with Intel atom CPU Z 3735F @ 1.33 GHz with 2 GB RAM and 32GB MMC card. For a fair comparison, wavelet transforms (both DWT and FrWF) and all coding algorithms are implemented on the same platform. Since the coders generate embedded bit-streams, the images are encoded only once at a bit rate of 1 bit/pixel, and are decoded at different bit rates from the same embedded bit-stream. Unless otherwise specified, all reported results are averages of the results obtained for all test images of the corresponding size.

A. Coding Efficiency

Coding efficiency is generally measured in terms of the average number of bits per pixel in the coded bit-stream to achieve a minimum desired quality of the reconstructed image.

TABLE III

Image Quality in Terms of PSNR, Number of Pixel Coefficients Found Significant for First Time, and Number of Refinement Bits Generated by SPECK, ZM-SPECK, and LMBTC for the Image 'Lena' (512×512)

Bit per pixel	SPECK [21]			ZM-SPECK			LMBTC [51]		
	Image quality (PSNR in dB)	Newly significant pixel	Refinement pixel	Image quality (PSNR in dB)	Newly significant pixel	Refinement pixel	Image quality (PSNR in dB)	Newly significant pixel	Refinement pixel
0.005	20.81	258	29	20.98	248	135	20.79	238	134
0.01	22.54	474	135	22.49	433	385	22.37	424	382
0.03125	25.64	1333	397	25.71	1269	968	25.55	1235	967
0.05	27.19	2177	970	27.02	1927	1938	26.99	1907	1930
0.0625	27.96	2757	970	27.83	2485	2273	27.76	2443	2254
0.75	28.59	3193	970	28.59	2969	2372	28.51	2922	2367
0.125	30.73	5745	2403	30.46	5011	4941	30.40	4923	4897
0.25	33.73	11597	5796	33.40	9979	10568	33.37	9897	10484

The objective quality of the reconstructed image is measured in terms of the Peak-Signal-to-Noise-Ratio (PSNR), defined as:

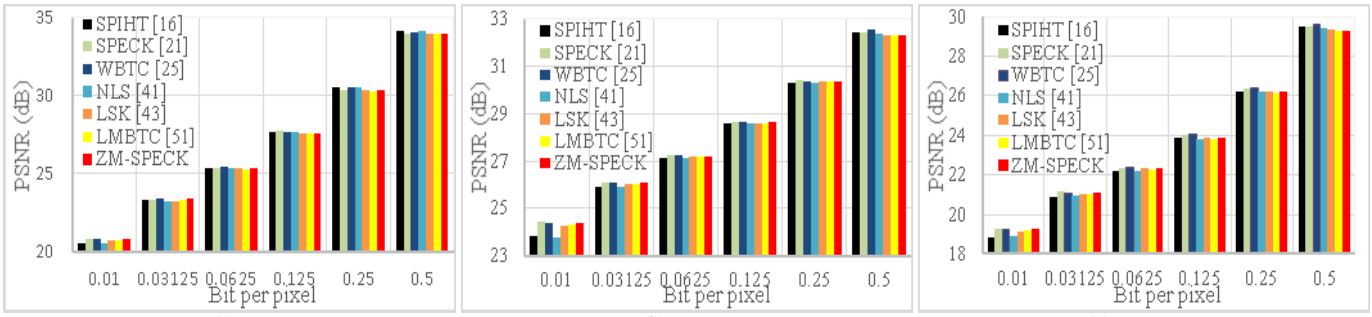
$$PSNR = 10\log_{10}\frac{255^2}{mse} \tag{5}$$

where *mse* is the mean square error of the reconstructed image *g(x, y)* with respect to the original image *f(x, y)*. For an $N \times N$ size image, *mse* is defined as:

$$mse = \frac{1}{NxN}\sum_{x=1}^{N}\sum_{y=1}^{N}[f(x,y) - g(x,y)]^2. \tag{6}$$

Since ZM-SPECK uses the set partitioning rules of SPECK, it is of interest to compare their coding efficiencies. Table III compares the coding efficiencies of the SPECK [21], LMBTC [51], and proposed ZM-SPECK algorithms for the image 'Lena' of size 512×512 pixels. It is worth to emphasize here that the performance of all codecs is exactly the same, irrespective of whether the wavelet transform is implemented using the conventional DWT or the FrWF. We observe from Table III that the coding efficiencies of ZM-SPECK and SPECK are generally equivalent, with only small PSNR variations in the range −0.31∼0.17 dB. It should be noted that both algorithms have the same coding efficiency if an image is decoded at a bit-rate that exhausts exactly at the end of a bit-plane.

However, if an image is decoded at a bit-rate that exhausts in the middle of a bit-plane (or pass), ZM-SPECK has slightly inferior coding efficiency compared to SPECK (see Table III, e.g., for 0.01 and 0.05 bpp). This is because ZM-SPECK merges the sorting and refinement passes into the sorting pass and treats the refinement pixels at par with newly found significant pixels, except that no significance bit is outputted for refinement pixels. Due to the merging of the sorting and refinement passes in ZM-SPECK, some bits which are used for finding significant coefficients in SPECK, are consumed for refining coefficients that have been found significant in earlier passes of ZM-SPECK. Therefore, ZM-SPECK has fewer bits (particularly at the early stages of a bit-plane) for searching of new significant coefficients. Since a bit corresponding to a new



(a) (b) (c)

Fig. 4. PSNR vs Bit rate for ZM-SPECK, SPECK, SPIHT, WBTC, NLS, LSK, and LMBTC coders for image dimensions (a) 256×256, (b) 512×512, and (c) 1024×1024 pixels.

TABLE IV

BD-PSNR GAINS (dB) OF ZM-SPECK IN THE RANGE 0.005-0.1 BIT PER PIXEL

Image Size	SPIHT [16]	SPECK [21]	WBTC [25]	NLS [41]	LSK [43]	LMBTC [51]
256x256	0.73	-0.20	-0.22	0.70	0.35	0.39
512x512	1.32	-0.16	-0.03	1.37	0.36	0.36
1024x1024	2.18	-0.15	0.15	2.17	0.74	0.39

significant coefficient contributes more to the improvement of the image quality (PSNR), than a bit used in the refinement of a coefficient, a lower number of decoded new significant coefficients implies lower PSNR.

If the bit-budget exhausts somewhere near the end of a bit-plane, ZM-SPECK has slightly higher coding efficiency than SPECK (for example at 0.005 and 0.03125 bpp in Table III). This is due to the fact that although ZM-SPECK encodes fewer new significant coefficients than SPECK, ZM-SPECK encodes more refinement bits, and the overall gain in PSNR due to refinement bits is higher than the loss of PSNR due to fewer new significant coefficients. Further, we observe from Table III that ZM-SPECK has higher coding efficiency than LMBTC for almost all bit rates. This is because of the block-based nature of ZM-SPECK.

Fig. 4 compares the coding performance of ZM-SPECK with other state-of-the-art coders (SPIHT, SPECK, and WBTC) and their listless versions (NLS and LSK, LMBTC) for the test images of dimension 256×256, 512×512, and 1024×1024 pixels. We observe from Fig. 4 that in general, the ZM-SPECK image coder has R-D performance equivalent to the other coders. However, at very low bit rates, ZM-SPECK outperforms other listless coders.

Table III and Fig. 4 indicate that the PSNR values of the various coding algorithms at a specified bits/pixel setting are very close to each other, suggesting a comparison in terms of the Bjontegaard delta PSNR (BD-PSNR) [56]. Table IV presents the BD-PSNR gain of ZM-SPECK with respect to the other coders. We observe that ZM-SPECK outperforms SPIHT, and other listless coders, such as NLS, LSK, and LMBTC, but is slightly inferior to SPECK and WBTC.

B. Memory Requirement Analysis

The image communication through portable multimedia devices and wireless sensor nodes is constrained mainly due to low computational power and limited available memory. Many low-cost sensor nodes have only on-board memory of the order of 10 kB [6]. The memory requirement of an image coder with sequential transform and coding stages is the maximum of the memory required at the transform and coding stages. The working memory (RAM) requirements of the ZM-SPECK coder and other state-of-art wavelet-based image coders, such as SPIHT [16], SPECK [21], NLS [41], WBTC [25], and LMBTC [51] with floating point FrWF and conventional DWT implementation for image sizes $N \times N$ ($N = 256$, 512, and 1024) are compared in Table V. We observe from Table V that transforming images using FrWF requires much less memory than the conventional DWT. To be more specific, the FrWF requires memory in the range of 2.25-9.0 kB, whereas the conventional DWT requires 0.5-8 MB of memory for wavelet transformation of images of size varying from 256×256 to 1024×1024 pixels. This is because FrWF stores only three image lines in memory, whereas the conventional DWT stores the entire image/coefficients in the system memory.

TABLE V

MEMORY REQUIREMENTS (IN kB) OF WAVELET TRANSFORM AND CODING STAGES OF SPIHT, SPECK, WBTC, Wi2l, NLS, LSK, LMBTC, AND ZM-SPECK

Image Size	Bit per pixel	FrWF	DWT	SPIHT [16]	SPECK [21]	WBTC [25]	Wi2l [48]	NLS [41]	LSK [43]	LMBTC [51]	ZM-SPECK
256x256	0.01	2.25	512	0.5	0.3	0.6	1.1	32	16	1	0
	0.03125			1.7	0.9	1.7	1.1	32	16	1	0
	0.0625			3.6	1.7	3.5	1.1	32	16	1	0
	0.125			7.4	3.0	7.0	1.1	32	16	1	0
	0.25			13.2	5.8	12.5	1.1	32	16	1	0
	0.5			24.1	11.4	23.8	1.1	32	16	1	0
512x512	0.01	4.5	2048	2.1	1.2	2.5	1.8	128	64	4	0
	0.03125			8.7	4.0	8.7	1.8	128	64	4	0
	0.0625			16.5	7.9	15.6	1.8	128	64	4	0
	0.125			34.2	16.0	32.3	1.8	128	64	4	0
	0.25			62.5	32.5	59.3	1.8	128	64	4	0
	0.5			108.7	53.4	103.6	1.8	128	64	4	0
1024x1024	0.01	9	8192	9.0	5.3	10.4	3	512	256	16	0
	0.03125			36.7	18.1	37.6	3	512	256	16	0
	0.0625			70.0	32.2	67.1	3	512	256	16	0
	0.125			149.0	75.4	142.9	3	512	256	16	0
	0.25			312.6	145.0	296.7	3	512	256	16	0
	0.5			546.9	254.6	511.3	3	512	256	16	0

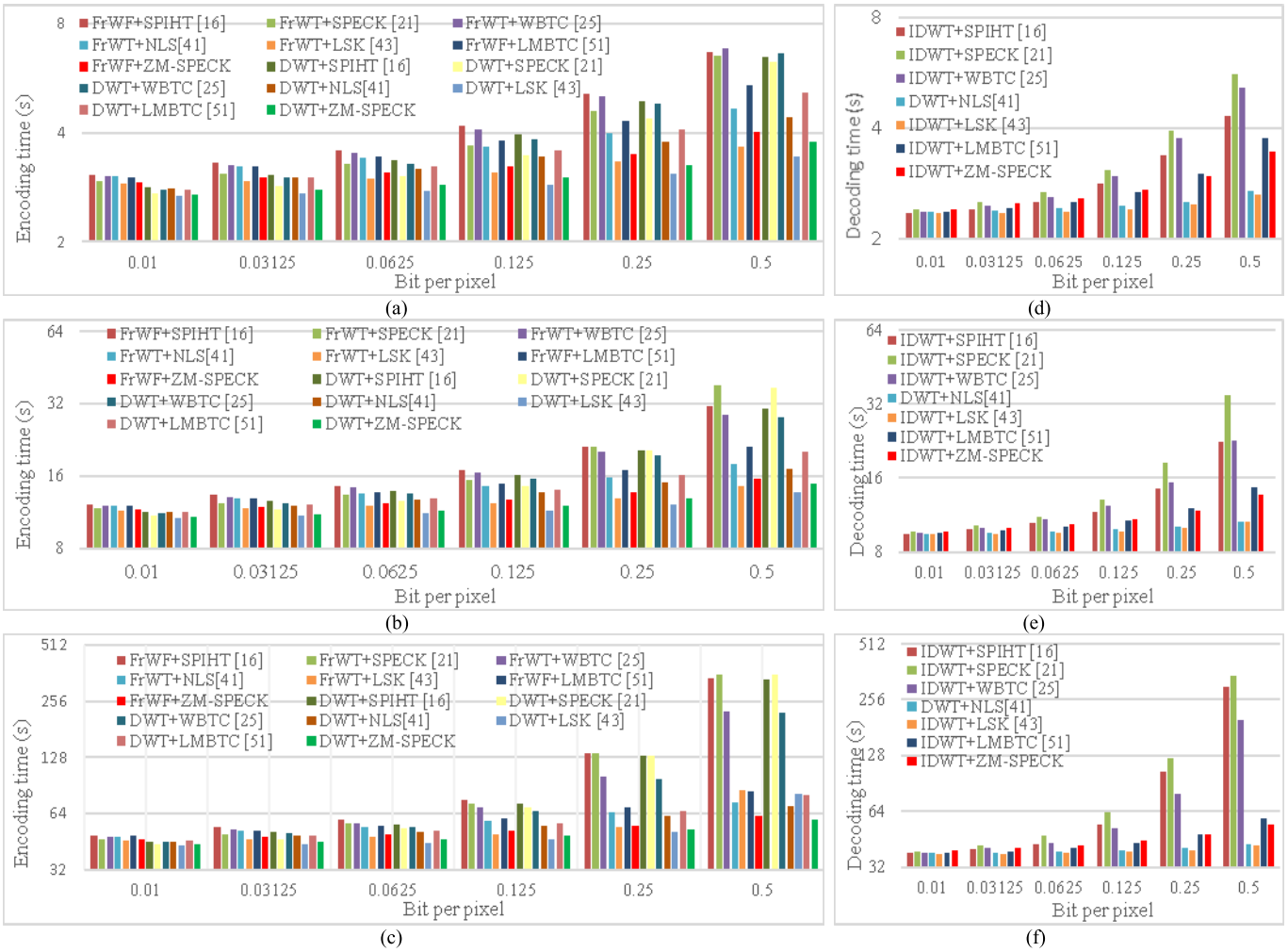


Fig. 5. Coding complexity measured in terms of time for ZM-SPECK, SPECK, SPIHT, WBTC, NLS, LSK, and LMBTC coder with FrWF and DWT. Encoding time for image dimensions (a) 256×256, (b) 512×512, and (c) 1024×1024 pixels. Decoding time for image dimensions (d) 256×256, (e) 512×512, and (f) 1024×1024 pixels.

From Table V we also observe that ZM-SPECK outperforms the other algorithms in terms of the coding memory requirement. This is because ZM-SPECK does not require any static (for state tables/markers) or dynamic (for linked lists) memory to store the significance states of coefficients or sets. ZM-SPECK achieves essentially zero coding memory requirements, irrespective of image size and bit rate. On the other hand, the memory requirements of SPIHT, SPECK, and WBTC grow with increasing bit-rate and image size. Their listless versions, namely NLS, LSK, and LMBTC, respectively, require fixed-size static memory depending on image size, but independent of the bit-rate. Although ZM-SPECK does not require any memory to store state information in the form of state tables, markers or linked lists, its execution requires some program variables. Measurements indicated that the ZM-SPECK program variables occupy approximately 44 bytes of memory. Although the memory required by the Wi2l coder is quite low (independent of bit rate, but dependent on image size) [48], the Wi2l coder generates a non-embedded bit-stream, making it unsuitable for scalable image transmission over heterogeneous networks.

The overall memory requirement of an image coder with sequential transform and coding stages is the maximum of the memory required at the individual stages. Therefore, the memory requirement of the ZM-SPECK-based image coder equals the memory required for the wavelet transform. From Table V, we observe that for low resolution images (256×256 and 512×512), the overall memory requirements of the FrWF-based Wi2l and LMBTC codecs are the same as those of the FrWF-based ZM-SPECK. For high resolution (1024×1024) images, the ZM-SPECK and Wi2l based image coders require the least overall memory (of the order of 9 kB) compared to all other coders. However Wi2l is a non-embedded coder, which is not suitable for scalable image network transmission.

From these results, it is evident that considering the overall memory requirement of an image coder (maximum of transform memory and coding algorithm's memory), FrWF combined with the ZM-SPECK coding algorithm consumes the least memory among all other options. Even for higher resolution (1024×1024), the overall memory requirement of FrWF + ZM-SPECK is of the order of 9 kB, and therefore

can be considered as feasible option for its implementation on low-cost memory-constrained portable multimedia devices and VSNs.

One may question the usefulness of the proposed ZM-SPECK coder, as it does not significantly reduce the overall memory requirements for low resolution images compared to the FrWF-based Wi2l and LMBTC coders, especially considering the fact that it is targeted for portable multimedia devices and sensor networks (where low resolution images are dominant). However, we believe that the ZM-SPECK coder is an important advance in low-memory image coding as it achieves an absolute zero coding memory requirement. Thus, any future advances in reducing the transform stage memory requirement, when combined with ZM-SPECK, will immediately result in reduced overall image coder memory requirements.

C. Complexity Analysis

We evaluate the computational complexity (encoding and decoding speeds) of the proposed ZM-SPECK codec by estimating the computational time required for encoding the transformed coefficients (obtained with FrWF and traditional DWT) and decoding the bit-stream at different bit-rates. The encoding time is the total time required for calculating the transform as well as the time required for encoding the transformed coefficients. The decoding time at a given bit-rate is the time required for decoding the corresponding number of bits in a bit-stream as well as reconstructing the image. The complexity of the ZM-SPECK codec is compared with other state-of-art-coders, such as SPIHT, SPECK, WBTC, NLS, LSK, and LMBTC, in terms of encoding and decoding times in Fig. 5. The encoding and decoding times are measured on a Windows 8.1 Netbook with Intel atom CPU Z 3735F @ 1.33 GHz having 2 GB RAM and a 32 GB MMC card.

From Figure 5, it is evident that the FrWF based ZM-SPECK encoder has the lowest complexity in comparison to other state-of-the-art coders, but slightly higher complexity than LSK [43]. This saving in encoding time is due to the fact that ZM-SPECK does not use any lists or markers, therefore avoiding multiple memory accesses. That is, the time involved in reading and writing of memory is much more than the time required for computing the set size in ZM-SPECK.

Comparing the decoding times, we observe that ZM-SPECK has shorter decoding times compared to SPIHT and SPECK, but has longer decoding time compared to NLS and LSK. This is due to the fact that the ZM-SPECK decoder conducts additional significance tests to identify the sets with refinement coefficients. Therefore, the computational complexity of the ZM-SPECK decoder is comparable to that of the ZM-SPECK encoder.

V. Conclusion

In order to meet the constraints of low-cost visual sensor nodes and other portable multimedia devices, a low-memory image codec is required. In this paper we have proposed a novel FrWF-based ZM-SPECK image coder, which reduces the memory required at the transform stage by using the FrWF, and reduces the memory required at the coding stage through a novel Zero Memory SPECK (ZM-SPECK) algorithm. Simulation results demonstrate that the proposed FrWF-based ZM-SPECK coder requires very low transform memory and zero state memory for encoding/decoding of coefficients. Moreover, ZM-SPECK has reduced computational complexity compared to the original SPECK algorithm, while preserving the SPECK coding efficiency. Due to these features, the proposed coder is suitable for resource-constrained devices, such as portable cameras, PDAs, and sensor nodes of wireless multimedia sensor networks (WMSNs). The FrWF-based ZM-SPECK coder is fast in comparison to other state-of-the-art coders at the expense of increased decoding complexity. Sensor nodes in WMSN are extremely resource constrained while sensor hubs have more resources; therefore lower transform and encoding memory and lower encoding complexity with increased decoding complexity make ZM-SPECK a suitable candidate for implementation in sensor nodes.

The ZM-SPECK coding algorithm presented in this article has important implications for future research on low-complexity multimedia processing in sensor nodes: Future research on low-memory image coding should focus on reducing the memory requirements of the wavelet transform. Since ZM-SPECK requires zero memory for coding the wavelet coefficients, any reductions in transform memory requirements will immediately translate into reductions of the memory required for the overall image coder.

References

[1] I. F. Akyildiz, T. Melodia, and K. R. Chowdhury, "Wireless multimedia sensor networks: Applications and testbeds," *Proc. IEEE*, vol. 96, no. 10, pp. 1588–1605, Oct. 2008.

[2] T. Melodia and I. F. Akyildiz, "Research challenges for wireless multimedia sensor networks," in *Distributed Video Sensor Networks*. London, U.K.: Springer, 2011, pp. 233–246.

[3] A. Seema and M. Reisslein, "Towards efficient wireless video sensor networks: A survey of existing node architectures and proposal for a Flexi-WVSNP design," *IEEE Commun. Surveys Tuts.*, vol. 13, no. 3, pp. 462–486, Third Quarter 2011.

[4] B. Tavli, K. Bicakci, R. Zilan, and J. M. Barcelo-Ordinas, "A survey of visual sensor network platforms," *Multimedia Tools Appl.*, vol. 60, no. 3, pp. 689–726, Oct. 2012.

[5] A. Sharif, V. Potdar, and E. Chang, "Wireless multimedia sensor network technology: A survey," in *Proc. 7th IEEE Int. Conf. Ind. Inform. (INDIN)*, Jun. 2009, pp. 606–613.

[6] S. Rein and M. Reisslein, "Low-memory wavelet transforms for wireless sensor networks: A tutorial," *IEEE Commun. Surveys Tuts.*, vol. 13, no. 2, pp. 291–307, Second Quarter 2011.

[7] A. Chefi, A. Soudani, and G. Sicard, "Hardware compression scheme based on low complexity arithmetic encoding for low power image transmission over WSNs," *AEU-Int. J. Electron. Commun.*, vol. 68, no. 3, pp. 193–200, Mar. 2014.

[8] M. Nasri, A. Helali, H. Sghaier, and H. Maaref, "Energy-efficient wavelet image compression in wireless sensor network," in *Proc. Int. Conf. Commun. Wireless Environ. Ubiquitous Syst., New Challenges (ICWUS)*, Sousse, Tunisia, 2010, pp. 1–7.

[9] T. Ma, M. Hempel, D. Peng, and H. Sharif, "A survey of energy-efficient compression and communication techniques for multimedia in resource constrained systems," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 3, pp. 963–972, Third Quarter 2013.

[10] D.-U. Lee, H. Kim, M. Rahimi, D. Estrin, and J. D. Villasenor, "Energy-efficient image compression for resource-constrained platforms," *IEEE Trans. Image Process.*, vol. 18, no. 9, pp. 2100–2113, Sep. 2009.

[11] B.-S. Chow, "A limited resources-based approach to coding for wireless video sensor networks," *IEEE Sensors J.*, vol. 9, no. 9, pp. 1118–1124, Sep. 2009.

[12] E. J. Tan, Z. Ignjatovic, M. F. Bocko, and P. P. K. Lee, "Non-uniformly tiled CMOS image sensors for efficient on-chip image compression," *IEEE Sensors J.*, vol. 12, no. 8, pp. 2655–2663, Aug. 2012.
[13] D. Santa-Cruz, R. Grosbois, and T. Ebrahimi, "JPEG 2000 performance evaluation and assessment," *Signal Process., Image Commun.*, vol. 17, no. 1, pp. 113–130, Jan. 2002.
[14] A. S. Lewis and G. Knowles, "Image compression using the 2-D wavelet transform," *IEEE Trans. Image Process.*, vol. 1, no. 2, pp. 244–250, Apr. 1992.
[15] J. M. Shapiro, "Embedded image coding using zerotrees of wavelet coefficients," *IEEE Trans. Signal Process.*, vol. 41, no. 12, pp. 3445–3462, Dec. 1993.
[16] A. Said and W. A. Pearlman, "A new, fast, and efficient image codec based on set partitioning in hierarchical trees," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, no. 3, pp. 243–250, Jun. 1996.
[17] Z. Xiong, K. Ramchandran, and M. T. Orchard, "Space-Frequency quantization for wavelet image coding," *IEEE Trans. Image Process.*, vol. 6, no. 5, pp. 677–693, May 1997.
[18] E. Khan and M. Ghanbari, "Very low bit rate video coding using virtual SPIHT," *IEE Electron. Lett.*, vol. 37, no. 1, pp. 40–42, Jan. 2001.
[19] J. Andrew, "A simple and efficient hierarchical image coder," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, vol. 3. Oct. 1997, pp. 658–661.
[20] A. Islam and W. A. Pearlman, "Embedded and efficient low-complexity hierarchical image coder," *Proc. SPIE*, vol. 3653, pp. 294–305, Jan. 1999.
[21] 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.*, vol. 14, no. 11, pp. 1219–1235, Nov. 2004.
[22] C. Chrysafis, A. Said, A. Drukarev, A. Islam, and W. A. Pearlman, "SBHP—A low complexity wavelet coder," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, vol. 4, Jun. 2000, pp. 2035–2038.
[23] D. Taubman, "High performance scalable image compression with EBCOT," *IEEE Trans. Image Process.*, vol. 9, no. 7, pp. 1158–1170, Jul. 2000.
[24] S.-T. Hsiang and J. W. Woods, "Embedded image coding using zeroblocks of subband/wavelet coefficients and context modeling," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, vol. 3. May 2000, pp. 662–665.
[25] A. A. Moinuddin, E. Khan, and M. Ghanbari, "Efficient algorithm for very low bit rate embedded image coding," *IET Image Process.*, vol. 2, no. 2, pp. 59–71, Apr. 2008.
[26] J. Oliver and M. P. Malumbres, "Low-complexity multiresolution image compression using wavelet lower trees," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 16, no. 11, pp. 1437–1444, Nov. 2006.
[27] L. W. Chew, L.-M. Ang, and K. P. Seng, "Survey of image compression algorithms in wireless sensor networks," in *Proc. Int. Symp. Inf. Technol.*, Kuala Lumpur, Malaysia, Aug. 2008, pp. 1–9.
[28] R. Sudhakar, R. Karthiga, and S. Jayaraman, "Image compression using coding of wavelet coefficients—A survey," *ICGST-GVIP J.*, vol. 5, no. 6, pp. 25–38, Jun. 2005.
[29] C. Chrysafis and A. Ortega, "Line-based, reduced memory, wavelet image compression," *IEEE Trans. Image Process.*, vol. 9, no. 3, pp. 378–389, Mar. 2000.
[30] J. Oliver and M. Perez Malumbres, "On the design of fast wavelet transform algorithms with low memory requirements," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 18, no. 2, pp. 237–248, Feb. 2008.
[31] C.-H. Yang, J.-C. Wang, J.-F. Wang, and C.-W. Chang, "A block-based architecture for lifting scheme discrete wavelet transform," *IEICE Trans. Fundam. Electron., Commun. Comput. Sci.*, vol. E90-A, no. 5, pp. 1062–1071, May 2007.
[32] Y. Bao and C.-C. J. Kuo, "Design of wavelet-based image codec in memory-constrained environment," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, no. 5, pp. 642–650, May 2001.
[33] L. W. Chew, W. C. Chia, L.-M. Ang, and K. P. Seng, "Very low-memory wavelet compression architecture using strip-based processing for implementation in wireless sensor networks," *EURASIP J. Embedded Syst.*, vol. 2009, no. 479281, pp. 1–16, Dec. 2009.
[34] L. W. Chew, W. C. Chia, L.-M. Ang, and K. P. Seng, "Low-memory video compression architecture using strip-based processing for implementation in wireless multimedia sensor networks," *Int. J. Sensor Netw.*, vol. 11, no. 1, pp. 33–47, Jan. 2012.
[35] W. C. Chia, L. W. Chew, L.-M. Ang, and K. P. Seng, "Low memory image stitching and compression for WMSN using strip-based processing," *Int. J. Sensor Netw.*, vol. 11, no. 1, pp. 22–32, Jan. 2012.
[36] S. Rein, S. Lehmann, and C. Gühmann, "Fractional wavelet filter for camera sensor node with external Flash and extremely little RAM," in *Proc. ACM Mobile Multimedia Commun. Conf. (MobiMedia)*, Jul. 2008, pp. 1–7.
[37] H. Arora, P. Singh, E. Khan, and F. Ghani, "Memory efficient set partitionning in hierarchical tree (MESH) for wavelet image compression," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, vol. 2, Mar. 2005, pp. 385–388.
[38] E. Khan, I. A. Arshad, and T. Varshney, "An error resilient and memory efficient scheme for wavelet image coding," *J. Appl. Quant. Methods*, vol. 5, no. 2, pp. 350–357, Jun. 2010.
[39] M. Akter, M. B. I. Reaz, F. Mohd-Yasin, and F. Choong, "A modified-set partitioning in hierarchical trees algorithm for real-time image compression," *J. Commun. Technol. Electron.*, vol. 53, no. 6, pp. 642–650, Jul. 2008.
[40] W.-K. Lin and N. Burgess, "Listless zerotree coding for color images," in *Proc. 32nd Asilomar Conf. Signals, Syst. Comput.*, vol. 1, Nov. 1998, pp. 231–235.
[41] F. W. Wheeler and W. A. Pearlman, "SPIHT image compression without lists," in *Proc. IEEE Conf. Acoust., Speech Signal Process.*, vol. 4. Jun. 2000, pp. 2047–2050.
[42] H. Pan, W.-C. Siu, and N.-F. Law, "A fast and low memory image coding algorithm based on lifting wavelet transform and modified SPIHT," *Signal Process., Image Commun.*, vol. 23, no. 3, pp. 146–161, 2008.
[43] M. V. Latte, N. H. Ayachit, and D. K. Deshpande, "Reduced memory listless SPECK image compression," *Digit. Signal Process.*, vol. 16, no. 6, pp. 817–824, Nov. 2006.
[44] N. R. Kidwai, M. Alam, E. Khan, and R. Beg, "A efficient memory no list set partitioned embedded block (NLSK) wavelet image coding algorithm for low memory devices," *Int. J. Signal Process., Image Process. Pattern Recognit.*, vol. 5, no. 4, pp. 93–106, Dec. 2012.
[45] R. K. Senapati, U. C. Pati, and K. K. Mahapatra, "Listless block-tree set partitioning algorithm for very low bit rate embedded image compression," *AEU-Int. J. Electron. Commun.*, vol. 66, no. 12, pp. 985–995, Dec. 2012.
[46] N. R. Kidwai, E. Khan, and R. Beg, "A memory efficient listless SPECK (MLSK) image compression algorithm for low memory applications," *Int. J. Adv. Res. Comput. Sci.*, vol. 3, no. 4, pp. 209–215, Jul./Aug. 2012.
[47] N. R. Kidwai, E. Khan, and R. Beg, "A memory efficient no list SPECK (NSK) wavelet image coder for memory-constrained applications," *J. Remote Sens. GIS (STM J.)*, vol. 3, no. 3, pp. 1–16, Dec. 2012.
[48] S. Rein, S. Lehmann, and C. Guhmann, "Wavelet image two-line coder for wireless sensor node with extremely little RAM," in *Proc. IEEE Data Compress. Conf. (DCC)*, Mar. 2009, pp. 252–261.
[49] L. Ye, J. Guo, B. Nutter, and S. Mitra, "Memory-efficient image codec using line-based backward coding of wavelet trees," in *Proc. IEEE Data Compress. Conf. (DCC)*, Mar. 2007, pp. 213–222.
[50] L. Ye, J. Guo, B. Nutter, and S. Mitra, "Low-memory-usage image coding with line-based wavelet transform," *Opt. Eng.*, vol. 50, no. 2, pp. 027005-1–027005-11, Feb. 2011.
[51] M. Tausif, N. R. Kidwai, E. Khan, and M. Reisslein, "FrWF-based LMBTC: Memory-efficient image coding for visual sensors," *IEEE Sensors J.*, vol. 15, no. 11, pp. 6218–6228, Nov. 2015.
[52] N. R. Kidwai, M. Alam, E. Khan, and R. Beg, "A fast and memory efficient wavelet based set partitioned embedded block image coding algorithm," in *Proc. Int. Conf. Multimedia, Signal Process. Commun. Technol. (IMPACT)*, Dec. 2011, pp. 320–323.
[53] S. Rein and M. Reisslein, "Performance evaluation of the fractional wavelet filter: A low-memory image wavelet transform for multimedia sensor networks," *Ad Hoc Netw.*, vol. 9, no. 4, pp. 482–496, Jun. 2011.
[54] G. Seetharaman, B. Zavidovique, and S. Shivayogimath, "Z-trees: Adaptive pyramid-algorithms for image segmentation," in *Proc. IEEEE Int. Conf. Image Process. (ICIP)*., vol. 3, Oct. 1998, pp. 294–298.
[55] N. R. Kidwai, "Efficient image coding for wireless sensor networks," Ph.D. dissertation, Dept. Electron. Commun. Eng., Integral Univ., Lucknow, India, Apr. 2014.
[56] G. Bjøntegaard, *Calculation of Average PSNR Differences Between RD-Curves*, document VCEG-M33, Technical Report Video Coding Experts Group (VCEG), International Telecommunication Union-Telecommunications Standardization Sector (ITU-T), Apr. 2001.

Naimur Rahman Kidwai received the B.Sc.(Engg.) degree in electronics engineering from the Zakir Hussain College of Engineering and Technology, Aligarh Muslim University, India, in 1996, the M.Tech. degree in digital communication from Uttar Pradesh Technical University, Lucknow, India, in 2006, and the Ph.D. degree from Integral University, Lucknow, in 2014. He is currently an Associate Professor with the Department of Electronics and Communication, Integral University.

Ekram Khan (SM'12) received the B.Sc.(Engg.) and M.Sc.(Engg.) degrees from Aligarh Muslim University (AMU), Aligarh, India, in 1991 and 1994, respectively, and the Ph.D. degree from the University of Essex, Colchester, U.K., in 2003, all in electronics engineering. He joined the Department of Electronics Engineering, AMU, in 1993 where he has been a Professor since 2009. His areas of research are low-complexity image/video coding, video transmission over wireless networks, and biomedical image processing. He is a Life Member of the Institution of Electronics and Telecommunication Engineers, India, and Systems Society of India.

Martin Reisslein (A'96–S'97–M'98–SM'03–F'14) received the Ph.D. degree in systems engineering from the University of Pennsylvania in 1998. He is currently a Professor with the School of Electrical, Computer, and Energy Engineering, Arizona State University, Tempe.