FrWF-Based LMBTC: Memory-Efficient Image Coding for Visual Sensors

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

Abstract—**After the successful development of JPEG2000, many state-of-the-art wavelet-based image coding algorithms have been developed. However, the traditional discrete wavelet transform (DWT) is implemented with memory intensive and time-consuming algorithms and, therefore, has very high system resource requirements. In particular, the very large requirement of memory poses a serious limitation for multimedia applications on memory-constrained portable devices, such as digital cameras and sensor nodes. In this paper, we propose a novel wavelet-based image coder with low memory requirements and low complexity that preserves the compression efficiency. Our encoder employs the fractional wavelet filter (FrWF) to calculate the DWT coefficients, which are quantized and encoded with a novel low memory block tree coding (LMBTC) algorithm. The LMBTC is a listless form of the wavelet block tree coding algorithm. Simulation results demonstrate that the proposed coder significantly reduces memory requirements and computational complexity and has competitive coding efficiency in comparison with other state-of-the-art coders. The FrWF combined with the LMBTC is, thus, a viable option for image communication over wireless sensor networks.**

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

I. Introduction

A. Motivation

Wireless Sensor Networks (WSNs) consist of a group of geographically distributed, low-power sensors, which monitor physical or environmental conditions [1]. Recent advances in micro-electromechanical systems, wireless communication technology, and low-cost digital imaging cameras have made it conceivable to build a wireless network of visual sensors (VSs) [2]–[6], called visual sensor network (VSN). Inside a VSN, each VS node has the ability to acquire, compress, and transmit visual data to the base station, also called sink. Visual sensor nodes with multimedia capabilities can ubiquitously retrieve and transmit video, audio, and image data from the environment. The development of such networks was originally motivated by military applications, such as battlefield surveillance. However, these networks are now used in many civilian applications, including environment and habitat monitoring, healthcare applications, home automation, and traffic control. Multimedia contents, especially images and videos require extensive bandwidth for transmission. Due to the limited available bandwidth, an image captured by a sensor node typically needs to be processed and compressed before transmission [7]–[13].

Efficient image compression techniques mainly rely on removing the redundant information from the raw data. A typical image coder consists of a transform (discrete cosine transform (DCT) or discrete wavelet transform (DWT)) block, as well as quantization and entropy coding blocks. Some modern image coders combine the quantization and encoding blocks into a single unit. In order to implement each of these blocks some on-chip memory is required. Depending on the image resolution and algorithmic complexity, often a large amount of working memory is required to compress the images. Due to space and energy restrictions and the high cost of providing large amounts of memory, on-chip memory available on sensor nodes is typically limited and has become a major constraint for the processing of large images [14], [15].

A wireless sensor network consists of a large number of sensor nodes. The cost of a single node is very important to justify the overall cost of the network. Low-cost sensor nodes are generally used for environmental monitoring or object tracking applications. These nodes have limited resources in terms of processing power and memory. The on-chip random access memory (RAM) of most low-cost sensor nodes is of the order of 10 kB [16]. In order to equip these nodes with a visual sensor (camera) to capture visual information and to transmit them over WSNs, memory-efficient and low-complexity image codecs are needed [17]–[20].

The memory requirements of the transform and quantization/encoding stages need to be minimized in order to encode images on low-memory sensors. Since JPEG2000 [21] was developed, many efficient DWT-based image coders have been developed, e.g., [22]–[29]. However, most of them require very large amounts of dynamic memory (RAM), restricting their uses for low-power portable devices. Most of the existing research on memory efficient image coders either tries to minimize the memory requirement of the DWT stage [16], [30]–[34] or that of the quantization/encoding stage [35]–[47].

Manuscript received February 5, 2015; accepted July 4, 2015. Date of publication July 14, 2015; date of current version August 26, 2015. The associate editor coordinating the review of this paper and approving it for publication was Dr. David Hecht. *(Corresponding author: Martin Reisslein.)*

M. Tausif was with the Department of Electronics Engineering, Aligarh Muslim University, Aligarh 202002, India (e-mail: mohdtausif32@gmail.com).

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 Department of Electronics Engineering, Aligarh Muslim University, Aligarh 202002, India (e-mail: ekhan.el@amu.ac.in).

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

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JSEN.2015.2456332

1530-437X © 2015 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

B. Related Work

1) Low-Memory DWT: The traditional method of computing the DWT of images requires large memory. In order to reduce the memory requirement, a line-based version of the DWT was proposed in [30]. A line-based DWT using the lifting scheme is also proposed in [31] and [48]. The line-based approach requires 26 kB of RAM for a six level transform of a 512 × 512 gray scale image [16]. Another approach to reduce memory is to apply the DWT on a block-by-block basis, rather than on the entire image [32], [33]. However, the block-based approach requires almost the same memory as the line-based approach [16]. Strip-based low memory image and video coding architectures are suggested in [49] and [50] for wireless sensor networks applications. Recently, the Fractional Wavelet Filter (FrWF) has been proposed to compute the DWT of images [34], which requires memory of the order of 2.304 kB to calculate the DWT of an image of size 256 × 256 pixels. To the best of our knowledge, only limited efforts, e.g., [51], have been made to minimize the overall memory requirements of image codecs.

2) Low-Memory Quantization and Encoding: For quantizing and/or encoding DWT coefficients, most algorithms exploit the fact that the majority of the coefficients are centered around zero and very few coefficients have large values. This means that most of the information is concentrated in a small fraction of the coefficients and therefore the image can be compressed efficiently. Thus, the success of a wavelet-based image coding algorithm depends on the exploitation of this energy clustering property of the wavelet transform. Over the years, a number of successful wavelet-based image coding algorithms have been proposed. The state-of-the-art image coding algorithms, such as embedded block coding with optimized truncation (EBCOT) [28] (used in JPEG2000 [21]), embedded zero tree of wavelet coefficients (EZW) [22], set partitioning in hierarchical trees (SPIHT) [23], set partitioned embedded block coding (SPECK) [24], wavelet block tree coding (WBTC) [25], sub-band block hierarchical partitioning (SBHP) [27], and embedded zero block coding (EZBC) [29] support a wide range of functionalities but either have a very high computational complexity (e.g., EBCOT), or very high data-dependent memory requirement (e.g., SPIHT, SPECK, WBTC, SBHP, and EZBC). A common feature of these algorithms is that they use data-dependent lists to keep track of already coded and yet to be coded transform coefficients. Due to these high memory requirements, these coders are typically unsuitable for WSNs.

In order to reduce the memory requirement of image coding algorithms, listless significance map coding was initially proposed by Lin and Burgess in their work on the Listless Zero-tree Coding (LZC) for color images [36]. LZC avoids the use of variable data-dependent lists by using fixed-size state tables or markers. The markers are placed in state memory and are updated as per partitioning decisions in the coding process. A number of low-memory versions of SPIHT have been developed in recent years [35], [37]–[39]. Among these, No List SPIHT (NLS) [38] is very popular. NLS uses 4 bit per coefficient marker. Listless versions of the SPECK algorithm have been proposed in [40] and [41]. Among low-memory versions of the SPECK algorithm, Listless SPECK (LSK) [40] is very popular and uses markers with 2 bit per coefficient. The No List SPECK coder (NLSK) [41] uses 0.75 bit per coefficient state memory to keep track of blocks and coefficients to be tested for their significance. An improved LSK (ILSK) is proposed in [42] to encode discrete Tchebichef transformed (DTT) coefficients. It uses a single array of memory and requires less memory than LSK at lower bit-rates (early coding passes), but its memory requirement increases with the bit-rate. Recently, listless versions of WBTC, called Listless Block Tree Coder (LBTC) [43] and modified wavelet block tree coding (MLBTC) [44], have been proposed. Though these coders use fixed-size memory, their memory requirements are quite high.

A backward version of SPIHT, called Backward Coding of Wavelet Trees (BCWT) has been reported in [45]. A low-memory version of BCWT, named as line-based BCWT has been proposed in [46] and [47]. Line-based BCWT requires less memory than BCWT, while it has the same coding efficiency as BCWT. Although, the listless implementations of state-of-the-art image coders result in significant memory reduction compared to their counterparts, the memory requirement is still very high for low-cost sensor nodes.

The Wavelet image two line coder (Wi2l) [51], which combines FrWF with line-based BCWT, has made significant progress towards low-memory image coding. However, the drawback of the Wi2l coder is that it generates a non-embedded bit-stream. That is, with the Wi2l coder, the lower bit rate (coarser quality) encodings are not necessarily embedded at the beginning of the bit stream for a given target bit rate. Embedded coders are highly desirable for flexible image coding and transmission in heterogeneous networks [22]. A few scalable extensions of the BCWT algorithm have been proposed in the literature [52]–[55], however at the cost of reduced rate-distortion performance or increased complexity. Thus, there is a need to design efficient, feature-rich and low-memory embedded image codecs for visual sensor nodes.

C. Contribution of This Paper

In this paper, we propose a novel low-memory block tree coding (LMBTC) algorithm, which is a listless form of WBTC. Further, to reduce the overall memory requirements, we combine the proposed LMBTC image coder with FrWF. The memory requirement at the transform stage is reduced by the FrWF and the memory requirement of the quantization and entropy coding stage is reduced by the LMBTC algorithm. The proposed coder generates an embedded bit-stream. Therefore, the proposed codec is suitable for low-memory devices. To the best of our knowledge, the proposed coder requires the least memory among the available state-of-art wavelet-based image coding algorithms, while retaining the coding efficiency and scalability feature. Simulation results demonstrate that the coding efficiency and computational complexity of the proposed LMBTC algorithm are at par with other state-of-the-art image coders.

The rest of the paper is organized as follows. Section II presents an overview of the FrWF and WBTC algorithms.

In Section III, we introduce the novel LMBTC image coder. Simulation results and related discussions are presented in Section IV and finally the paper is concluded in Section V.

II. Background

In this section the FrWF scheme to compute the DWT of images and the WBTC algorithm are briefly reviewed.

A. Fractional Wavelet Filter (FrWF)

A recent technique known as the Fractional Wavelet Filter (FrWF) has been proposed in [34] to compute the DWT of images. For an image of size $N \times N$, the FrWF uses three buffers, each of dimensions N. The buffers are, 's' for the current input line, LL_HL for the LL/HL sub-band destination line, and LH_HH for the LH/HH sub-band destination line [56]. The memory (in Bytes) required for the wavelet transform of an $N \times N$ image with 'level' decomposition levels using floating-point and fixed-point arithmetic [16] are:

$$Bytes_{float} = \begin{cases} 9N, & level = 1 \\ \dfrac{12N}{level}, & level > 1 \end{cases} \quad (1)$$

$$Bytes_{fixed} = \begin{cases} 5N, & level = 1 \\ \dfrac{6N}{level}, & level > 1. \end{cases} \quad (2)$$

B. Overview of WBTC Algorithm

Since the proposed coding algorithm is based on the WBTC algorithm [25], this section presents an overview of the WBTC coding algorithm and related terminology. WBTC is a wavelet-based image coding algorithm which exploits redundancies among sub-bands, along with partial exploitation of redundancies within sub-bands. Similar to SPIHT, WBTC is a bit plane coding algorithm where magnitude and bit plane ordering of the coefficients are used such that first transmitted coefficients yield the largest decrease in mean squared error distortion.

Consider an image X of size $R \times C$ pixels that after N_{d} levels of wavelet transformation exhibits a pyramidal sub-band structure. The transformed image is represented by an indexed set of transform coefficients $c_{i,j}$ with row index i and column index j. The coefficients are grouped in blocks of size $m \times n$ coefficients and then block-trees are formed with roots in the topmost (LL) sub-band. A block-tree is a tree of all descendent blocks of a root block. This approach has three distinct advantages over SPIHT. First, it combines many clustered zero-trees of SPIHT, which may occur in the early passes, thus creating zero-trees with more elements. Secondly, intra-sub-band correlations can be partially exploited. Thirdly, because of its block-based nature, compared to pixel-based techniques, the memory requirement for storing the lists and the encoding time are significantly reduced.

Except for the lowest and highest resolution bands, each block has four offspring blocks that correspond to the same spatial orientation in the higher frequency sub-bands. In the LL-band, out of each group of 2×2 blocks, one (top-left) block has no descendent, and each of the other three blocks has four offspring blocks in the high frequency sub-bands of their corresponding orientations. By creating a block-tree, many of SPIHT's spatial orientation trees (SOTs) are combined into a single spatial orientation block tree (SOBT). A set of all descendent blocks is referred as type 'A' block-tree, while the set of grand descendent blocks (set of descendent blocks minus set of offspring blocks) is referred as type 'B' block-tree. In particular, for a block size of 2×2, four SOTs of SPIHT are combined into a single WBTC SOT. Significant information is stored in three ordered lists: a list of insignificant blocks (LIB), a list of insignificant block sets (LIBS), and a list of significant pixels (LSP). At the initialization step, the blocks in the LL-band are added to the LIB, and those with descendants are added to the LIBS as type 'A' entries. The LSP starts as an empty list. Similar to SPIHT and SPECK, WBTC is a bit-plane based coding algorithm comprising of two main stages (passes) within each bit-plane: the sorting and refinement passes.

The coding process starts with the most significant bit plane and proceeds towards the finest resolution. During the sorting pass, the encoder first traverses through the LIB, testing the significance of a block against the current threshold. For each block in the LIB, one bit is used to describe its significance. If the block is not significant, then it is a zero-block and a '0' is sent, it remains in the LIB and no more bits will be generated. Here, insignificant information of $m \times n$ individual coefficients is conveyed using a single '0' bit, whereas SPIHT generates $m \times n$ '0' bits. This is how WBTC partially exploits intra-sub-band correlation. Otherwise, if the block is significant, then the block is a non-zero block and a '1' is sent. A significant block is partitioned into four adjacent blocks (quad-tree partitioning). The division operation is recursively repeated until no further division is needed or the smallest possible block size (individual coefficient) is attained. At this stage, four coefficients and their significance are individually tested. If a coefficient is insignificant, then a '0' is sent and the coefficient is moved to the LIB as a single coefficient block. Otherwise, if a coefficient is significant, then a '1' is sent and its sign bit is also coded and the coefficient is moved to LSP. After testing all four individual coefficients in the block, the current block is deleted from the LIB. The encoder then examines the LIBS and performs a significance test on each set. Insignificant sets remain in the LIBS, while the significant sets are partitioned into subsets.

A significant type 'A' set with root block $B_{k,l}^{m,n}$ is partitioned into a type 'B' set $L_{k,l}^{m,n}$ and four offspring blocks $O_{k,l}^{m,n}$. The type 'B' set is added to the end of the LIBS while the four offspring blocks (each of the same size as the root block $B_{k,l}^{m,n}$) are immediately examined for their significance in the same manner as if they were in the LIB. Here, any zero-block will result in bit savings. A significant type 'B' set is partitioned into four type 'A' sets; all of them are added to the end of the LIBS. Since all the newly generated insignificant sets are added to the end of the LIBS, they are all processed in the same manner for a given threshold until each set is examined.

After each sorting pass, the coefficients in the LSP, except those added in the current bit plane, are refined with one bit. The algorithm then repeats the above procedure by decreasing the current threshold level by a factor of two until the desired bit rate is achieved. Similar to SPIHT, WBTC provides an embedded bit-stream that allows bit rate scalability

TABLE I
SET STRUCTURES IN TWO DIMENSIONAL ARRAY (2D INDEXING) AND LINEAR ARRAY (LINEAR INDEXING)

λ level dyadic wavelet transformed two dimensional array X of size (*R* x *C*)	λ level dyadic wavelet transformed linear array $\Im$ of length $N_{\text{pix}}=RC$
m x *n*	$\beta = mn$
$B_{k,l}^{m,n} = \{c_{i,j} \vdots k \le i < k+m, l \le j < l+m\}$	$B_k^{\beta} = \{c_i \vdots k \le i < k+\beta\}$
$Q_{k,l}^{m,n} = \left\{ B_{2k,2l}^{\frac{m}{2},\frac{n}{2}}, B_{2k+\frac{m}{2},2l}^{\frac{m}{2},\frac{n}{2}}, B_{2k,2l+\frac{n}{2}}^{\frac{m}{2},\frac{n}{2}}, B_{2k+\frac{m}{2},2l+\frac{n}{2}}^{\frac{m}{2},\frac{n}{2}} \right\}$	$Q_k^{\beta} = \left\{ B_k^{\frac{\beta}{4}}, B_{k+\frac{\beta}{4}}^{\frac{\beta}{4}}, B_{k+\frac{2\beta}{4}}^{\frac{\beta}{4}}, B_{k+\frac{3\beta}{4}}^{\frac{\beta}{4}} \right\}$
$D_{k,l}^{m,n} = \{O_{k,l}^{m,n}, L_{k,l}^{m,n}\}$	$D_k^{\beta} = \{O_k^{\beta}, L_k^{\beta}\}$
$O_{k,l}^{m,n} = \{B_{k,2l}^{m,n}, B_{k+m,2l}^{m,n}, B_{k,2l+n}^{m,n}, B_{k+m,2l+n}^{m,n}\}$ horizontal SOBT $O_{k,l}^{m,n} = \{B_{2k,l}^{m,n}, B_{2k+m,l}^{m,n}, B_{2k,l+n}^{m,n}, B_{2k+m,l+n}^{m,n}\}$ vertical SOBT $O_{k,l}^{m,n} = \{B_{2k,2l}^{m,n}, B_{2k+m,2l}^{m,n}, B_{2k,2l+n}^{m,n}, B_{2k+m,2l+n}^{m,n}\}$ diagonal SOBT	$O_k^{\beta} = \{B_{4k}^{\beta}, B_{4k+\beta}^{\beta}, B_{4k+2\beta}^{\beta}, B_{4k+3\beta}^{\beta}\}$
$L_{k,l}^{m,n} = \{B_{k,2^j l}^{2^j m, 2^j m}; 2^j l + 2^j m \le C, j > 1\}$ horizontal SOBT $L_{k,l}^{m,n} = \{B_{2^j k,l}^{2^j m, 2^j m}; 2^j k + 2^j m \le R, j > 1\}$ vertical SOBT $L_{k,l}^{m,n} = \{B_{2^j k,2^j l}^{2^j m, 2^j m}; 2^j k + 2^j m \le R \,\&\, 2^j l + 2^j m \le C, j > 1\}$ diagonal SOBT	$L_k^{\beta} = \{B_{4^j k}^{4^j \beta}; 4^j k + 4^j \beta \le 2^p\}$, whereby *p* is an integer such that $R = C = 2^p$

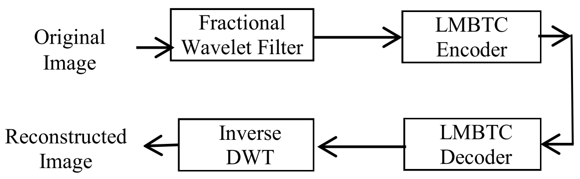


Fig. 1. Block diagram of proposed image codec for WSNs: The Fractional Wavelet Filter (FrWF) computes the wavelet transform coefficients, which are encoded with the novel Low Memory Block Tree Coding (LMBTC) algorithm.

i.e., progressive refinement of the image quality. For very low bit rates, the rate distortion performance of WBTC is superior to that of SPIHT [25]. The use of linked lists in WBTC requires large run time memory (dynamic memory) and necessitates memory management. Multiple memory accesses and complex memory management limit the application of WBTC in resource-constrained environments, such as handheld multimedia devices and wireless sensor networks.

III. PROPOSED FrWF-BASED LMBTC IMAGE CODEC

A. Overview

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 compression technique that gives good quality for high compression ratios and requires low memory is required for resource-constrained WSNs. The compression can also save energy within the network, as the coding energy is typically lower than the transmission energy [51]. We combine FrWF, a recently developed technique requiring only low memory to compute the discrete wavelet transform of images (as reviewed in Section II.A.), with the novel Low Memory Block Tree Coding (LMBTC) algorithm to develop a new image coder. The block diagram of the proposed codec is shown in Fig. 1. The input image is first transformed using FrWF and then quantized and entropy coded by the LMBTC algorithm. The bit-stream thus generated is embedded in nature. This bit-stream is decoded by the decoder and finally the image is reconstructed using the traditional inverse DWT.

B. Proposed Low Memory Block Tree Coding (LMBTC)

1) Data Structures: LMBTC is a listless form of the WBTC algorithm that replaces data-dependent lists with a small-size static memory. The transformed image, which is generally stored in raster fashion, is then converted into a linear index. An important property of the linear indexing is that it efficiently supports the operations on coefficient positions, needed for tree-based and block-based algorithms with one operation instead of two [38]. Table I defines the symbols and set structures used in WBTC (2D indexing) and LMBTC (which uses linear indexing). It is evident that a block of size $m \times n$ has mn consecutive indices while the coordinates of a set at a higher resolution level in a block tree can be obtained by multiplication by 4. The linear vector data structure and its linear indexing facilitates addressing of a set/block and block tree while facilitating breadth search in hierarchical trees.

2) Significance Testing: The significance function of a block set B against a threshold $T = 2^t$ is given as [57]:

$$S_n(B) = \begin{cases} 1 & if \quad T \le \max_{i \in B}(|c_i|) < 2T \\ 0 & if \quad \max_{i \in B}(|c_i|) < T \\ NULL & if \quad \max_{i \in B}(|c_i|) \ge 2T, \end{cases} \tag{3}$$

where NULL represents no output.

The significance of a block tree set BT (Type A or B) is given as

$$S_n(BT) = \begin{cases} 1 & \text{if} \quad T \le \max_{i \in BT}(|c_i|) < 2T \\ 0 & \text{if} \quad \max_{i \in BT}(|c_i|) < T. \end{cases} \tag{4}$$

The use of the LSP is avoided in LMBTC by merging the refinement pass with the sorting pass through modifying the significance function of WBTC as given in (3). A coefficient/block set is significant if it includes either a newly significant pixel, or a pixel requiring refinement, or both. Thus, LMBTC generates a bit stream of the same size as WBTC, but in a different order. The different order may result in slight degradation of the decoded image quality. The slight degradation occurs if the bit budget is exhausted in the middle of a bit plane, as the bit budget will also be used in refinements thereby reducing the newly identified significant pixels. However, at the end of a sorting pass, LMBTC encodes the same information as WBTC.

Consider an image X of size (R, C) with the dc removed and the image having been wavelet transformed using FrWF. The transformed image is read into the linear array $\Im$ of $N_{pix} = RC$ coefficients using linear indexing. The linear array $\Im$ exhibits a hierarchical pyramidal structure defined by the levels of decomposition, with the leftmost position in the array being the root.

3) Encoding Sequence: The proposed algorithm follows the set structures and partitioning rules of WBTC. The pseudo code of the LMBTC encoder is given in TABLE II. The encoder algorithm is performed for each bit plane n starting from the most significant bit plane with threshold $T = 2^t$ and decrementing down to 0 or until a prescribed bit budget is achieved. The algorithm begins by consecutively coding blocks in the LL band. Each block set is tested against the threshold $T = 2^t$ and its significance is encoded using (4). A newly significant block set or a block set containing a refinement is recursively partitioned into four adjacent blocks using quad partitioning until partitioning is no longer needed or a block of size 2×2 is attained. For a significant 2×2 block, significance of each of the four coefficients is encoded. For a newly significant coefficient, its sign bit is coded, and if the coefficient is found significant in previous passes, it needs refinement and its n^{th} bit is transmitted. To encode the remaining sub-band coefficients, they are linked through spatial orientation block -trees with their node block in the LL sub-band. In order to define a uniform child-parent relationship, it is assumed that the first quarter of the LL sub-band coefficients have no descendants and the remaining three-quarter of the coefficients have their children in sub-bands of corresponding orientation.

A static memory referred to as Static memory for Insignificant Block Tree (SIBT) of size $\{1 \times \frac{N}{4\beta}\}$ keeps track of block tree partitioning in the pass. The static memory SIBT maps all possible block tree nodes of the transformed image and records the states of the block tree nodes. The SIBT is initialized at the beginning of the encoding with the states of the tree nodes in the LL band as state '1', and the remaining nodes are set to state '0'. A state '1' represents a type 'A' bock tree while block-trees that are not considered for the significance test are represented by state '0'. The algorithm proceeds by checking the state of all tree nodes in the SIBT using the significance function given by (4) and their significance is encoded. If the type 'A' block tree is found significant, its descendant blocks (offspring blocks) are immediately encoded as explained earlier.

Next, the significance of the grand-descendant set is encoded and if it is found significant, four new type 'A' block-trees are formed. In particular, offspring blocks are formed as new nodes by changing the state of the offspring nodes from '0' to '1' in the SIBT and the state of the parent node is marked as '3'. For a significant type 'A' block tree with an insignificant set of grand decedents, the parent node is marked as type 'B' tree by changing the state of the node from '1' to '2'. The new type 'A' block-trees are tested for significance in the same sorting pass, while new type 'B' sets will be tested for significance in the next pass.

For a type 'B' bock-tree node, at first, descendent blocks (offspring blocks) are processed. Then, the significance of the set of grand-descendants is encoded and, if found significant, four new type 'A' block-trees are formed with offspring blocks as new nodes by changing the states of the offspring nodes from '0' to '1' in the SIBT.

If the state of a block node is '3' (i.e., the block tree has already been found significant in previous passes), the offspring blocks are processed as explained earlier.

TABLE II
PSEUDO CODE OF PROPOSED LMBTC ENCODING ALGORITHM

```
(1) Initialization:
Set 't' = $\lfloor \log_2(\max\{|\Im|\}) \rfloor$ as initial threshold; β is  block size; Initialize
  SIBT of size (1 x $N_{pix}$ /4β) with block tree nodes in LL-band set to state
  '1' and set remaining elements to '0'
(2) Sorting pass
/* LL-band coding */
For each block in LL-band do: call Process ς ( )
/* Block tree coding */
For each location j in SIBT do
  ❖ If SIBT(j)=1; (Type 'A' node) then
      ➢ Find node index i=jβ;
          Output $S_n(D_i^\beta)$
          ▪ If $S_n(D_i^\beta)$=1
              For each offspring block ∈ $O_i^\beta$ do: call Process ς ( )
          • If $L_i^\beta \neq$ (NULL) then
              Output $S_n(L_i^\beta)$
              ♦ If $S_n(L_i^\beta)$=1
                  Set SIBT (4j:4j+3) to '1'and Set SIBT (j) =3
              ♦ Else Set SIBT (j) =2;
          • Else  Set SIBT (j) =3;
  ❖ If SIBT(j)=2; (Type 'B' node) then
      ➢ Find node index i=jβ;
          For each offspring block ∈ $O_i^\beta$ do: call Process ς ( )
          Output Sn($L_i^\beta$)
          ▪ If Sn($L_i^\beta$)=1
              Set SIBT (4j:4j+3) as '1'and Set SIBT (j) =3
  ❖ If SIBT(j)=3; (Type 'B' node) then
      ➢ Find node index i=jβ;
          For each offspring block ∈ $O_i^\beta$  do: call Process ς ( )
(3) Quantization step update
    Decrement n by 1 and go to step 2

Process ς ( )  /* Function for processing of block*/
{
For a block $B_i^\beta$  do;
  ❖ Output $S_n(B_i^\beta)$
      ➢ If $S_n(B_i^\beta) \neq 0$
          ▪ If β=1; a coefficient
              • If $S_n(B_i^\beta)$ =1
                  Output sign bit
              • Else  Output nth bit
          ▪ Else For each block ∈ $Q_i^\beta$  do: call Process ς ( )
}
```

TABLE III
COMPARISION OF MEMORY REQUIRED BY FRACTIONAL WAVELET FILTER (FrWF) AND CONVENTIONAL DWT AS WELL AS PROPOSED LMBTC FOR DIFFERENT TRANSFORM LEVELS AND BLOCK SIZES

Image Size [pixels]	Wavelet Transform Decomposition Level	Required Memory (kB)				
		Transform stage only		Proposed LMBTC Coder with Block Size (β)		
		FrWF	DWT	β = 4	β = 16	β = 64
256×256	3	4.864	688.128	1.024	0.256	0.064
	5	6.246	698.368	1.024	0.256	0.064
512×512	3	9.728	2752.512	4.096	1.024	0.256
	5	12.493	2793.472	4.096	1.024	0.256

4) Decoding Sequence: The decoder follows the same overall procedure as the encoder with some low-level changes and an additional step of significance testing of coefficient/block sets to identify coefficient/block sets containing coefficients requiring refinement. To decode, (use input) instead of output, and set the (bits) and signs of coefficients, the decoder performs mid-tread de-quantization for coefficients that are not fully decoded [57].

5) Memory Requirements: The LMBTC algorithm uses a static memory SIBT to store the state of a node. Only four markers are used to define a node state thereby requiring two bit per node. For an image size of $R \times C$ with L transform levels, the LMBTC algorithm requires a memory size [57] of:

$$Memory_{LMBTC} = \frac{2RC}{4\beta} bits = \frac{RC}{16\beta} bytes. \quad (5)$$

For example, for an image size of 512 × 512 pixels with 5 levels of transform, the static memory sizes required by the LMBTC algorithm, for block sizes $\beta = 4$, $\beta = 16$, and $\beta = 64$ are 4.096 kB, 1.024 kB, and 0.256 kB, respectively.

IV. SIMULATION RESULTS

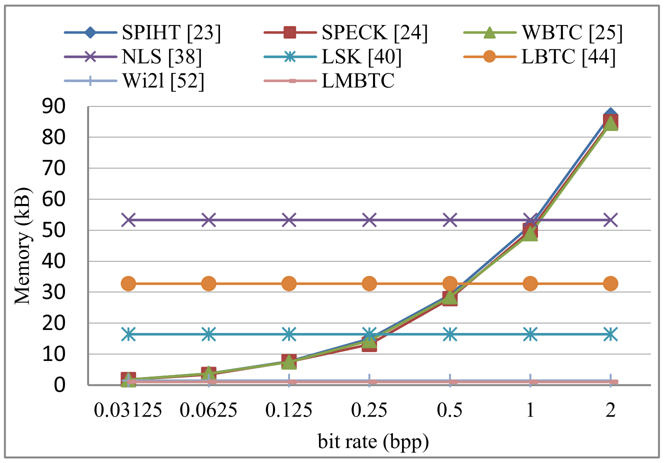
This section compares the memory requirement, coding efficiency, and computational complexity of the LMBTC-based image codec with other codecs, including SPIHT, SPECK, NLS, LSK, and WBTC, for nine different test images. The test images are from the Waterloo Repertoire (http://links.uwaterloo.ca) and the standard image database (http://sipi.usc.edu/database). Among these nine images, five are of dimension 512 × 512 (each 8 bits/pixel) and the remaining four are of dimension 256 × 256 (each 8 bits/pixel). The five images of dimension 512 × 512 are *Lena*, *Barbara*, *Baboon*, *Goldhill*, and *Boat*. The remaining four images of size 256 × 256 are *Barbara*, *Goldhill*, *Bridge*, and *Baboon*. Unless otherwise noted, all reported results are averages of the considered images for a given image dimension. Up to five levels of wavelet decomposition are performed using the 9/7 Daubechies filter by both the fractional wavelet filter (FrWF) scheme and the traditional DWT scheme. Floating point transform coefficients are quantized to the nearest integers, and read into the linear array using linear indexing. The coefficients in the linear array are then encoded using the LMBTC, SPIHT, SPECK, NLS, LSK, and WBTC algorithms.

The image is decoded by the decoder and reconstructed by the inverse traditional DWT. The decoding of the bit-stream is typically done outside the sensor network, at workstations that are not constrained in memory and processing power. Thus, the reconstruction is done using the traditional inverse DWT. For a fair comparison all the codecs (LMBTC, SPIHT, SPECK, NLS, LSK, and WBTC) are implemented using MATLAB 2011 and are executed on a Pentium I3 computer system equipped with a 2.4 GHz processor and 4 GB RAM. The results are presented in the following three subsections that focus on the memory analysis, coding efficiency, and complexity analysis.

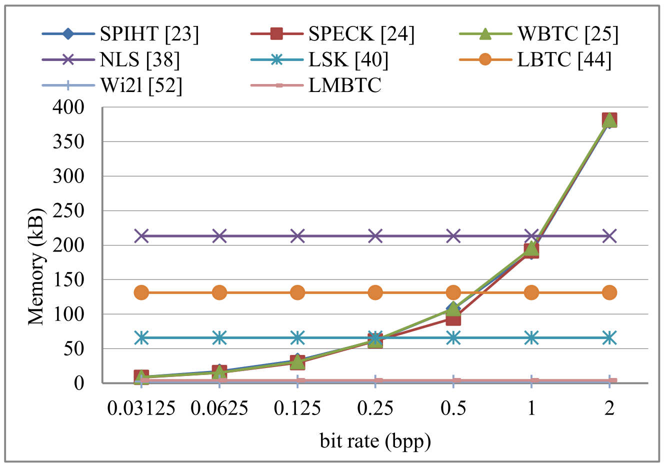
A. Memory Analysis

Limited memory is one of the main constraints, when image coding algorithms are implemented on wireless sensor nodes. Many of the low-cost sensor nodes have on-board memory (RAM) of the order of 10 kB [16]. Table III compares the working memory required by FrWF and conventional DWT (for different transform levels) as well as by the LMBTC algorithm (for different block sizes). It can be observed from the table that FrWF requires much less memory than the conventional DWT. Specifically, the memory requirement of FrWF is in the range of 5-12.5 kB (depending on image size and number of decomposition levels), whereas the conventional DWT needs memory of the order of 0.7-2.8 MB. This is because the FrWF stores only three image lines in memory, whereas the conventional DWT stores the entire image in memory. Further, Table III indicates that the memory requirement of the LMBTC algorithm is a function of the image size and the block size, as indicated by Eqn. (5). It is evident that increasing the block size reduces the memory requirement of LMBTC quite significantly. This is due to the fact that a larger block size in the LMBTC algorithm reduces the number of elements in the SIBT. From Table III we also observe the increase of the memory requirement with increasing image sizes, which require more coefficients to be stored and processed. The table also illustrates that the memory requirement of LMBTC is independent of the transform level. In nutshell, the LMBTC algorithm needs on-board RAM of the order of hundreds of bytes to a few kilo bytes, depending on the image size and block size. Overall the memory requirement of an image coder is the maximum of the two memory requirements, i.e., the memory requirements of the wavelet transform and the encoding/quantization stages. Most of the memory is consumed by the wavelet transform stage, which therefore governs the overall memory requirement.

The results in Table III, indicate that the FrWF-based LMBTC with 5 levels of wavelet decomposition requires less than 10 kB of memory for images of size 256 × 256 pixels, for block sizes ranging from 4 to 64. Moreover, an image



(a)



(b)

Fig. 2. Comparison of memory requirements of image coding algorithms as a function of bit rate [in bit per pixel, bpp] for (a) 256 × 256 image, (b) 512 × 512 image.

TABLE IV

EFFECT OF BLOCK SIZE IN LMBTC CODER IN TERMS OF PSNR AS WELL AS ENCODING AND DECODING TIMES FOR BARBARA (512 × 512) IMAGE; THE CODER MEMORY REQUIREMENTS ARE GIVEN IN TABLE III

bpp	PSNR (dB)			Encoding time (s)			Decoding time (s)		
	β=4	β =16	β =64	β=4	β =16	β =64	β=4	β =16	β =64
0.03125	30.23	30.26	30.27	0.248	0.164	0.155	0.129	0.087	0.0714
0.0625	30.75	30.75	30.75	0.401	0.284	0.256	0.237	0.190	0.152
0.125	31.18	31.19	31.19	0.688	0.453	0.367	0.438	0.339	0.281
0.25	32.03	32.03	32.03	1.283	0.807	0.606	0.841	0.633	0.513
0.5	33.47	33.46	33.44	2.371	1.545	1.067	1.672	1.317	1.107

of size 512×512 can be wavelet transformed with 3 decomposition levels with less than 10 kB of memory and then requires only about 4 kB of memory for the LMBTC encoding of the wavelet transform coefficients.

Figs. 2(a) and (b) compare the memory requirement of the LMBTC algorithm (with block size of 4) with other state-of-art wavelet-based coding algorithms, such as SPIHT [23], SPECK [24], WBTC [25], NLS [38], LSK [40], Wi2l [51], and LBTC [43], for image sizes 256 × 256 and 512 × 512, respectively. All images are transformed using FrWF with 5 levels of decomposition. The figures show the average memory (averaged over all test images of corresponding size). Fig. 2(a) indicates that for an image size of 256 × 256, the LMBTC algorithm requires the least memory among the considered coders. Although the memory required by the Wi2l image coding algorithm is quite low, the Wi2l coder generates a non-embedded bit-stream, making it unsuitable for heterogeneous networks. For a 512 × 512 image, LMBTC requires 4.096 kB, 1.024 kB, and 0.256 kB for block sizes of 4, 16, and 64, respectively, whereas the Wi2l coder requires 2.046 kB of memory. Therefore, the LMBTC coder with large block size has lower memory requirements than the Wi2l coder.

Further, it may be noted that the lower memory requirement of LMBTC with large block size is achieved at the cost of insignificant quality reduction of the decoded image (as evident from Table IV). The increasing block size in LMBTC increases the location/significance bits of the significant blocks and block trees. Therefore, bits saving due to larger insignificant blocks and block trees are compensated by the extra bits required in searching for the significant coefficients.

From both Figs. 2(a) and (b), we also observe that the memory required by the SPIHT, SPECK, and WBTC coding algorithms increases as the bit rate increases, whereas the memory requirement is independent of the bit rate for the other algorithms, i.e., the listless coding algorithms.

B. Coding Efficiency (Rate-Distortion Performance)

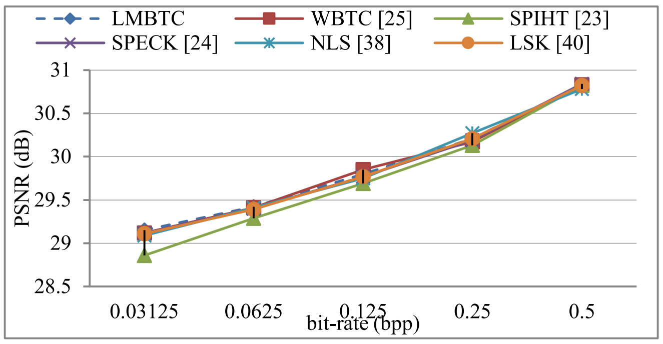
Coding efficiency is the measure of rate-distortion (RD) performance of an image coder. Coding efficiency is commonly measured in terms of bits/pixel to achieve a minimum desired quality of the image. The quality of the reconstructed image is measured in terms of Peak-Signal-to-Noise-Ratio (PSNR), defined [16] as:

$$PSNR = 10\log_{10}\frac{255^2}{\text{MSE}}, \tag{6}$$

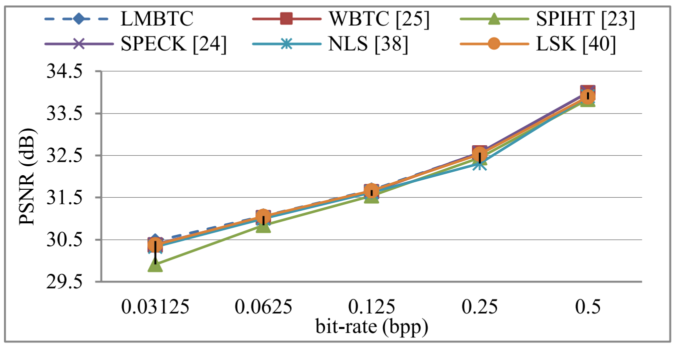
where MSE is mean square error of the reconstructed image with the original image

$$MSE = \frac{1}{N}\sum\nolimits_{x,y}\left[f(x,y) - g(x,y)\right]^2. \tag{7}$$

Here, N denotes the total number of pixels in each image, $f(x, y)$ is the original image, and $g(x, y)$ is the reconstructed image. For perfect reconstruction, the PSNR value is infinity. However image degradations with a PSNR of 40 dB or higher are nearly invisible by human observers [16]. If the PSNR is in the range of 25-30 dB, then the image quality



(a)



(b)

Fig. 3. PSNR image quality for different coding algorithms combined with FrWF as a function of bit rate for (a) 256×256 image, (b) 512 × 512 image.

is typically acceptable. In this section the coding efficiency of the LMBTC based codec is evaluated for all test images. The test images are transformed using FrWF and then coded using the WBTC, SPIHT, SPECK, NLS, LSK, and LMBTC coding algorithms at a range of bit-rates. The average PSNR values (averaged over all test images of the same size) of the reconstructed images at various bit-rates for 256 × 256 and 512 × 512 images are given in Figs. 3(a) and (b), respectively.

From Figs. 3(a) and (b), we observe that up to 0.125 bpp, the LMBTC codec gives very slightly higher PSNR values than the other codecs. However, at bit-rates higher than 0.125 bpp, the performance of the LMBTC codec is slightly inferior compared to the other codecs. The reason for this performance characteristic is that in the LMBTC coding algorithm, the sorting and refinement passes are merged, and if the bit budget is exhausted before the end of a bit-plane, then fewer significant coefficients are encoded compared to the corresponding coding algorithm with separate sorting and refinement passes (i.e., WBTC). It may be noted that a bit representing a new significant pixel provides more improvement in PSNR than a bit representing a refinement pixel. However, it is expected that at the end of a bit-plane, the coding efficiency of LMBTC coding algorithm is equivalent to that of WBTC. Since WSNs require generally a high compression (small bpp) [16], the FrWF-based LMBTC codec appears highly suitable for WSNs.

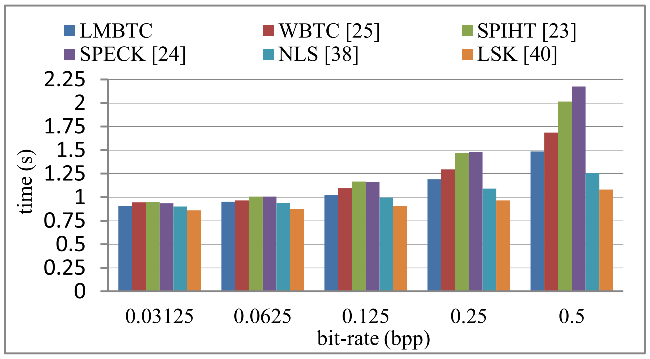
C. Complexity Analysis

The computational complexity of the proposed codec is evaluated by estimating the computation time required for encoding the transformed coefficients (with FrWF and traditional DWT) and for decoding the generated bit-stream at each bit-rate. The complexity of the LMBTC codec is compared with other state-of-art codecs (each with FrWF and DWT). The encoding and decoding times are measured on a Pentium I3 Computer system with 2.4 GHz processor and 4 GB RAM. The encoding time and decoding time (combined with FrWF) for image size of 256 × 256 are given in Figs. 4(a) and (b), respectively. The encoding time is the total time required for calculating the transform as well as the time required for encoding the transform coefficients. The decoding time is the time required for decoding the bit-stream as well as the time required for reconstructing the image.

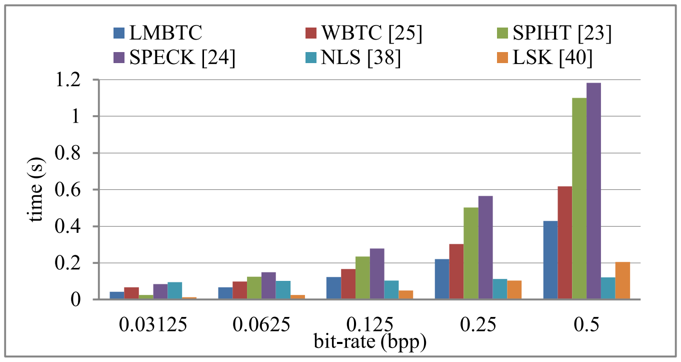
From these figures we observe that for each codec, the encoding time increases as the bit-rate increases. This is expected because more coefficients need to be encoded and hence the encoding time increases as the bit-rate increases. This is also evident from Table IV. Further, at each bit-rate, the encoding time of the listless coding algorithms, namely LMBTC, NLS, and LSK, is lower than their corresponding list-based algorithms, such as WBTC, SPIHT, and SPECK. The list-based algorithms use linked-lists to keep track of the order of the transform coefficients that have been coded or are yet to be coded. The processing and management of these lists lead to higher computational demands and thus longer encoding times compared to their listless versions. Especially at high bit-rates, the SPECK and SPIHT algorithms have long encoding times because of the processing and managing of long lists. The multiple accesses of linked lists further add to the computational complexity of these codecs [25].

The listless codecs (NLS, LSK, and LMBTC) use fixed-size state tables instead of lists. The state tables do not require any memory or processing management. Therefore, these listless codecs require less time for encoding the transform coefficients compared to the list-based SPIHT, SPECK, and WBTC codecs. Among the listless codecs, NLS uses 12 markers, each of 4 bits, whereas LSK uses only 4 markers, each of 2 bits [43]. Hence, the encoding time of LSK is shorter than that for NLS. LMBTC has an additional step of significance testing of blocks, therefore the encoding time of LMBTC is slightly higher than for LSK and NLS.

The encoding times when the algorithms employ FrWF are slightly longer than for DWT. This is because the time required by FrWF to compute the transform is longer than that required by DWT. From Fig. 4(b) we observe that the decoding time increases with increasing bit-rate. Fig. 4(b) also indicates that the decoding times of the WBTC, SPIHT, and SPECK algorithms are longer compared to the listless NLS, LSK, LMBTC coding algorithms. This is because list-based codecs need to perform multiple memory accesses due to their bit-plane coding nature [14]. Comparing Fig. 4(a) with Fig. 4(b), we observe that the encoding time is longer than the decoding time because encoders compare coefficients or blocks against a threshold to check their significance, but no comparisons are required at the decoders [14]. In additional evaluations that are not included due to space



(a)



(b)

Fig. 4. (a) Comparison of encoding time for image size (256 × 256) for different image coding algorithms combined with FrWF. (b) Comparison of decoding time for image size (256 × 256) for different image coding algorithms.

TABLE V

COMPARISON OF PSNR, MEMORY, ENCODING TIME, AND DECODING TIME FOR DIFFERENT CODECS AT 0.25 bpp

Image Size	Performance Metrics	FrWF	DWT	SPIHT [23]	SPECK [24]	WBTC [25]	NLS [38]	LSK [40]	Proposed LMBTC (β=4)
256×256	PSNR (dB) (with FrWF)	-	-	30.13	30.19	30.17	30.27	30.21	30.21
	PSNR (dB) (with DWT)	-	-	30.15	30.31	30.31	30.29	30.33	30.37
	Memory (kB)	6.246	698.368	14.931	14.142	14.348	53.248	16.384	1.024
	Encoding time (s)	0.842	0.369	0.624	0.645	0.455	0.25	0.125	0.359
	Decoding time (s)	-	0.019	0.503	0.566	0.302	0.121	0.104	0.224
512×512	PSNR (dB) (with FrWF)	-	-	32.45	32.56	32.56	31.91	32.53	32.56
	PSNR (dB) (with DWT)	-	-	32.61	32.78	33.06	32.73	32.76	32.83
	Memory (kB)	12.493	2793.472	62.567	60.927	61.857	212.937	65.537	4.096
	Encoding time (s)	1.86	0.843	3.081	2.898	1.901	0.922	0.481	1.346
	Decoding time (s)	-	0.069	1.451	2.597	1.394	0.513	0.385	0.866

constraints, we found that the images of size 512 × 512 show similar trends to those observed for 265×256 size images in Figs. 4(a) and (b). However, the absolute encoding and decoding times for each codec are higher for the large-sized images.

Finally, Table V summarizes and compares the coding efficiency and memory, as well as encoding and decoding times of all considered image codecs with and without FrWF at 0.25 bpp, for image sizes of 256 × 256 and 512 × 512 pixels. In these tables, the block size is set to 4 in the WBTC and LMBTC algorithms. These tables indicate that the PSNR image qualities achieved by LMBTC are comparable with other image codecs; whereas, the memory requirement of LMBTC is significantly smaller than the other image codecs. The complexity (encoding and decoding time) of the LMBTC codec is lower than that of the SPIHT, SPECK, and WBTC codecs, while it is higher than that of the NLS and LSK codecs.

V. CONCLUSION

In order to satisfy the memory constraints of low-cost visual sensor nodes, a low-memory image codec is required. In this study the memory requirement at the transform stage is reduced by using the Fractional Wavelet Filter (FrWF) [56] and the memory for encoding the transform coefficients is reduced by using a novel Low Memory Block Tree Coding (LMBTC) algorithm. From the simulation results, it is observed that the FrWF requires only 6.246 kB memory for five levels of wavelet decomposition of a 256 × 256 image; the memory required by FrWF increases with the image size and the number of wavelet transform decomposition levels. The transform coefficients are encoded using the LMBTC algorithm. LMBTC requires only 0.064 kB memory for a 256 × 256 image and 0.256 kB for a 512 × 512 image, when the block size is 64. The overall memory, which is the maximum of the memory requirements of FrWF and LMBTC (with block size 4), is 6.246 kB and 12.493 kB for 256 × 256 and 512 × 512 size images, respectively. The memory required by the proposed LMBTC coder is independent of the bit-rate. Compared with other state-of-art coders, such as SPIHT, SPECK, WBTC, NLS, and LSK, the proposed LMBTC coder has significantly reduced memory requirements, and can be implemented on low-memory sensor nodes. The memory required by the Wi2l coder [51] is of the same order, but the Wi2l coder generates a non-embedded bit-stream. In contrast, the LMBTC coder generates a fully embedded bit-stream, which is highly desirable for flexible image encoding and network transmission. Overall, the proposed FrWF-based LMTBC image coder appears very well suited for wireless sensor networks. Future research directions include the examination of the proposed efficient image coding within the specific constraints of embedded sensor platforms [5], [6] as well as innovate multimedia network transport paradigms, e.g., information-centric and name-based networking [58], [59].

REFERENCES

[1] K. Sohraby, D. Mioli, and T. Znati, *Wireless Sensor Networks: Technology, Protocols, and Applications*, 1st ed. Hoboken, NJ, USA: Wiley, 2007.

[2] 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.

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

[4] B. Song, C. Ding, A. T. Kamal, J. A. Farrell, and A. K. Roy-Chowdhury, "Distributed camera networks," *IEEE Signal Process. Mag.*, vol. 28, no. 3, pp. 20–31, May 2011.

[5] 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.

[6] 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.

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

[8] P. N. Huu, V. Tran, and T. Miyoshi, "Low-complexity and energy-efficient algorithms on image compression for wireless sensor networks," *IEICE Trans. Commun.*, vol. 93, no. 12, pp. 3438–3447, Dec. 2010.

[9] K. Khursheed, M. Imran, N. Ahmad, and M. O'Nils, "Selection of bi-level image compression method for reduction of communication energy in wireless visual sensor networks," *Proc. SPIE*, vol. 8437, p. 84370M, Jun. 2012.

[10] 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.

[11] A. Mammeri, B. Hadjou, and A. Khoumsi, "A survey of image compression algorithms for visual sensor networks," *ISRN Sensor Netw.*, vol. 2012, pp. 760320-1–760320-19, Oct. 2012.

[12] M. Nasri, A. Helali, H. Sghaier, and H. Maaref, "Images compression techniques for wireless sensor network applications," *Int. J. Speech Technol.*, vol. 18, no. 2, pp. 205–216, Nov. 2014.

[13] M. Rehman, M. Sharif, and M. Raza, "Image compression: A survey," *Res. J. Appl. Sci., Eng. Technol.*, vol. 7, no. 4, pp. 656–672, 2014.

[14] K. K. Hasan, U. K. Ngah, and M. F. M. Salleh, "Efficient hardware-based image compression schemes for wireless sensor networks: A survey," *Wireless Pers. Commun.*, vol. 77, no. 2, pp. 1415–1436, 2014.

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

[16] 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.

[17] 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.

[18] B.-S. Chow, "Mathematical morphology for applications to sensor networks," *IEEE Sensors J.*, vol. 12, no. 12, pp. 3473–3479, Dec. 2012.

[19] 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.

[20] T. Dutta, "Medical data compression and transmission in wireless ad hoc networks," *IEEE Sensors J.*, vol. 15, no. 2, pp. 778–786, Feb. 2015.

[21] 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.

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

[23] 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.

[24] 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.

[25] S.-M. Guo, Y.-W. Lee, C.-Y. Hsu, and S.-J. Guo, "Performance improvement of set partitioning embedded block algorithm for still image compression," in *Modern Advances in Applied Intelligence*, vol. 8482. New York, NY, USA: Springer-Verlag, 2014, pp. 270–278.

[26] 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.

[27] 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. (ICASSP)*, vol. 4. Jun. 2000, pp. 2035–2038.

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

[29] 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.

[30] 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.

[31] J. Oliver and M. 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.

[32] 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.

[33] Y. Bao and C.-C. J. Kuo, "Design of wavelet-based image coder in memory-constrained environment," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, no. 5, pp. 642–650, May 2001.

[34] S. Rein, S. Lehmann, and C. Gühmann, "Fractional wavelet filter for camera sensor node with external flash and extremely little RAM," in *Proc. 4th ACM Mobile Multimedia Commun. Conf. (MobiMedia)*, Jul. 2008, pp. 1–7.

[35] H. Arora, P. Singh, E. Khan, and F. Ghani, "Memory efficient image coding with embedded zero block-tree coder," in *Proc. IEEE Int. Conf. Multimedia Expo (ICME)*, vol. 1. Jun. 2004, pp. 679–682.

[36] W.-K. Lin and N. Burgess, "Listless zerotree coding for color images," in *Proc. IEEE Signals, Syst. Comput.*, vol. 1. Nov. 1998, pp. 231–235.

[37] F. W. Wheeler and W. A. Pearlman, "Low-memory packetized SPIHT image compression," in *Proc. IEEE Asilomar Conf. Signals, Syst. Comput.*, vol. 2. Oct. 1999, pp. 1193–1197.

[38] F. W. Wheeler and W. A. Pearlman, "SPIHT image compression without lists," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, vol. 4. Jun. 2000, pp. 2047–2050.

[39] 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.

[40] 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.

[41] 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.

[42] R. K. Senapati, U. C. Pati, and K. K. Mahapatra, "Reduced memory, low complexity embedded image compression algorithm using hierarchical listless discrete Tchebichef transform," *IET Image Process.*, vol. 8, no. 4, pp. 213–238, Apr. 2014.

[43] R. K. Senapati, U. C. Pati, and K. K. Mahapatra, "Listless block-tree set partitioning algorithm for very low bit rate embedded image compression," *Int. J. Electron. Commun.*, vol. 66, no. 12, pp. 985–995, Dec. 2012.

[44] R. K. Senapati and P. Mankar, "Improved listless embedded block partitioning algorithms for image compression," *Int. J. Image Graph.*, vol. 14, no. 4, p. 1450020, Oct. 2014.

[45] J. Guo, S. Mitra, B. Nutter, and T. Karp, "A fast and low complexity image codec based on backward coding of wavelet trees," in *Proc. Data Compress. Conf. (DCC)*, Mar. 2006, pp. 292–301.

[46] L. Ye, J. Guo, B. Nutter, and S. Mitra, "Memory-efficient image codec using line-based backward coding of wavelet trees," in *Proc. Data Compress. Conf. (DCC)*, Mar. 2007, pp. 213–222.

[47] L. Ye, J. Guo, B. S. Nutter, and S. D. Mitra, "Low-memory-usage image coding with line-based wavelet transform," *Opt. Eng.*, vol. 50, no. 2, pp. 027005-1–027005-11, Feb. 2011.

[48] 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.

[49] 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.

[50] 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.

[51] S. Rein, S. Lehmann, and C. Gühmann, "Wavelet image two-line coder for wireless sensor node with extremely little RAM," in *Proc. Data Compress. Conf. (DCC)*, Mar. 2009, pp. 252–261.

[52] J. Guo, S. Mitra, B. Nutter, and T. Karp, "Backward coding of wavelet trees with fine-grained bitrate control," *J. Comput.*, vol. 1, no. 4, pp. 1–7, Jul. 2006.

[53] O. López, M. Martínez-Rach, J. Oliver, and M. P. Malumbres, "Impact of rate control tools on very fast non-embedded wavelet image encoders," *Proc. SPIE*, vol. 6508, p. 650829, Jan. 2007.

[54] O. M. L. Granado, M. O. Martínez-Rach, P. P. Peral, J. O. Gil, and M. P. Malumbres, "Rate control algorithms for non-embedded wavelet-based image coding," *J. Signal Process. Syst.*, vol. 68, no. 2, pp. 203–216, 2012.

[55] J. Guo, S. Mitra, T. Karp, and B. Nutter, "A resolution- and rate- scalable image subband coding scheme with backward coding of wavelet trees," in *Proc. IEEE Asia Pacific Conf. Circuits Syst. (APCCAS)*, Dec. 2006, pp. 442–445.

[56] 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.

[57] N. R. Kidwai, "Efficient image coding for wireless sensor networks," Ph.D. dissertation, Dept. Electron. Commun. Eng., Integral Univ., Lucknow, India, Apr. 2014.

[58] A. Seema, L. Schwoebel, T. Shah, J. Morgan, and M. Reisslein, "WVSNP-DASH: Name-based segmented video streaming," *IEEE Trans. Broadcast.*, vol. PP, no. 99, 2015, doi: 10.1109/TBC.2015.2400816.

[59] A. Detti, B. Ricci, and N. Blefari-Melazzi, "Mobile peer-to-peer video streaming over information-centric networks," *Comput. Netw.*, vol. 81, pp. 272–288, Apr. 2015.

Mohd Tausif received the B.Tech. degree in electronics engineering from Gautam Buddha Technical University, Lucknow, India, in 2012, and the M.Tech. degree in communication and information system from Aligarh Muslim University, Aligarh, India, in 2014. He is currently pursuing the Ph.D. degree from IIT Patna, India. His research interests include signal processing and low-complexity image coding algorithms.

Naimur Rahman Kidwai received the B.Sc. (Eng.) degree in electronics engineering from the Z. H. College of Engineering and Technology, AMU Aligarh, 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. He has 19 years of experience, including 15 years of teaching experience. His areas of expertise are signal processing, digital communication, and image processing.

Ekram Khan (M'07–SM'12) received the B.Sc.(Eng.) and M.Sc.(Eng.) 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. He has authored or co-authored over 90 papers in refereed academic journals and international conference proceedings. 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, India. He received a Commonwealth Scholarship to pursue the Ph.D. degree from the University of Essex. He has received a Research Award from the IBM J. T. Watson Research Laboratory, USA, for the best disruptive idea in ICME 2002, held in EPFL, Switzerland. He has successfully completed the number of research projects funded by various funding agencies in India and U.K. He had spent summer of 2005 and 2006, as an academic visitor, with the University of Essex (funded by Royal Society, U.K).

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.