J. Vis. Commun. Image R. xxx (2016) xxx–xxx

Contents lists available at ScienceDirect

J. Vis. Commun. Image R.

journal homepage: www.elsevier.com/locate/jvci

Scalable line-based wavelet image coding in wireless sensor networks ☆

Stephan Rein, Martin Reisslein *

School of Electrical, Computer, and Energy Eng., Arizona State University, Tempe, AZ 85287-5706, USA

ARTICLE INFO

Article history:
Received 8 December 2015
Revised 11 July 2016
Accepted 12 July 2016
Available online xxxx

Keywords:
Low-memory image coding
Sensor node
Scalable image compression
Wavelet image coding

ABSTRACT

Existing scalable wavelet image coding approaches, such as set partitioning in hierarchical trees and its derivatives, employ a memory-intensive tree-based coding structure. Existing tree-based wavelet coding approaches are therefore not suitable for memory-constrained sensor nodes. In this paper, we introduce a scalable wavelet image coding approach based on a line structure that requires very little memory. The proposed line-based approach is suitable for scalable wavelet image coding in memory-constrained sensor nodes, requiring only a few kilobytes of memory for a 256×256 pixel image. The presented line-based wavelet coding algorithm accesses the image data line by line and thus conforms with the data access patterns in current flash memory technology. Our performance evaluations demonstrate that the proposed scalable line-based image wavelet coding approach has no overhead compared to one-run (non-scalable) wavelet image coding and has competitive compression performance compared to JPEG 2000 and the recent Google WebP image format.

1. Introduction

1.1. Motivation: scalable image coding and communication

Scalable image coding generates a base stream providing a low (base) image quality as well as scalable (refinement) streams that successively improve the image quality. Scalable image coding is a promising technique for image communication in bandwidth constrained networks, such as wireless sensor networks, since the base image quality can be displayed at the receiver after the first bytes of the base stream have been received. As scalable (refinement) streams are received, the image quality is improved.

Wavelet transform based image compression achieves superior image quality when using extremely high data compression [1,2]. High compression is especially useful for low-cost camera sensor networks with low-quality links. However, as reviewed in detail in Section 1.5, set partitioning in hierarchical trees (SPIHT) and similar tree-based wavelet image coding approaches have been designed for implementation on a personal computer (PC) with abundant random access memory (RAM). For a low-cost sensor node with a 16-bit micro controller with limited memory only very few approaches for scalable wavelet image coding have been investigated, see Section 1.5. Typically, such low-RAM sensor nodes employ an external flash memory for storing the image data [3,4]. This external flash memory has a line-based structure, i.e., read and write accesses on a line-by-line basis are significantly more efficient than random access patterns [5,6]. We have specifically designed our low-memory scalable image coding approach to take the line-based organization of external flash memory into consideration.

1.2. Application setting: wireless image sensor network

Node platforms and network structures for wireless camera sensor networks have been extensively researched in recent years [3,4,7–10]. Visual information processing constitutes often one of the most resource demanding tasks in visual sensor networks [11,12]. In particular, the image compression is a demanding task for sensor nodes [13–15]. The considered application setting for the proposed scalable image wavelet coding approach is a low-cost camera sensor in a natural environment. (The proposed approach has not been evaluated for artificial or medical images.) The sensor node is connected via a network to a sink node, e.g., a tablet. An image captured by the sensor camera is to be communicated to the tablet. The scalability feature shall improve the user experience.

The link between sensor node and tablet is typically limited. In particular, the sensor node uploads the data over a wireless sensor network which may be slow due to multiple wireless hops and simple wireless transceivers. The tablet downloads the data over an access network which can also be a bottleneck. The compression

☆ This paper has been recommended for acceptance by M.T. Sun.

* Corresponding author.
E-mail addresses: rein@gmx.net (S. Rein), reisslein@asu.edu (M. Reisslein).
URL: http://mre.faculty.asu.edu (M. Reisslein).

http://dx.doi.org/10.1016/j.jvcir.2016.07.006
1047-3203/

algorithm is part of the sensor node and the tablet software. In particular, the sensor node requires the low-memory feature of the proposed coding algorithm, while the receiving tablet may not require it (although for the receiver, the memory allocation is similarly low). The sensor node sends a base quality of the image to the tablet and the tablet displays the base quality image. Next, the sensor node sends sets of image refinement data to the tablet, which are utilized to update the previously displayed image, resulting in image quality improvements. The update process can be repeated multiple times.

Key parameters in this application scenario are the base amount of data resulting in the base image quality and the amounts of data for each update. The proposed algorithm allows for updates on the receiver at the granularity of one or multiple quantization levels, whereby one quantization level relates to the decoding of one additional bit in all pixels of an image, see Section 2. A finer granularity, e.g., bit-wise granularity for single pixels, would require significantly increased computation by the sender and receiver and is thus not addressed by the presented algorithm.

1.3. Contribution

This paper proposes a low-complexity algorithm for scalable wavelet image compression. Low-complexity refers to low RAM requirements (in the range of a few kilobytes), efficient utilization of line-based flash memory [16], and limited conceptual complexity, as required for a typical low-cost sensor node. More specifically, the proposed scalable wavelet image compression requires only RAM for two image lines, similar to the non-scalable two-line codec [17,18]. The proposed coding method can be implemented with a few pages of source code. We demonstrate that the technique is competitive to state-of-the-art wavelet image coding.

1.4. Article structure

The paper is structured as follows. Related work is reviewed in Section 1.5. In Section 2, we describe the underlying principles and the coding notation for this work. Section 3 describes the proposed algorithm for scalable image compression. In Section 4, we analyze the computational complexity of the proposed scalable line-based image compression approach. In Section 5, the scalability feature of the given system is verified and its rate-distortion (RD) compression performance is compared to JPEG 2000 and the Google WebP converter. Section 6 concludes the paper.

1.5. Related work

Several studies have focused on reducing the memory required for the compression (encoding) of the transform coefficients resulting from the wavelet transform of an image. One group of studies has focused on the essential lists maintained for the popular set partitioning in hierarchical trees (SPIHT) form of wavelet-based image compression, see e.g., [19–24]. An extensive set of studies has focused on reducing memory requirements by processing only prescribed portions of the image. Strip-based approaches are examined in [25–27], while block-based approaches are studied in [28–31]. These studies have substantially reduced the memory requirements to the order of roughly 10 kByte or more for common image formats. For instance, the approach in [27] requires eight image lines, i.e., $8 \times 256 \times 2$ Bytes $= 4096$ Bytes for a 256×256 image of 16 bit samples. Eight to twelve image lines of memory are needed for the approach in [32]. Similar line-based approaches have been explored in [33,34] and require at least twelve image lines of memory. An image coding approach based on generic hierarchical transform, requiring tens to hundreds of image lines, is developed in [35]. In contrast, to these existing low-memory wavelet image compression approaches, which require at least memory for eight (or more) lines of the image, we propose and evaluate a novel scalable coding approach that reduces the memory requirements down to two image lines.

In order to achieve this significant reduction in memory requirements we build on the backward coding wavelet tree approach [36–39]. This backward encoding approach essentially conducts the SPIHT processing backwards, starting from the lowest wavelet transform level. The original backward encoding approach requires approximately 20 kByte of RAM memory for a 256×256 image. By breaking coding units in the backward coding approach into smaller pieces and interleaving coding tasks, the two-line codec [17,18] reduces the memory requirements for compressing a 256×256 image to approximately 1.5 kByte. Similar reductions in memory requirements have been achieved by the scalable image coder based on block tree coding (BTC) [40]. However, BTC proceeds along the wavelet coefficient tree structure and does not consider the line-based organization of common external memory structures [5,6] of sensor nodes with low random access memory (RAM). In this study, we develop and evaluate a scalable two-line wavelet image coder that accesses the externally stored image data line by line.

The original backward encoding approach [37–39] as well as the two-line approach [17,18] do not support scalable image coding. A number of extensions of the backward coding approach have sought to add the scalable encoding feature [41–44] while maintaining the relatively low memory requirement of approximately 20 kByte for a 256×256 image. In this article, we introduce and evaluate a complementary scalable coding extension to the two-line image coder [17,18] that retains the low 1.5 kByte memory requirement for a 256×256 image, but provides flexible scaling of encoded image size (in encoding bits per byte of image data) and image quality.

2. Principles and notation for wavelet coding

2.1. Notation for sets of wavelet coefficients

Before the coding technique to be described can compress the image, a wavelet transform has to be computed. For computation of the transform we apply the low-memory fractional wavelet filter method [45–47]. While the fractional wavelet filter requires multiple line-based accesses to the sensor node flash memory, it has the least RAM requirements among the currently available low-memory wavelet transform approaches. The coding approach introduced in this article is completely independent from the wavelet transform. Thus, future improved low-memory wavelet transform approaches can be readily combined with our proposed coding approach. The wavelet coefficients are computed and represented using a 16 bit fixed-point format.

The dimension of the original image is denoted by N (N columns and N rows of image pixels). The initial wavelet transform (transform level $L = 1$) of the original image computes four subbands, namely subbands *LL*, *LH*, *HL*, and *HH*. The transform can be repeated on the *LL* subband, resulting in four subbands of the next higher wavelet transform level $L = 2$, see Fig. 1(a) for a two-level transform. A subband has the dimension $N/2^L$, where we typically perform the transform for the levels $L = 1, 2, \ldots, 6$ (in general, higher levels do not result in more compression). Note that the transformed image that is taken as a basis for the compression also has the dimension N, similarly as the original image.

We introduce a *line-wise* notation for the wavelet transform coefficients, where sets of four coefficients are located within two successive lines of a wavelet subband. Specifically, we denote

Please cite this article in press as: S. Rein, M. Reisslein, Scalable line-based wavelet image coding in wireless sensor networks, J. Vis. Commun. (2016), http://dx.doi.org/10.1016/j.jvcir.2016.07.006

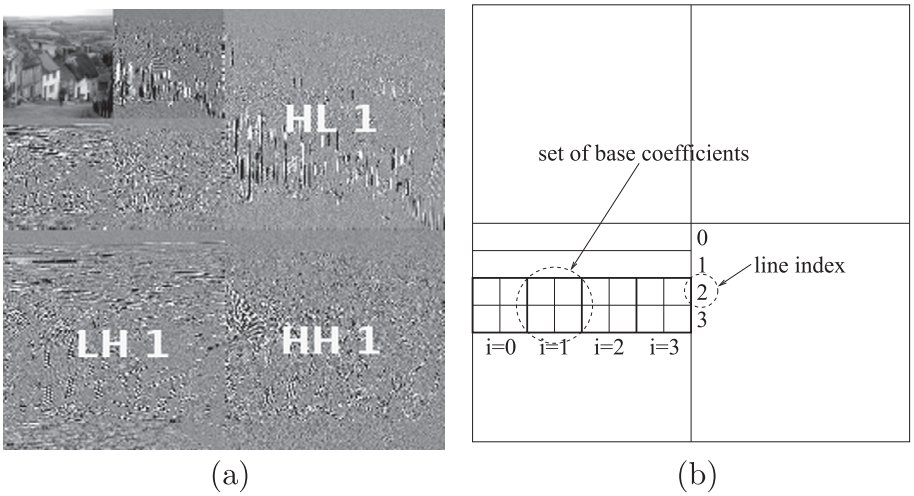

Fig. 1. Illustration of wavelet transform and line-wise notion of sets of wavelet transform coefficients. Figure (a) shows the subbands of a two-level transform. For an image with dimension $N = 16$, figure (b) illustrates the two lines $l = 2$ and $l + 1 = 3$ within the wavelet subband $b = LH$ of the first transform level $L = 1$, denoted as the two single-line matrices $\mathcal{L}_{L=1,b=LH,l=2}$ and $\mathcal{L}_{L=1,b=LH,l=3}$; each line contains $N/2^L = 8$ coefficients indexed by the horizontal coefficient position $h = 0, 1, \ldots, N/2^L - 1 = 7$. The two lines contain the base sets $\mathcal{B}_{L=1,b=LH,l=2}(i),\ i = 0, 1, 2, N/2^{L+1} - 1 = 3$, whereby i indicates the horizontal base set position. Each base set contains four wavelet transform coefficients. For instance, base set $\mathcal{B}_{L=1,b=LH,l=2}(i = 1)$, indicated by the dashed circle, contains the wavelet transform coefficients $\mathcal{L}_{1,LH,2}(h = 2),\ \mathcal{L}_{1,LH,2}(h = 3),\ \mathcal{L}_{1,LH,3}(h = 2),\ \mathcal{L}_{1,LH,3}(h = 3)$.

$\mathcal{L}_{L,b,l}$ for the row l of wavelet transform coefficients from wavelet transform level L and subband b, $b = LH, HL, HH$, where $l = 0$ denotes the top line of the subband.

We denote the *base* set of four coefficients within the two lines l and $l + 1$ at horizontal base set position $i = 0, 1, 2, \ldots, N/2^{(L+1)} - 1$ by

$$\mathcal{B}_{L,b,l}(i) = \begin{pmatrix} \mathcal{L}_{L,b,l}(2i) & \mathcal{L}_{L,b,l}(2i+1) \\ \mathcal{L}_{L,b,l+1}(2i) & \mathcal{L}_{L,b,l+1}(2i+1) \end{pmatrix}. \quad (1)$$

For instance, the dashed circle in Fig. 1(b), indicates the base set $\mathcal{B}_{L=1,b=LH,l=2}(i = 1)$. The proposed coding algorithm traverses the sets of a line one-by-one from left ($i = 0$) to right ($i = N/2^{(L+1)} - 1$).

Wavelet transform coefficients within an image generally exhibit a tree-structure, which is utilized by the encoding algorithm. With growing distance from the root, the coefficient values in the tree tend to decrease. The two lines $\mathcal{L}_{L,b,l}$ and $\mathcal{L}_{L,b,l+1}$ have child (successor) lines at $\mathcal{L}_{L-1,b,2l}$ and $\mathcal{L}_{L-1,b,2l+1}$ as well as $\mathcal{L}_{L-1,b,2(l+1)}$ and $\mathcal{L}_{L-1,b,2(l+1)+1}$, that is, each single line has two child lines in the next lower wavelet level, see Fig. 2. To address the coefficients in the child lines we denote the set of four *child sets* (containing sixteen *child coefficients*) of the base set $\mathcal{B}_{L,b,l}(i)$ as

$$\mathcal{C}_{L,b,l}(i) = \begin{pmatrix} \mathcal{B}_{L-1,b,2l}(2i) & \mathcal{B}_{L-1,b,2l}(2i+1) \\ \mathcal{B}_{L-1,b,2l+2}(2i) & \mathcal{B}_{L-1,b,2l+2}(2i+1) \end{pmatrix}. \quad (2)$$

We also use the term *super set* to denote these units of sixteen coefficients (the super set is *superior* to each of the contained base sets). Each of the four included sets again has four child sets, and thereby a tree including all coefficients of a subband can be constructed. We denote the set of all *descending* child sets of a base set $\mathcal{B}_{L,b,l}(i)$ as $\mathcal{D}_{L,b,l}(i)$, that is, the set $\mathcal{D}_{L,b,l}(i)$ contains the child coefficients of $\mathcal{B}_{L,b,l}(i)$, the grand children, and so on. The set $\mathcal{B}^{+}_{L,b,l}(i)$ contains the set $\mathcal{B}_{L,b,l}(i)$ and $\mathcal{D}_{L,b,l}(i)$, see Table 1 for a list of all denoted sets.

2.2. Notation for quantization levels

A *quantization level* (for brevity also referred to as *level*) serves as a specific coefficient bound and is represented as a numeric bit position of the considered coefficient(s). The numeric value 0 relates to the right-most bit (i.e., the least significant bit). In the following example, the bit position 2 is marked: $\underset{}{0\ 1\ 0\ \underset{\uparrow}{1}\ 1\ 0}$. We distinguish between quantization levels that are (i) directly derived from given (sets of) coefficients, and (ii) quantization levels that are prescribed (selected) for the encoding process of coefficients or levels.

2.2.1. Quantization levels of (sets of) coefficients

The *maximum* quantization level $q(c)$ of a coefficient c characterizes the bit position of the most significant non-zero bit. For example, the coefficient $c_2 = 001110$ given in binary notation has the maximum quantization level $q(c_2) = 3$. The *minimum* quantization level of a coefficient is the bit position of the least significant non-zero bit. For the given example $c_2 = 001110$, the minimum quantization level equals 1. Maximum quantization levels can also be given for sets of coefficients. In that case, the maximum of all levels in the considered set of coefficient is calculated.

Table 2 summarizes our notation for quantization levels. Note that when we address maximum quantization levels of the coefficients of a set, we generally assume inclusion of all descending coefficients of the set. However, note that the super set quantization level $q(D_{L,b,l}(i))$ denotes the quantization level required to encode the super set $C_{L,b,l}(i)$; the superset quantization level does not denote the quantization level of the individual coefficients of the super set.

2.2.2. Quantization levels as encoding bounds

When encoding sets of coefficients, in general only an excerpt of the original representation (bit sequence) of the coefficient is selected (thus achieving compression). The quantization level of the original coefficient that serves as the most significant bit in the encoded representation is called *upper bound*, while the level that specifies the encoded least significant bit is called the *lower bound*. The lower bound for all coefficients of an image is generally denoted as $q_{\min}$. For example, encoding the coefficient 0111 with the upper bound 2 and the lower bound 1, results in the encoded sequence 11.

When decoding a given sequence, the decoder must, of course, be aware of the employed upper and lower encoding bounds. Therefore, the respective bounds/quantization levels have to be

Please cite this article in press as: S. Rein, M. Reisslein, Scalable line-based wavelet image coding in wireless sensor networks, J. Vis. Commun. (2016), http://dx.doi.org/10.1016/j.jvcir.2016.07.006

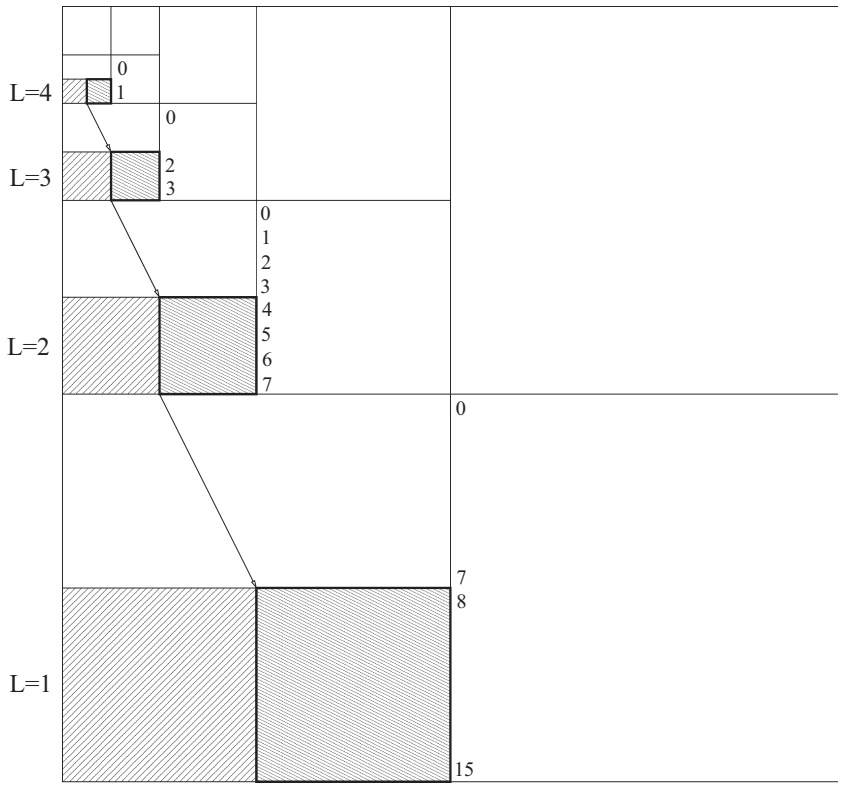

Fig. 2. Illustration of tree structure of the wavelet transform coefficients for an image of dimension $N = 32$ and four wavelet transform levels. Each line l in a given subband b at a given level L has two child (successor) lines at lines $2l$ and $2l + 1$ in subband b of the next lower level $L - 1$. As indicated by the gray shading, line $l = 1$ in subband $b = LH$ of transform level $L = 4$ has the two child lines 2 and 3 in subband $b = LH$ of transform level $L = 3$. Then, in turn, the base set of four coefficients $\mathcal{B}_{L=3,b=LH,l=0}(i = 1)$, which is highlighted by the square with the thick lines at level $L = 3$, has sixteen child coefficients denoted by $\mathcal{C}_{L=3,b=LH,l=0}(i = 1)$ and illustrated by the thick-line square on the right half of lines 4–7.

Table 1
Overview of the coefficient notation.

$\mathcal{B}_{L,b,l}(i)$	*Base set* of 4 coefficients at transform level L, subband b, line l at horizontal base set position i
$\mathcal{C}_{L,b,l}(i)$	16 *child* coefficients of base set $\mathcal{B}_{L,b,l}(i)$, also called *super set*
$\mathcal{D}_{L,b,l}(i)$	All *descending* coefficients of $\mathcal{B}_{L,b,l}(i)$, including $\mathcal{C}_{L,b,l}(i)$, children of $\mathcal{C}_{L,b,l}(i)$, and so on
$\mathcal{B}^{+}_{L,b,l}(i)$	Base set $\mathcal{B}_{L,b,l}(i)$ and all descending coefficients in $\mathcal{D}_{L,b,l}(i)$
$D_{L+1,b,l/2}(i/2)$	= the sets $B^{+}_{L,b,l}(i)$, $B^{+}_{L,b,l}(i+1)$, $B^{+}_{L,b,l+2}(i)$, and $B^{+}_{L,b,l+2}(i+1)$

Table 2
Overview of the quantization level notation.

$q(c)$	Quantization level of a single coefficient c
$q\left(\mathcal{B}^{+}_{L,b,l}(i)\right)$	Maximum quantization level of $\mathcal{B}_{L,b,l}(i)$ and $\mathcal{D}_{L,b,l}(i)$, calculated as the maximum of the quantization levels of all coefficients included in the set $\mathcal{B}^{+}_{L,b,l}(i)$
$q(\mathcal{D}_{L,b,l}(i))$	Maximum quantization level of $\mathcal{D}_{L,b,l}(i)$ only, also called *super set quantization level*
$q(D_{L+1,b,l/2}(i/2))$	= Maximum of the levels $q\left(B^{+}_{L,b,l}(i)\right)$, $q\left(B^{+}_{L,b,l}(i+1)\right)$, $q\left(B^{+}_{L,b,l+2}(i)\right)$, and $q\left(B^{+}_{L,b,l+2}(i+1)\right)$

encoded in advance. When encoding a quantization level, similarly as when encoding a coefficient, an upper bound has to be employed. For example, encoding the level 000100 results in the encoded sequence 01 when 3 is chosen as upper bound. The following zero bits do not need to be sent, as a quantization level has only a single non-zero bit. Therefore, when the decoder decodes a level and encounters the first non-zero bit, the decoder knows that this is the last bit relating to that level. This last bit is thus also called the *stop bit* for the level.

3. Scalable wavelet image compression

3.1. Basic principle: base and scalable stream

The basic application setting of the introduced scalable image compression is that there is an image of high quality at the sender side (original image), which needs to be communicated in different requested qualities $q_{\min}$ to the receiver, whereby the previously sent version with lower quality of the image is reused. We now consider the application scenario of *refining* a previously sent image, see Fig. 3. We distinguish between a previous and a current/new quality, denoted by minimum quantization levels $q_{\min\text{Prv}}$ and $q_{\min}$, respectively, where the condition $q_{\min\text{Prv}} > q_{\min}$ holds true (a smaller minimum quantization level refers to a higher image quality). In a first run, the original image had been encoded to a *base stream* with the quality $q_{\min\text{Prv}}$. The base stream was communicated to the receiver and decoded there. The base stream was encoded and decoded with the Wavelet Image Two-Line Coder (Wi2l) [17,18]. Now, in a second run, the receiver requests a higher quality $q_{\min}$ of the image. The sender then encodes the *scalable stream* with the proposed scalable line-based wavelet image coder. The scalable stream is used at the receiver side together with the previously sent base stream to refine the image, thus achieving the new quality $q_{\min}$.

The receiver can now request a second refinement of the image. Following the request, the sender encodes a new scalable stream to allow the refinement at the receiver side. Note that we again denote the new quality with $q_{\min}$ and the previous quality with $q_{\min\text{Prv}}$, see Fig. 4. The quality $q_{\min\text{Prv}}$ equals the quality $q_{\min}$ in Fig. 3. For the refinement, the receiver uses the new scalable stream and a base stream to decode the image in the new quality

Please cite this article in press as: S. Rein, M. Reisslein, Scalable line-based wavelet image coding in wireless sensor networks, J. Vis. Commun. (2016), http://dx.doi.org/10.1016/j.jvcir.2016.07.006

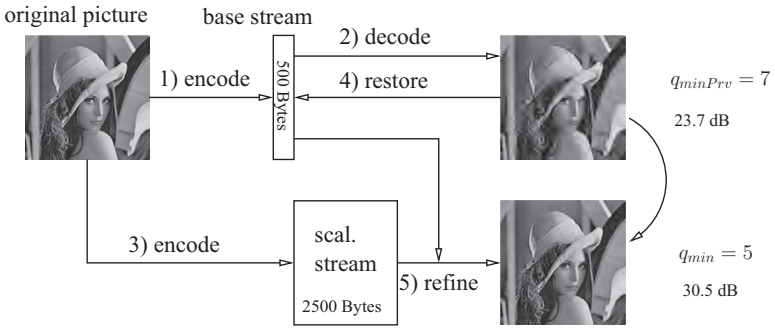

Fig. 3. Principle of updating a base stream with the quality $q_{\text{minPrv}} = 7$ to an image with a higher quality and correspondingly smaller quantization level $q_{\text{min}} = 5$. The scalable algorithm reuses the previously sent base stream and the new scalable stream to decode a refined version of the image.

$q_{\text{min}} = 3$. The employed base stream is constructed at the receiver side from the previously decoded image with quality $q_{\text{minPrv}} = 5$ using the Wi2l coder and is a different base stream than in Fig. 3. The constructed base stream contains quantization level information (which is generally required for tree-based wavelet coding) for the refinement of quantization levels and coefficients.

In principle, it is also possible to use the two previous streams (base and scalable) at the receiver instead of constructing a new base stream from the previous image. However, the previous streams may not be available at the receiver any more (while the previous image is still there). Note that the base stream at the receiver side is built on the fly to deliver the missing information when processing the scalable stream; thus, it does not require much additional random access memory.

The outlined on-request successive refinements require that the entire image is encoded again, each time a new refinement is requested. An alternative application scenario of the proposed encoder is to pre-specify some quantization levels, e.g., base stream $q = 7$, first refinement $q = 5$, second refinement $q = 3$, and encode the entire image once to produce the base stream as well as the scalable streams for the first and second refinement, and store all three streams on the SD card. Then, the sensor node can first send the base stream to the receiver, followed by the first scalable refinement stream and the second scalable refinement stream. A more general alternative application scenario is that a given image is encoded once at the sensor node producing encoded bit streams from some prescribed maximum considered quantization scale (e.g., $q = 9$ in our evaluations, see Section 5) to the smallest quantization scale of $q = 0$. Then, as the receiver requests a base image quality, e.g., with $q = 7$, the encoded streams for $q = 9$, $q = 8$, and $q = 7$ are sent from the sensor node to the receiver and decoded at the receiver into the $q = 7$ base image. If the receiver then requests a refinement to $q = 5$, the pre-encoded streams for $q = 6$ and $q = 5$ are sent from the sensor node to the receiver.

In the following subsections, we describe the encoding of scalable information in detail (the decoding works the reverse way and alternately processes information from the base and scalable stream). Importantly, the introduced algorithm for quality refinements does *not* introduce additional bits (e.g., for signaling) compared to the case of coding and transmitting the image without scalability. In particular, the total size of data (in bits) of the base stream and the scalable stream to achieve the new quality q_{min} equals exactly the size of a base stream that would be required to decode the image with quality q_{min} without using the scalability feature.

We base the Wi2l-scalable encoder on the backward coding principles [36–39]. That is, we start the encoding with the lowest wavelet transform level. This backwards coding reduces the required memory as quantization level information does not need to be retained in memory. More specifically, the backwards coding allows for application of a specific scheme to link the quantization level information from the already encoded coefficients with the required level information for the next coefficients to be encoded. This scheme of exploiting the linking quantization level information for memory-efficient scalable coding is introduced in detail in the following subsections. Note that with the proposed backward coding based scalable coding approach, each scalable stream is backward coded (from lowest to highest wavelet transform level), stored on the SD-card as an intermediate buffer, and then

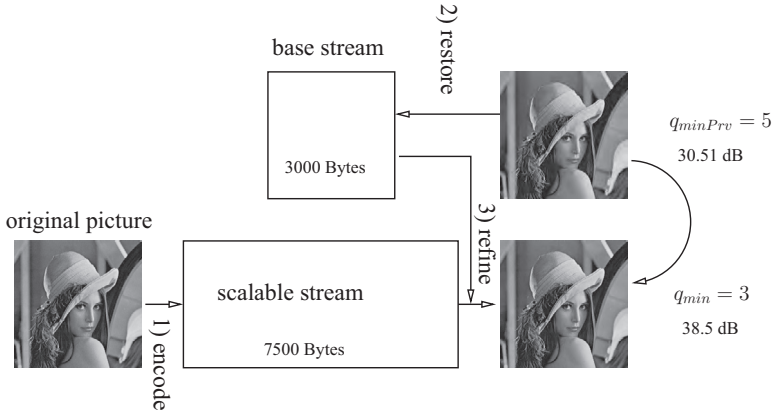

Fig. 4. Second refinement of an image: q_{minPrv} equals now the quality $q_{\text{min}} = 5$ from Fig. 3. The base stream for the quality $q_{\text{minPrv}} = 5$ is now constructed on the fly at the receiver from the image with $q_{\text{minPrv}} = 5$ (as the receiver may have discarded the previously received base and scalable streams).

Please cite this article in press as: S. Rein, M. Reisslein, Scalable line-based wavelet image coding in wireless sensor networks, J. Vis. Commun. (2016), http://dx.doi.org/10.1016/j.jvcir.2016.07.006

sent out in the reverse order (from highest to lowest wavelet transform level) to the receiver.

3.2. Refining coefficients via scalable stream

The scalable stream provides the differential information to refine already communicated wavelet coefficients. In tree-based encoding, the coding is performed in units of four sets, i.e., $\mathcal{B}_{L,b,l}(i)$, $\mathcal{B}_{L,b,l}(i+1)$, $\mathcal{B}_{L,b,l+2}(i)$, and $\mathcal{B}_{L,b,l+2}(i+1)$, where each set contains four coefficients. For each of these four sets, an upper bound is calculated, i.e., for the set $\mathcal{B}_{L,b,l}(i)$, the bound $q\left(\mathcal{B}^{+}_{L,b,l}(i)\right)$ is calculated by taking the maximum of the levels $q(\mathcal{B}_{L,b,l}(i))$ and $q(\mathcal{D}_{L,b,l}(i))$; for the sets $\mathcal{B}_{L,b,l}(i+1)$, $\mathcal{B}_{L,b,l+2}(i)$, $\mathcal{B}_{L,b,l+2}(i+1)$, the upper bounds are calculated similarly. Thus, coefficients and upper bounds of sets of four coefficients need to be encoded. The upper bounds (of four coefficients and their descendants) are in turn encoded in sets of four using the maximum quantization level $q(D_{L+1,b,l/2}(i/2))$, which is the maximum quantization level of the 16 coefficients to be encoded (building a super set when encoding the next higher wavelet level L) and their descendants. Thus, the scalable stream needs to contain refinement information for coefficients and quantization levels.

When a set of coefficients is to be processed, we distinguish between two cases: If the set has not yet been encoded in a previous run, we denote the respective coding operation with the term *encoding*. Coefficient sign bits are part of that operation. Otherwise, if previous coefficient information is available, we refer to the operation as *refinement* of a set. Note that there can be *insignificant* coefficients in such a set – that are coefficients that would actually not require any (new) refinement information. A refinement always requires $q_{\text{minPrv}} - q_{\text{min}}$ bits for each coefficient (including possibly insignificant ones). Coefficients that have been insignificant in all previous runs and now become significant will require an additional sign bit. This is the case if the quantization level $q(c)$ of the coefficient is smaller than q_{minPrv}. For example, a given coefficient $c = 000101$ has quantization level $q(c) = 2$; if we consider $q_{\text{minPrv}} = 4$, then $q(c) < q_{\text{minPrv}}$, thus the coefficient has not been encoded in the previous run.

Therefore, if a set of coefficients is to be processed, it is first checked if the set has already been encoded in a previous run, resulting in a refinement if $q(\mathcal{B}_{L,b,l}(i)) \geqslant q_{\text{minPrv}}$. If that is the case, $q_{\text{minPrv}} - 1$ serves as upper bound for the set, otherwise the actual maximum quantization level is used as upper bound for encoding. We therefore use the term *Refine/Encode* in the following algorithm description. As an example we select four coefficients $c_1 = 22$, $c_2 = 19$, $c_3 = 3$, and $c_4 = 1$ of a base set $\mathcal{B}_{L,b,l}(i)$ to be encoded to the scalable stream, which are given in binary notation as follows:

$$\begin{array}{lll|ll|l} c_1 = & 1 & 0 & 1 & 1 & 0 \\ c_2 = & 1 & 0 & 0 & 1 & 1 \\ c_3 = & 0 & 0 & 0 & 1 & 1 \\ c_4 = & 0 & 0 & 0 & 0 & 1 \\ & & \uparrow & & \uparrow & \\ & & q_{\text{minPrv}} & & q_{\text{min}} & \\ & & = 3 & & = 1 & \end{array}$$

We consider the maximum quantization levels $q\left(\mathcal{B}^{+}_{L,b,l}(i)\right) = 4$ and assume $q(\mathcal{D}_{L,b,l}(i)) = 2$. As minimum quantization levels we select $q_{\text{minPrv}} = 3$ and $q_{\text{min}} = 1$. We can conclude now that the set has already been encoded in a previous run, since $q\left(\mathcal{B}^{+}_{L,b,l}(i)\right) = 4 \geqslant q_{\text{minPrv}} = 3$. That means, that for c_i, $i = 1, \ldots, 4$, the binary sequences 11, 01, 01, and 00 have to be encoded. As c_3 and c_4 have been insignificant in the previous run, they require a sign bit as well, if they become significant.

To give another example, we set the coefficients c_1 to 00010 and c_2 to 00011. We assume a maximum quantization level $q\left(\mathcal{B}^{+}_{L,b,l}(i)\right) = 1$ of the set. As $q\left(\mathcal{B}^{+}_{L,b,l}(i)\right) = 1 < q_{\text{minPrv}} = 3$, the set has not yet been encoded before, and $q\left(\mathcal{B}^{+}_{L,b,l}(i)\right) = 1$ serves as the upper bound for encoding each of the coefficients in the set Thus, the sequence 1110 is encoded for the four coefficients, and each of the three coefficients c_1, c_2, c_3 now requires a sign bit. Note that the coefficient c_4 is coded without a sign bit, as it is still insignificant. Its sign bit would be coded in a next run with a $q_{\text{min}} = 0$.

For the coding of quantization levels, there exists similarly as for the coding of coefficients an actual lower encoding bound and an upper encoding bound. If a level has not yet been encoded in a previous run, it is encoded as usual in the tree-based coding with the bit positions for the upper and lower bounds. The lower bound is given as the maximum of q_{min} and the level to be coded itself, whereby q_{min} defines the new quality. (The first non-zero bit thus serves as a stop bit, as noted in Section 2.2.2.)

The upper encoding bound is selected due to the type of quantization level; there exist two types of quantization levels, levels of base sets that refer to 4 coefficients and levels of super sets, referring to 16 coefficients. Quantization levels of base sets are encoded in groups of 4, that is, taking the base sets $\mathcal{B}^{+}_{L,b,l}(i)$, $\mathcal{B}^{+}_{L,b,l}(i+1)$, $\mathcal{B}^{+}_{L,b,l+2}(i)$, and $\mathcal{B}^{+}_{L,b,l+2}(i+1)$, the four respective quantization levels are encoded altogether with a common upper bound given as $q(\mathcal{D}_{L+1,b,l/2}(i/2))$. The second type of quantization level, which is a quantization level of a super set, given as $q(\mathcal{D}_{L,b,l}(i))$, uses $q\left(\mathcal{B}^{+}_{L,b,l}(i)\right)$ as an upper bound. To explain the wavelet level $L+1$, we note that the mentioned four base sets at a certain wavelet level L build a super set at the next higher wavelet level $L+1$. More precisely, the sets $\mathcal{B}^{+}_{L,b,l}(i)$, $\mathcal{B}^{+}_{L,b,l}(i+1)$, $\mathcal{B}^{+}_{L,b,l+2}(i)$, and $\mathcal{B}^{+}_{L,b,l+2}(i+1)$ equal the set $\mathcal{D}_{L+1,b,l/2}(i/2)$.

As we introduce the scalable coding, there can be the case that a level has already been encoded in a previous run. The upper encoding bound, i.e., $q\left(\mathcal{B}^{+}_{L,b,l}(i)\right)$ or $q(\mathcal{D}_{L+1,b,l/2}(i/2))$, is then larger or equal than the previous minimum quantization level $q_{minPr\upsilon}$. In that case, no new information needs to be conveyed. If, however, the level has been encoded as zero in the previous run, the level is *refined* using $q_{minPr\upsilon} - 1$ as the upper encoding bound. As an example, we consider the three (super set) quantization levels $q(\mathcal{D}(1)) = 1$, $q(\mathcal{D}(2)) = 3$, and $q(\mathcal{D}(3)) = 0$, with the assumed upper encoding bounds as given in the following table (for simplicity we leave the wavelet level, band, and line indices out, each level relates to the given upper bound):

$$\begin{array}{lll|ll|ll} q(\mathcal{D}(1)) = & (0 & 0 & 0 & 1 & 0)_2 & \text{with } q(\mathcal{B}^{+}(1)) = 1 \\ q(\mathcal{D}(2)) = & (0 & 1 & 0 & 0 & 0)_2 & \text{with } q(\mathcal{B}^{+}(2)) = 4 \\ q(\mathcal{D}(3)) = & (0 & 0 & 0 & 0 & 1)_2 & \text{with } q(\mathcal{B}^{+}(3)) = 3 \\ & & \uparrow & & \uparrow & & \\ & & q_{\text{minPrv}} & & q_{\text{min}} & & \\ & & = 3 & & = 1 & & \end{array}$$

The level $q(\mathcal{D}(1))$ has not yet been encoded ($q(\mathcal{B}^{+}(1)) = 1 < q_{minPr\upsilon} = 3$), thus the bit 1 is sent using $q(\mathcal{B}^{+}(1))$ as an upper bound. The level $q(\mathcal{D}(2))$ has already been sent in a previous run ($q(\mathcal{B}^{+}(2)) = 4 \geqslant q_{minPr\upsilon} = 3$), no information needs to be conveyed. The level $q(\mathcal{D}(3))$ has already been sent as well ($q(\mathcal{B}^{+}(3)) = 3 \geqslant q_{minPr\upsilon} = 3$) but it has been zero. The sequence 00 is sent using $q_{minPr\upsilon} - 1$ as upper bound.

Please cite this article in press as: S. Rein, M. Reisslein, Scalable line-based wavelet image coding in wireless sensor networks, J. Vis. Commun. (2016), http://dx.doi.org/10.1016/j.jvcir.2016.07.006

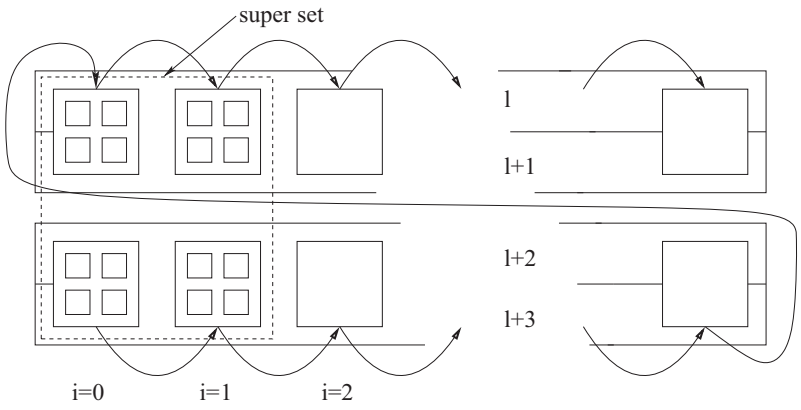

Fig. 5. Encoding of four lines: The algorithm traverses the four lines in sets of four coefficients, starting with the lower two lines, and then it continues with the next upper two lines. Information that relates to a unit of 16 coefficients (which become a super set when the next higher wavelet level is encoded) is saved via a specific buffer, see Fig. 6.

3.3. Encoding of four lines

In the previous section we have described, how single coefficients in the base sets are encoded into the scalable stream. In this section, we give a formal description of the scalable algorithm for encoding of four lines of a wavelet subband. In principle, a complete image can be encoded using such sets of four consecutive lines. We distinguish between the *lower* two lines (line numbers $l+2$ and $l+3$, where l denotes a line of a wavelet subband) and the *upper* two lines (l and $l+1$) of such a set, see Fig. 5. The algorithm starts to encode base sets $\mathcal{B}_{L,b,l}(i),\ i=0,1,\ldots,\frac{N}{2^{L+1}}-1$, of four coefficients in the lower two lines from the left to the right side. When the last set of the lower two lines has been reached and processed, the algorithm continues with the left-most set of the upper two lines.

For the encoding of a coefficient base set $\mathcal{B}_{L,b,l}(i)$, the maximum quantization level information $q(\mathcal{D}_{L,b,l}(i))$ of 16 descending coefficients is required; this level serves to compute the maximum quantization level of the base set $q\left(\mathcal{B}^{+}_{L,b,l}(i)\right)$, which in turn serves as the upper bound for encoding the coefficients in the base set $\mathcal{B}_{L,b,l}(i)$. We can conclude that for encoding the given base set, the complete set $\mathcal{D}_{L,b,l}(i)$ needs to be available, that is, all the sub-trees that have their origin in one of the coefficients of the base set. That is why the classical wavelet coding schemes traverse the coefficients within a *tree* and frequently pass the sub-trees to calculate the required level information. In this paper, however, a *line*-based mechanism is introduced, thus a special mechanism is necessary in order to keep the required information. For this purpose, we employ a specific *buffer* that saves maximum quantization level information, similarly as in [17,18], where the size of this buffer is $\sum_{l=1}^{(\log_2 N)-2}\frac{N}{2^{l+1}}$ Bytes (N is the picture pixel dimension). The interaction of the proposed algorithm with this buffer is illustrated in Fig. 6.

Fig. 6 illustrates in part (a) the retrieval of the maximum quantization levels $\mathcal{D}_{L,b,l+2}(i)$ and $\mathcal{D}_{L,b,l+2}(i+1)$ from the buffer (which have been saved when encoding the respective descending super set) for all descending coefficients of $\mathcal{B}_{L,b,l+2}(i)$ and $\mathcal{B}_{L,b,l+2}(i+1)$. The retrieved levels are employed to calculate the maximum quantization levels $q\left(\mathcal{B}^{+}_{L,b,l+2}(i)\right)$ and $q\left(\mathcal{B}^{+}_{L,b,l+2}(i+1)\right)$, which are in turn required for encoding the two base sets $\mathcal{B}_{L,b,l+2}(i)$ and $\mathcal{B}_{L,b,l+2}(i+1)$. The two levels are saved to the buffer for later retrieval, see Fig. 6 (c). In Fig. 6(b), the base sets $\mathcal{B}_{L,b,l}(i)$ and $\mathcal{B}_{L,b,l}(i+1)$) are coded, and similarly as before the respective level information is retrieved from the buffer to compute the maximum quantization levels $q\left(\mathcal{B}^{+}_{L,b,l}(i)\right)$ and $q\left(\mathcal{B}^{+}_{L,b,l}(i+1)\right)$ of the two sets. However, this time the level information is not saved to the buffer but kept temporarily. Fig. 6(c) illustrates that these levels together with the levels saved in (a) are employed to calculate the maximum quantization level $q(\mathcal{D}_{L+1,b,l/2})$ to be saved to the buffer for later retrieval when encoding coefficients of the next higher level.

We summarize the three types of operations in the encoding algorithm to be described: (1) interaction with the buffer to *save* or *retrieve* maximum quantization level information, (2) computation of maximum quantization level information, and (3) encoding or refining of coefficient or level information. In the following we give a formal description of the proposed encoding scheme, starting with the description of the lower two lines and continuing with the two upper two lines. As a parameter, the selected wavelet subband b, the subband line l, and the levels $q_{\min}$ and $q_{\min\mathrm{Prv}}$ for the new and previous quality are used. Note that we present a text-oriented description to make the scheme more readable, while it still highly related to the employed source code in the C programming language, which is freely available from http://mre.faculty.asu.edu/Wi2l. For improved readability, we introduce the shorthand notations q_{16} for the super set maximum quantization level, which has been denoted $q(\mathcal{D}_{L,b,l+2})$ so far, and q_4 for the base set quantization levels, which has been denoted by $q\left(\mathcal{B}^{+}_{L,b,l+2}(i)\right),\ q\left(\mathcal{B}^{+}_{L,b,l+2}(i+1)\right), q\left(\mathcal{B}^{+}_{L,b,l}(i)\right),\ q\left(\mathcal{B}^{+}_{L,b,l}(i+1)\right)$ so far.

3.3.1. Refine (two lower) lines $l+2$ and $l+3$

Repeat the following steps (1.a) to (1.f) for horizontal positions $i=0,\ 2,\ 4,\ldots,\ \frac{N}{2^{L+1}}-2$:

(1.a) Retrieve $q_{16}=q(\mathcal{D}_{L,b,l+2}(i))$

(1.b) Calculate $q_4=q\left(\mathcal{B}^{+}_{L,b,l+2}(i)\right)$ using the coefficients in $\mathcal{B}_{L,b,l+2}(i)$ and the level q_{16}. Keep that level for step (1.f).

(1.c) Refine/Encode q_{16} retrieved in step (1.a) using $q_{\min\mathrm{Prv}}-1$ as upper bound if $q_4 \geqslant q_{\min\mathrm{Prv}}$ (refinement) and the level q_{16} has been zero, otherwise use q_4 as upper bound. The first non-zero bit serves as a stop-bit.

(1.d) Refine/Encode the coefficients in $\mathcal{B}_{L,b,l+2}(i)$ using $q_{\min}$ as lower bound. Use $q_{\min\mathrm{Prv}}-1$ as upper bound if $q_4 \geqslant q_{\min\mathrm{Prv}}$ (refinement), otherwise use q_4 as upper bound (set has no yet been encoded in a previous run). Encode a sign bit for significant coefficients that have been insignificant in the previous runs.

(1.e) Repeat the steps (1.a)–(1.d) for $i=i+1$

(1.f) Save the two levels $q\left(\mathcal{B}^{+}_{L,b,l+2}(a)\right),\ a=i,\ i+1$, calculated in step (1.b) for later retrieval, see step (2.f) below.

Please cite this article in press as: S. Rein, M. Reisslein, Scalable line-based wavelet image coding in wireless sensor networks, J. Vis. Commun. (2016), http://dx.doi.org/10.1016/j.jvcir.2016.07.006

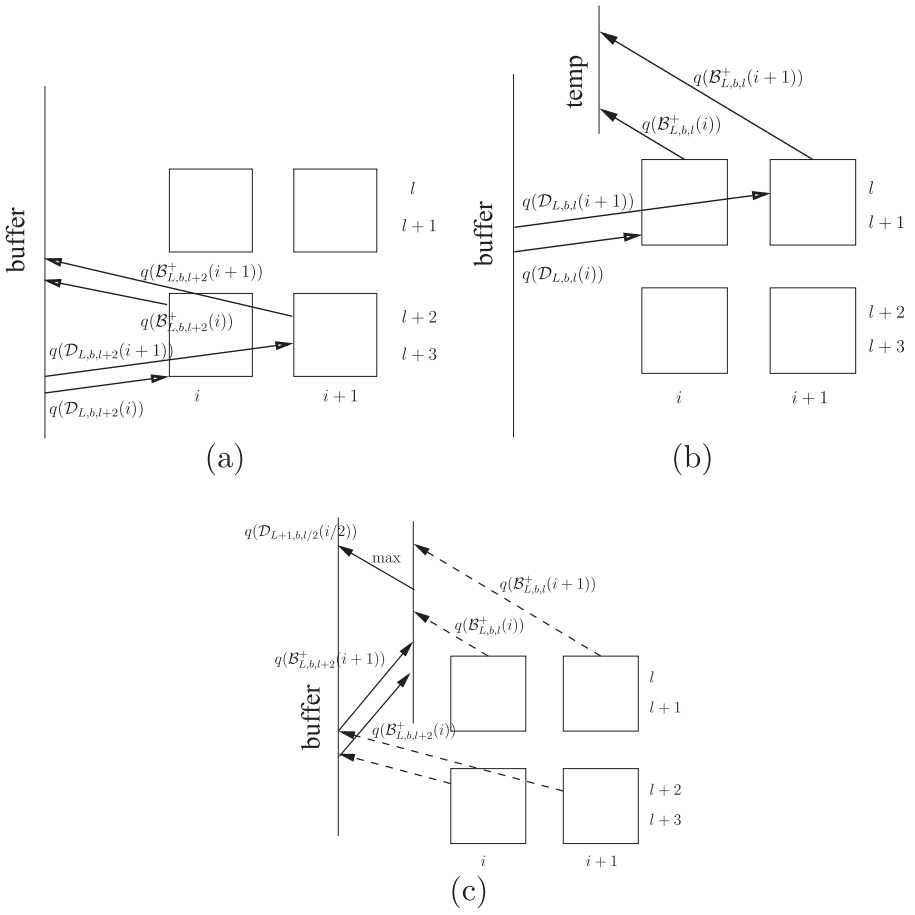

Fig. 6. Interaction with the *buffer* to encode a set of 16 coefficients. In (a), sets i and $i+1$ in the lower two lines are encoded. The levels $q(\mathcal{D}_{L,b,l+2}(i))$ and $q(\mathcal{D}_{L,b,l+2}(i+1))$ are retrieved to compute $q\left(\mathcal{B}^+_{L,b,l+2}(i)\right)$ and $q\left(\mathcal{B}^+_{L,b,l+2}(i+1)\right)$, which are saved to the buffer. In (b), the levels $q\left(\mathcal{B}^+_{L,b,l}(i)\right)$ and $q\left(\mathcal{B}^+_{L,b,l}(i+1)\right)$ are computed, but this time only temporarily kept for usage in (c), where they are used together with the information from (a) to calculate the maximum quantization level $q(\mathcal{D}_{L+1,b,l/2}(i/2))$.

3.3.2. Refine (two upper) lines l and $l+1$

Steps (2.a)–(2.e) are the same as steps (1.a)–(1.e) which refine lines $l+2$ and $l+3$; the only modification is that the parameters $l+2$ and $l+3$ in steps (1.a)–(1.e) need to be replaced with l and $l+1$, respectively, for steps (2.a)–(2.e).

(2.f) Retrieve $q\left(\mathcal{B}^+_{L,b,l+2}(a)\right)$, $a=i,\ i+1$, saved in step (1.f) when refining the lines $l+2$ and $l+3$.
(2.g) Calculate $q(\mathcal{D}_{L+1,b,l/2}(i/2))$ by taking the maximum of the levels calculated in (2.b) and retrieved in (2.f).
(2.h) Refine/Encode $q\left(\mathcal{B}^+_{L,b,l+a}(i)\right)$, $a=0,\ 2$, and $q\left(\mathcal{B}^+_{L,b,l+a}(i+1)\right)$, $a=0,\ 2$, using either $q_{\text{minPrv}}-1$ or the level from step (2.g) as upper bound, similarly as in step (1.c).
(2.i) Save the level calculated in step (2.g) to be retrieved in step (1.a) when encoding lines of higher wavelet levels.

3.4. Recursive encoding of complete image

We have so far described how four consecutive lines of a subband are encoded with to the proposed scalable method. To encode a complete image: (1) the encoding of four lines has to be repeated such that a complete wavelet subband is encoded, (2) the action of (1) has to be repeated until all of the three subbands *HL*, *LH*, and *HH* have been processed, and (3) the last level *LL*-subband with 16 coefficients has to be encoded similarly as described in Section 3.2, including the maximum quantization levels of each of the subbands and the maximum quantization level of the complete image; this level serves as an upper bound for encoding the coefficients of the last subband and is encoded in turn using a pre-defined maximum level (in our implementation we have selected $q_{\max}=14$).

Regarding the ordering of encoding of subband lines, we propose to traverse the lines recursively such that the required quantization levels of lower subbands are calculated next prior to the encoding of two lines. That is, before two lines $l,\ l+1$ in level L are encoded, the child lines $2l+a,\ a=0,\ 1,\ 2,\ 3$, in level $L-1$ are encoded. This will allow for usage of a minimally sized buffer for saved maximum quantization levels, thus saving random access memory. The proposed recursive traversal order is achieved with the following small modification of step (a) of the procedures for coding lines $l,\ l+1$ and $l+2,\ l+3$ in Section 3.3:

(a) Refine lines $2(l+2)+a,\ a=0,\ 1,\ 2,\ 3$ at level $L-1$, and then retrieve $q_{16}=q(\ldots)$ and
(a) Refine lines $2l+a,\ a=0,\ 1,\ 2,\ 3$ at level $L-1$, and then retrieve $q_{16}=q(\ldots)$

The modification first encodes the respective four child lines before the two new lines are processed (the child lines trigger in turn the encoding of child lines). Due to the recursion, the complete subband can then be encoded via encoding of the lines $l=0,\ 1,\ 2,\ 3$. of the subbands in the last level.

4. Computational cost

In this section, we analyze the access pattern of the introduced line-based scalable coding scheme and compare it to the classical tree-based approach, i.e., the SPIHT algorithm. The access patterns

Please cite this article in press as: S. Rein, M. Reisslein, Scalable line-based wavelet image coding in wireless sensor networks, J. Vis. Commun. (2016), http://dx.doi.org/10.1016/j.jvcir.2016.07.006

of the considered algorithms reflects the read and write access of the wavelet coefficients. The algorithms do rather not perform mathematical calculations but describe an order in which the coefficients in the transformed image have to be accessed; thus, the computational cost is determined by the order of coefficient access. Second, there exist memory buffers that store coefficient quantization level information, and access to these buffers also accounts for the computational cost. In this work, the buffers for quantization level information are considered to be located in the RAM memory, while the transformed wavelet image is located on flash memory (with substantially slower access times).

An exact analysis of computational cost and the corresponding coding times is not possible for the algorithm, as each image has a different compressibility, resulting in an individual series of algorithm steps and different computational cost for each given image. We analyze therefore the principal steps of each of the two coding schemes and evaluate the computational cost specifically for the worst case of no compressibility. As we will explain later, the total number of read operations for wavelet coefficients does not change with a different compressibility, as for a new minimum quantization level $q_{\min}$ also the already encoded sets of coefficients have to be checked for possible updates. We first consider the proposed scalable scheme and then continue with SPIHT.

We consider images of dimension $N \times N$ pixels. In order to capture the time-consuming coding operations, we denote r and c for *reading* and *coding* a coefficient and denote cl for the coding of a *coefficient level*. These operations generally require flash memory accesses in the considered sensor node platform. For intermediate storage of wavelet level information, the operators g and p are introduced in order to *get* or *put* levels to the RAM memory buffer. (We note that the number of writing operations (to flash memory in the scalable Wi2l approach and to RAM memory in the SPHIT approach) correspond to the size (in Bytes) of the compressed image.).

4.1. Scalable Wi2l

The following consideration is based on the algorithm description in Section 3.3.

4.1.1. Encoding of lower two lines

Let $\phi_{L,b,l+2}$ denote the number of operations for encoding the lower two lines $l+2$ and $l+3$, where L and b denote the wavelet level and wavelet band, respectively. In the two lines, there exist $N/2^{L+1}$ sets of four coefficients. For each set, a quantization level $q(\mathcal{D}_{L,b,l+2}(i))$ and the four coefficients in $\mathcal{B}_{L,b,l+2}(i)$ have to be retrieved, and the retrieved level and coefficients have to be encoded. Furthermore, the level $q\left(\mathcal{B}^{+}_{L,b,l+2}(a)\right)$ needs to be saved to the buffer. The required operations for the two lines are thus

$$\phi_{L>1,b,l+2} = \frac{N}{2^{L+1}}(g + 4r + cl + 4c + p). \tag{3}$$

For the first wavelet level, there are no descendant coefficients. Thus, for $L = 1$ the preceding equation has to be modified in that there is no retrieval and coding of $q(\mathcal{D}_{L,b,l+2}(i))$:

$$\phi_{L=1,b,l+2} = \frac{N}{2^{L+1}}(4r + 4c + p). \tag{4}$$

4.1.2. Encoding of upper two lines

For the upper two lines, the operations are the same with the difference that the level $q\left(\mathcal{B}^{+}_{L,b,l+2}(a)\right)$ needs to be retrieved from the buffer (instead of being saved to the buffer), the quantization levels $q\left(\mathcal{B}^{+}_{L,b,l+a}(i)\right)$, $a = 0,\ 2$, and $q\left(\mathcal{B}^{+}_{L,b,l+a}(i+1)\right)$, $a = 0,\ 2$, need to be refined, and the level $q(\mathcal{D}_{L+1,b,l/2}(i/2))$ to be saved:

$$\begin{aligned}\phi_{L>1,b,l} &= \frac{N}{2^{L+1}}\left(g + 4r + cl + 4c + g + 2cl + \frac{1}{2}p\right)\\ &= \frac{N}{2^{L+1}}(2g + 4r + 3cl + 4c + p/2).\end{aligned} \tag{5}$$

Similarly as for the lower two lines, for $L = 1$ the operations are given as

$$\phi_{L=1,b,l} = \frac{N}{2^{L+1}}(g + 4r + 2cl + 4c + p/2). \tag{6}$$

4.1.3. Encoding of complete level of a wavelet subband

For the encoding of a complete level of a wavelet subband with $N/2^L$ lines, we denote $\phi_{L,b}$ for the total number of operations. With Eqs. (3)–(6), the number of required operations for $L > 1$ are

$$\begin{aligned}\phi_{L>1,b} &= \frac{1}{4}\frac{N}{2^L}(\phi_{L>1,b,l+2} + \phi_{L>1,b,l})\\ &= \frac{N^2}{4^{L+2}}(6g + 3p + 16c + 8cl + 16r),\end{aligned} \tag{7}$$

and for $L = 1$ as

$$\begin{aligned}\phi_{L=1,b} &= \frac{1}{4}\frac{N}{2^L}(\phi_{L=1,b,l+2} + \phi_{L=1,b,l})\\ &= \frac{N^2}{64}(16r + 2g + 3p + 16c + 4cl).\end{aligned} \tag{8}$$

4.1.4. Encoding of complete image

For encoding of the complete image, 3 subbands have to be encoded, whereby each subband contains $\log_2 N - 2$ levels. The total number of operations Φ for an image (excluding the last two levels, which are encoded separately, and are also excluded from the SPHIT analysis) is given as

$$\Phi = 3\sum_{L=2}^{\log_2 N-2} \phi_{L>1,b} + \phi_{L=1,b} \approx N^2\left(r + c + \frac{5}{16}cl + 4g + 4p\right). \tag{9}$$

4.2. SPIHT

The SPIHT algorithm recursively encodes base sets $\mathcal{B}_{L,b,l}(i)$ of 4 coefficients. If we assume that the image is not compressible (thus modeling the worst case in terms of computational cost) a succinct description of the algorithm for encoding of a complete wavelet subband is:

Encode$(\mathcal{B}_{L,b,l}(i))$ {
1. Calculate $q_4 = q\left(\mathcal{B}^{+}_{L,b,l}(i)\right)$
2. Encode q_4
3. if $q_4 \geqslant q_{\min}$
   (a) Encode the coefficients in $\mathcal{B}_{L,b,l}(i)$
   (b) Calculate $q_{16} = q(\mathcal{D}_{L,b,l}(i))$
   (c) Encode q_{16}
   (d) if $q_{16} \geqslant q_{\min}$ **Encode** (all sets in $\mathcal{C}_{L,b,l}(i)$)
}

The given description refers to the so-called *list of insignificant sets* (LIS) which constitutes the recursive part of SPIHT. There exist two other lists/buffers in SPIHT, the *list of insignificant pixels* (LIP) (a coefficient is called *pixel* in SPIHT) and the *list of significant pixels*

Please cite this article in press as: S. Rein, M. Reisslein, Scalable line-based wavelet image coding in wireless sensor networks, J. Vis. Commun. (2016), http://dx.doi.org/10.1016/j.jvcir.2016.07.006

Table 3
Scalable amount of image data of proposed coding approach for the refined quantization levels $q_{\min} = 8, \ldots, 0$. Each table cell gives the amount of scalable image data [in Bytes] of the proposed scalable image coder (first row), the corresponding single-run (non-scalable) amount of image data [in Bytes] of Wi2l [18], which achieves equivalent compression to SPIHT, and the corresponding PSNR image quality [in dB]. The first column $q_{\min} = 9$ gives the base stream image data amount and PSNR quality. Each successive column $q_{\min} = 8, 7, \ldots, 0$) gives the aggregate amount of image data (of the preceding $q_{\min} + 1$ non-scalable (base) stream plus the scalable refinement stream).

Image	Metric	$q_{\min}$				
		9	8	7	6	5
bridge	Data am., scal. enc. [Byte]		169	581	1849	5496
	Data am., single enc. [Byte]	51	168	580	1848	5494
	PSNR img. qu. [dB]	17.85	19.31	21.3	23.72	27.03
goldh.	Data am., scal. enc. [Byte]		116	363	1164	3527
	Data am., single enc. [Byte]	38	114	362	1162	3526
	PSNR img. qu. [dB]	19.74	21.19	23.18	25.36	28.46
lena	Data am., scal. enc. [Byte]		201	545	1379	3097
	Data am., single enc. [Byte]	77	200	544	1378	3096
	PSNR img. qu. [dB]	18.46	21.1	23.67	26.85	30.52
		4	3	2	1	0
bridge	Data am., scal. enc. [Byte]	12,436	21,615	31,386	40,903	49,947
	Data am., single enc. [Byte]	12,435	21,614	31,384	40,902	49,946
	PSNR img. qu. [dB]	31.68	37.39	42.48	45.15	46.73
goldh.	Data am., scal. enc. [Byte]	8530	16,766	26,686	36,647	45,967
	Data am., single enc. [Byte]	8529	16,764	26,685	36,645	45,966
	PSNR img. qu. [dB]	32.31	37.26	42.23	45.08	46.75
lena	Data am., scal. enc. [Byte]	6037	10,675	18,709	29,356	39,489
	Data am., single enc. [Byte]	6035	10,674	18,707	29,355	39,487
	PSNR img. qu. [dB]	34.50	38.50	42.21	44.99	46.69

(LSP). Sets of coefficients that are encoded in Step (3) (a) leave the LIS, and the respective isolated coefficients either move to the LIP or LSP. The SPIHT algorithm does not exactly describe the memory structures and buffers that are required to implement the algorithm (low memory and speed was not a design goal of SPIHT), therefore we do not discuss the respective operations here. Similarly, the operations required to compute the level $\mathcal{B}_{L,b,l}(i)$ in step (1) are not detailed (as well as the related level in Step (3) (b)); this computation would actually require to traverse the wavelet tree until the first level.

The Encode function has to be called for $\mathcal{B}_{L=\log_2 N-1,b,l=0}(i=0)$, thus recursively encoding the complete subband. Regarding the scalability, it is computationally worthwhile to mention that such a function only triggers a refinement of a single bit for each coefficient, that is, $q_{\text{minPrv}} - q_{\min} = 1$ (while the scalable Wi2l can refine the coefficients for any number of bits in one run).

Note that step (3) (d) actually performs four recursive function calls, each for one of the four sets in $\mathcal{C}_{L,b,l}(i)$. In case of significant coefficients, there is for each set $\mathcal{B}_{L,b,l}(i)$ coding of two quantization levels and four coefficients. Due to the recursion, the wavelet tree is traversed until wavelet level $L = 1$, and for each wavelet level the total number of operations is four times as high than in the previous level. For the first wavelet level, there is no encoding of $q(\mathcal{D}_{L,b,l}(i))$ levels as there are no child coefficients. The resulting number of operations ϕ_b is:

$$\phi_b = \sum_{L=1}^{\log_2 N-2} 4^{L-1}(2cl + 4r + 4c) + 4^{\log_2 N-2}(cl + 4r + 4c) \tag{10}$$

$$= (2cl + 4r + 4c)\left(-\frac{1}{3} + \frac{1}{3}\frac{N^2}{16}\right) + \frac{N^2}{16}(cl + 4r + 4c) \tag{11}$$

$$\approx \frac{1}{3}\frac{N^2}{16}(5cl + 16r + 16c) \tag{12}$$

For encoding the complete image, all three subbands have to be encoded, resulting in the total number of

$$\Phi_{SPIHT} = 3 \cdot \phi_b = N^2\left(r + c + \frac{5}{16}cl\right). \tag{13}$$

operations for encoding a complete image.

4.3. Discussion of analytical results

There are three main findings from the computational complexity analysis: (i) SPIHT and scalable Wi2l both require the same amount of coefficient read as well as coefficient and level coding operations: An image of dimension N requires $N^2(r + c + 5cl/16)$ operations; however, this holds only true for $q_{\text{minPrv}} - q_{\min} = 1$, otherwise SPIHT requires $q_{\text{minPrv}} - q_{\min}$ times more operations (than scalable Wi2l).

(ii) The SPIHT algorithm does not exactly describe the operations due to retrieval of level information, while the proposed scalable scheme gives an exact description of all required (flash) memory/(RAM) buffer operations (in fact, the description almost reflects the C-source code, alleviating the realization for the practitioner). A comparison in terms of computational cost for the buffer operations is thus not covered by the given SPHIT analysis. However, scalable Wi2l will in any case not require more buffer operations than SPIHT, because for each quantization level, SPIHT has to traverse all coefficients: Even if there exist zerotrees (insignificant coefficients detected in Step 3) of the SPIHT description), the remaining coefficients in the LIP and LSP have to be checked for possible updates. The computation of level information in Steps (1) and (3) (b) will even require to scan the coefficients multiple times (not covered by the given SPHIT analysis).

(iii) The access pattern of coefficients for SPIHT is purely recursive, while the proposed algorithm processes sets of wavelet coefficients line-wisely. Furthermore, the memory requirements in terms of level information are fixed and very low, while the lists in SPIHT require memory on the order of the total number of significant sets of coefficients.

5. Performance evaluation

In this section we verify the functionality of our proposed scalable Wi2l encoding scheme and compare the compression results to JPEG 2000 and the recent Google WebP compression. For the evaluation we use selected images with the dimension $N \times N = 256 \times 256$ pixels from the *Greyscale Set 1* available at http://links.uwaterloo.ca/. We convert the TIFF images to plain text using the Linux *convert* tool. For the wavelet transform we use the fractional wavelet filter implementation (with $wLevel = 6$ and default values) available at http://mre.faculty.asu.edu/fwf. The C programming language source code of our scalable encoder, which is based on the code for the related non-scalable wavelet image two-line coder (Wi2l) [18], is freely available from http://mre.faculty.asu.edu/Wi2l. For the comparison we used the *jasper* JPEG 2000 library and the Google WebP conversion tool. To obtain the PSNR qualities with our own system we perform a reverse wavelet transform of the refined wavelet image at the receiver side and compute the PSNR of the reconstructed and the original image.

Please cite this article in press as: S. Rein, M. Reisslein, Scalable line-based wavelet image coding in wireless sensor networks, J. Vis. Commun. (2016), http://dx.doi.org/10.1016/j.jvcir.2016.07.006

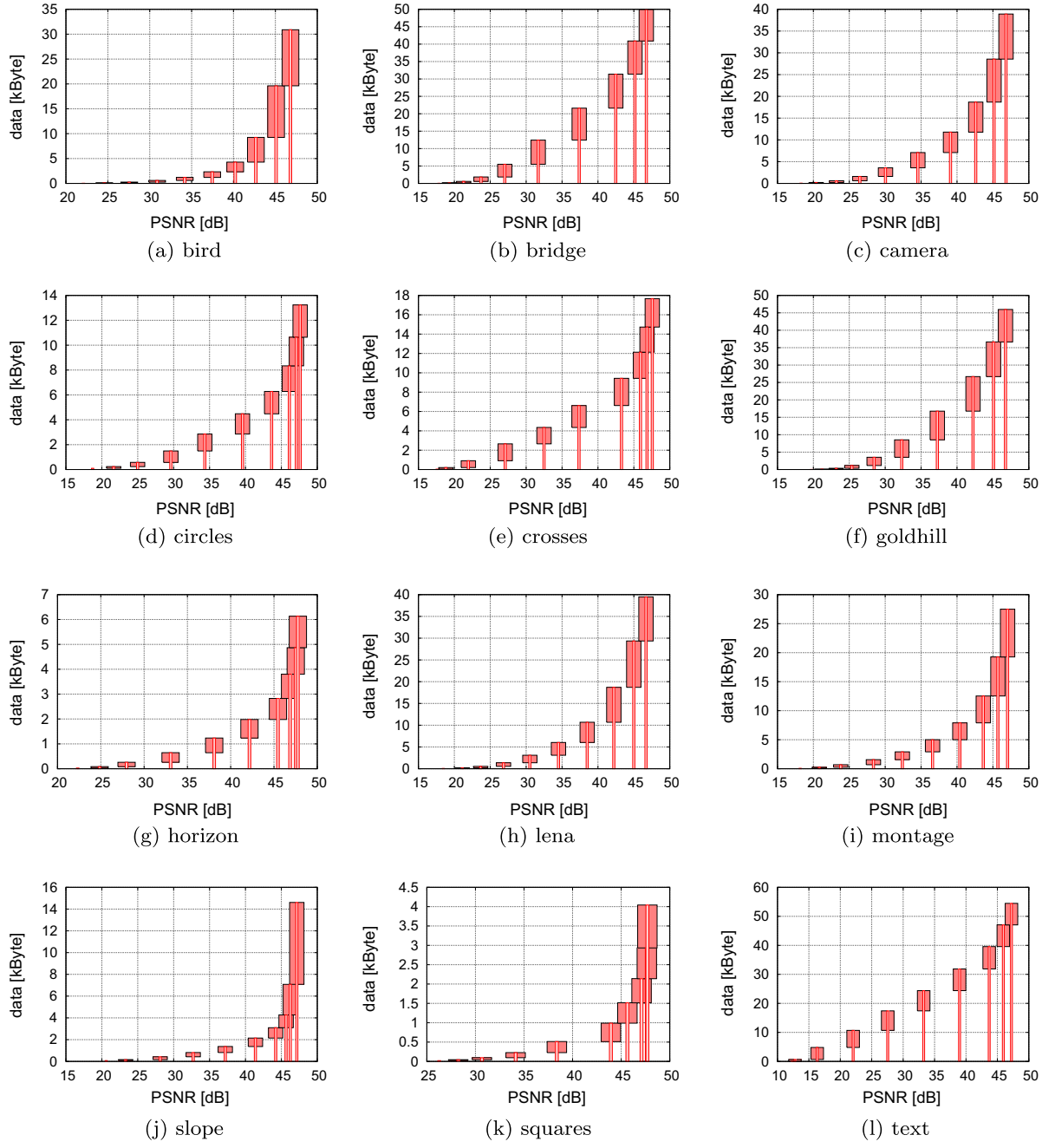

Fig. 7. Scalable amount of data (wide rectangles, in kB = 1000 B) transmitted by the proposed coding approach vs. achieved PSNR qualities for the refined quantization levels $q_{\min} = 8, \ldots, 0$. The narrow bars give the corresponding single-run (non-scalable) compression achieved by Wi2l [18], which achieves equivalent compression to SPIHT.

5.1. Scalability feature

We first evaluate the scalability feature of the presented coding algorithm. For each scalable amount of information that is transferred to the receiver in addition to the there already available base stream image, we compare the achieved compression performance with the corresponding one-run (non-scalable) compression. In particular, for three selected test images, Table 3 gives in the left-most column for $q_{\min} = 9$ the data amount (in number of Bytes) for a single-run encoding as well as the corresponding PSNR image quality in dB. Each subsequent column for $q_{\min} = 8$ down to $q_{\min} = 0$ gives the total amount of scalable image encoding data (for a base image stream $q_{\min} + 1$ and the refinement stream from $q_{\min} + 1$ to $q_{\min}$). The difference between the scalable encoding data amount in a given column $q_{\min}$ and the single encoding amount in the neighboring column to the left (for $q_{\min} + 1$) gives the amount of scalable image data that has to be transmitted for refining the image from the $q_{\min} + 1$ base image quality to the refined $q_{\min}$ quality level. For example, consider the Lena image in Table 3, with a base quality of 34.5 dB (for $q_{\min} = 4$) requiring 6035 Bytes for a single-run encoding. The refined 38.5 dB PSNR image quality (for $q_{\min} = 3$) requires a total of 10,675 Bytes of scalable image data; in particular 6035 Bytes for the base stream and 4640 Bytes for the scalable refinement stream. Notice that the corresponding $q_{\min} = 3$ non-scalable encoding with the Wi2l coder [18] achieves exactly the same 38.5 dB PSNR image quality and requires 10,674 Bytes. Generally, we observe from Table 3 that the scalable encoding requires only one or two Bytes more data than the

Please cite this article in press as: S. Rein, M. Reisslein, Scalable line-based wavelet image coding in wireless sensor networks, J. Vis. Commun. (2016), http://dx.doi.org/10.1016/j.jvcir.2016.07.006

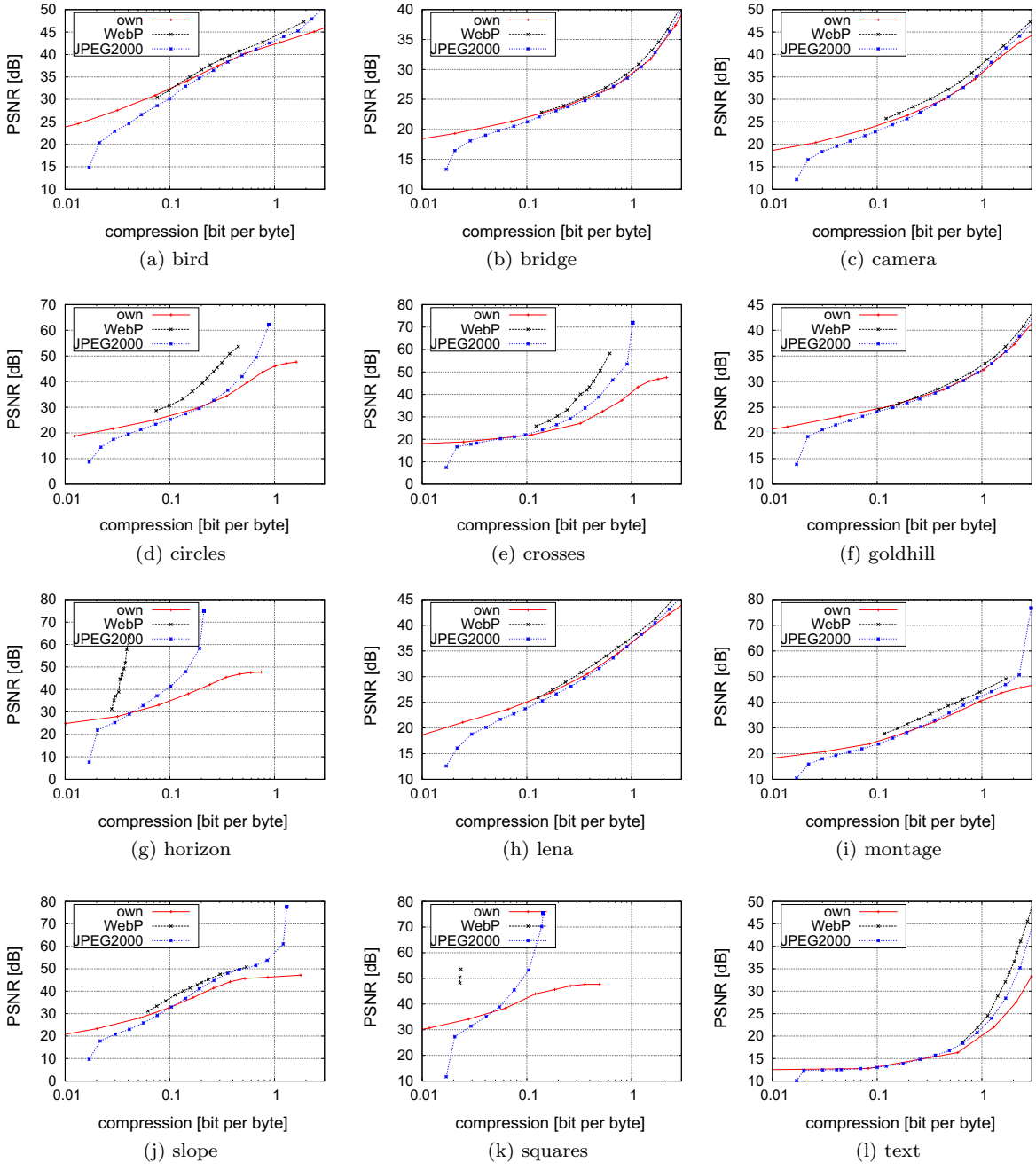

Fig. 8. Compression comparison for the scalable system (*own*) with JPEG 2000 and Google's WebP image format: PSNR image quality [dB] as a function of compression ratio [bits of encoded image data per byte of raw image data]. For the PSNR image qualities below 40 dB, the image quality of our approach is higher or nearly the same compared to JPEG 2000 and WebP.

corresponding non-scalable encoding. These one or two additional Bytes are due to sending the image dimension in our implementation of the scalable algorithm, whereas the non-scalable implementation does not send the dimension (and assumes a 256×256 pixel image dimension). However, these one or two additional Bytes are negligible compared to the encoded image data amounts, which are typically hundreds of Bytes.

Fig. 7 gives the amount of scalable (refinement stream) data (wide rectangular boxes) and the respective achieved image PSNR qualities for a wide set of twelve test images. The amount of data to achieve the same PSNR qualities with only one run (without scalability) using the Wi2l coder [18], which gives the same compression as SPIHT, is given by the narrow bars in the plot. For instance, we observe from Fig. 7(h) that for the Lena image, the wide rectangular box at PSNR image quality 38.5 dB, corresponds to a refinement of the image from quantization level $q_{\min} = 4$ to $q_{\min} = 3$ and contains 4640 Bytes of scalable data. The size of the corresponding base stream is 6035 Bytes. The Wi2l coder needs 10,674 Bytes to achieve the same image quality of 38.5 dB in one run. We observe from Fig. 7 that the narrow bars always end at the top of the wide boxes, indicating that the scalability does not introduce additional cost (except for the one or two additional Bytes that have been identified in Table 3).

We observe from the graphical representation of the scalable encoding results in Fig. 7 that the amount of scalable data in general increases with higher desired PSNR image quality. For

Please cite this article in press as: S. Rein, M. Reisslein, Scalable line-based wavelet image coding in wireless sensor networks, J. Vis. Commun. (2016), http://dx.doi.org/10.1016/j.jvcir.2016.07.006

instance, we observe for the bird test image from Fig. 7(a) that in the low-quality range, a few hundred Bytes of scalable data improve the PSNR image quality in steps of approximately 3 dB. However, for the quality improvement from 34.2 dB to 37.5 dB approximately 1100 Bytes of scalable data are needed; then 2000 more Bytes to achieve 40.2 dB, and then approximately 5000 more Bytes to achieve 42.7 dB.

We also observe from Fig. 7 that the amount of scalable data (height of wide box) equals exactly (neglecting the one or two extra Bytes) the difference of two consecutive narrow bars. This observation holds for all bars (quality improvements) and test images. Thus, a main result from the extensive evaluation in Fig. 7 is that the proposed line-based low-memory encoder achieves scalability of the image quality across a wide set of quality ranges and image contents without incurring any significant overhead compared to the non-scalable line-based low-memory coder [18], which attains exactly the rate-distortion performance of the SPIHT coding algorithm.

5.2. Comparison with JPEG 2000 and WebP

We next compare the rate-distortion (RD) compression performance of the scalable coder with JPEG 2000 and WebP. Fig. 8 gives the compression performance in bits per byte (bpb) for the levels $q_{\min} = 8, \ldots, 2$, whereby our proposed encoder is designated by `own` in the plots. We observe that for low to moderate PSNR image qualities, the three coding approaches give very similar compression performance, whereby our proposed line-based approach gives slightly better performance than JPEG 2000 and WebP for low PSNR values. For instance, we observe from Fig. 8(h) specifically for the Lena image that all three encoding approaches give similar compression performance for PSNR values below 40 dB. For PSNR values below 27 dB, our proposed line-based approach gives higher PSNR values than JPEG 2000 and WebP.

On the other hand, for high PSNR image qualities, we observe the opposite behavior, i.e., our line-based approach gives worse compression performance than JPEG 2000 and WebP. For instance, for the Lena image (Fig. 8(h)), the proposed line-based approach is not competitive for PSNR values above 40 dB. In that high quality range, however, quality improvements are barely, if at all, visible.

We observe from Fig. 8 that WebP achieves generally competitive compression performance. For instance, for the goldhill image, see Fig. 8(f), WebP achieves a PSNR of 36.8 dB with a compression of 1.68 bpb, while JPEG 2000 achieves 35.9 dB with 1.69 bpb, and our line-based approach achieves 37.3 dB for 2.05 bpb. However, WebP compression provides only very coarse granularity, giving a base image quality of 24.6 dB. The wavelet based systems start to provide base images earlier; specifically, the proposed system delivers a base quality of 19.7 dB for the first 38 Bytes (not visible in the plot) and 21.2 dB for the first 114 Bytes. The achieved compression of our approach is the same as with the SPIHT algorithm. We also observe from Fig. 8 that WebP achieves very good compression performance for artificial images; this characteristic is most pronounced for the squares and horizon test images (Fig. 8 (g) and (k)), which consist of square and horizontal line drawings. For natural test images, which are more typical for image sensor network applications, our proposed gives competitive compression performance compared to WebP, especially in the lower image quality range.

6. Conclusion

We have proposed a line-based approach for quality-scalable wavelet image compression on low-memory sensor nodes. Common existing scalable image coding approaches are too complex to be applied in sensor networks. Our key contribution in this paper is to make scalability applicable to sensor nodes with very little RAM and line-based external memory (e.g., in the form of flash memory). Our proposed approach builds on established techniques, namely on the wavelet transform as well as on the wavelet image two line coder [18] and the backward coding principle [41], which are based on the tree-based coding principle [48].

Our line-based encoding approach has extremely low complexity, yet achieves compression performance competitive to the state-of-the art approaches, especially for high compression rates. Our extensive evaluations demonstrated the proper scalable compression functionality of the proposed line-based approach. Our method performs a re-ordering of the binary data and thus does not introduce any significant overhead compared to non-scalable wavelet image coding. The entropy of our line-based image coding approach is thus the same as for SPIHT, which has been compared to JPEG 2000 in [1]. We found that our line-based approach achieves similar compression as JPEG 2000 and WebP in the high compression ratio (low image quality) region, which is important for sensor networks. In the low compression rate region, our approach gives worse compression compared to JPEG 2000 and WebP. Our wavelet based approach achieves fine granular scalable image coding, providing a base image quality for very small base image data sets on the order of tens of bytes, whereas WebP gives only coarse granular scalability.

An important direction for future research on low-memory scalable image compression for sensor nodes is to explore combinations of the computation of the transform and the encoding of the wavelet transform coefficients. To date, research has examined low-memory approaches for either the wavelet transform, or the scalable encoding of the wavelet transform coefficients. Through merging the wavelet transform and encoding stages, it may be possible to exploit synergies between the two stages that reduce the overall memory required for image compression. Another important research direction is to examine the energy consumption of low-memory image compression on a variety of visual sensor node platforms. Such future energy consumption studies could build on and extend existing energy consumption characterization techniques, e.g., [49–51], to capture the specific image coding related energy expenditures.

References

[1] D. Taubman, M. Marcellin, JPEG2000—Image Compression, Fundamentals, Standard and Practice, Kluwer Academic Publishers, 2004.
[2] C.-H. Hung, H.-M. Hang, A reduced-complexity image coding scheme using decision-directed wavelet-based contourlet transform, J. Vis. Commun. Image Represent. 23 (7) (2012) 1128–1143.
[3] B. Tavli, K. Bicakci, R. Zilan, J.M. Barcelo-Ordinas, A survey of visual sensor network platforms, Multimed. Tools Appl. 60 (3) (2012) 689–726.
[4] A. Seema, M. Reisslein, Towards efficient wireless video sensor networks: a survey of existing node architectures and proposal for a Flexi-WVSNP design, IEEE Commun. Surv. Tutorials 13 (3) (2011) 462–486.
[5] F. Chen, D.A. Koufaty, X. Zhang, Understanding intrinsic characteristics and system implications of flash memory based solid state drives, ACM SIGMETRICS Perform. Eval. Rev. 37 (1) (2009) 181–192.
[6] L.M. Grupp, A.M. Caulfield, J. Coburn, S. Swanson, E. Yaakobi, P.H. Siegel, J.K. Wolf, Characterizing flash memory: anomalies, observations, and applications, in: Proc. IEEE/ACM Int. Symp. on Microarchitecture (MICRO), 2009, pp. 24–33.
[7] I.F. Akyildiz, T. Melodia, K.R. Chowdhury, Wireless multimedia sensor networks: applications and testbeds, Proc. IEEE 96 (10) (2008) 1588–1605.
[8] D. Costa, L.A. Guedes, F. Vasques, P. Portugal, Energy-efficient packet relaying in wireless image sensor networks exploiting the sensing relevancies of source nodes and DWT coding, J. Sens. Actuat. Netw. 2 (3) (2013) 424–448.
[9] B. Rinner, W. Wolf, Toward pervasive smart camera networks, in: H. Aghajan, A. Cavallaro (Eds.), Multi-Camera Networks: Principles and Applications, Academic Press, 2009, pp. 483–496.
[10] B. Song, C. Ding, A. Kamal, J. Farrell, A. Roy-Chowdhury, Distributed camera networks, IEEE Signal Proc. Mag. 28 (3) (2011) 20–31.
[11] M. Imran, K. Khursheed, N. Ahmad, A.W. Malik, M. O'Nils, N. Lawal, Complexity analysis of vision functions for implementation of wireless smart cameras using system taxonomy, Proc. of SPIE, vol. 8437, 2012. pp. 84370C-1-84370C-20.

Please cite this article in press as: S. Rein, M. Reisslein, Scalable line-based wavelet image coding in wireless sensor networks, J. Vis. Commun. (2016), http://dx.doi.org/10.1016/j.jvcir.2016.07.006

[12] T. Ma, M. Hempel, D. Peng, H. Sharif, A survey of energy-efficient compression and communication techniques for multimedia in resource constrained systems, IEEE Commun. Surv. Tutorials 15 (3) (2013) 963–972.
[13] P.N. Huu, V. Tran-Quang, T. Miyoshi, Low-complexity and energy-efficient algorithms on image compression for wireless sensor networks, IEICE Trans. Commun. 93 (12) (2010) 3438–3447.
[14] T. Ma, P. Shrestha, M. Hempel, D. Peng, H. Sharif, Low-complexity image coder/decoder with an approaching-entropy quad-tree search code for embedded computing platforms, in: Proc. IEEE ICIP, 2011, pp. 297–300.
[15] A. Mammeri, B. Hadjou, A. Khoumsi, A survey of image compression algorithms for visual sensor networks, ISRN Sens. Netw. 2012–760320 (2012) 1–19.
[16] G. Mathur, P. Desnoyers, P. Chukiu, D. Ganesan, P. Shenoy, Ultra-low power data storage for sensor networks, ACM Trans. Sens. Netw. 5 (4) (2009) 1–34.
[17] S. Rein, S. Lehmann, C. Gühmann, Wavelet image two-line coder for wireless sensor node with extremely little RAM, in: Proc. of the IEEE Data Compression Conference (DCC'09), Snowbird UT, 2009, pp. 252–261.
[18] S.A. Rein, F.H. Fitzek, C. Gühmann, T. Sikora, Evaluation of the wavelet image two-line coder: a low complexity scheme for image compression, Signal Process.: Image Commun. 37 (2015) 58–74.
[19] H. Arora, P. Singh, E. Khan, F. Ghani, Memory efficient set partitioning in hierarchical tree (mesh) for wavelet image compression, in: Proc. IEEE ICASSP, 2005, pp. 385–388.
[20] N.R. Kidwai, E. Khan, R. Beg, A memory efficient listless SPECK (MLSK) image compression algorithm for low memory applications, Int. J. Adv. Res. Comput. Sci. 3 (4) (2012) 209–215.
[21] M.V. Latte, N.H. Ayachit, D. Deshpande, Reduced memory listless speck image compression, Digit. Signal Process. 16 (6) (2006) 817–824.
[22] M. Loomans, C. Koeleman, P. de With, Low-complexity wavelet-based scalable image video coding for home-use surveillance, IEEE Trans. Consum. Electron. 57 (2) (2011) 507–515.
[23] M. Sakalli, W. Pearlman, M. Farshchian, SPIHT algorithms using depth first search algorithm with minimum memory usage, in: Proc. Conf. on Information Sciences and Systems, 2006, pp. 1158–1163.
[24] C.-Y. Su, B.-F. Wu, A low memory zerotree coding for arbitrarily shaped objects, IEEE Trans. Image Proc. 12 (3) (2003) 271–282.
[25] R.K. Bhattar, K.R. Ramakrishnan, K.S. Dasgupta, Strip based coding for large images using wavelets, Signal Process.: Image Commun. 17 (6) (2002) 441–456.
[26] W.C. Chia, L.W. Chew, L. Ang, K.P. Seng, Low memory image stitching and compression for WMSN using strip-based processing, Int. J. Sens. Netw. 11 (1) (2012) 22–32.
[27] L. Chew, L.-M. Ang, K. Seng, New virtual SPIHT tree structures for very low memory strip-based image compression, IEEE Signal Proc. Lett. 15 (2008) 389–392.
[28] Y. Bao, C. Kuo, Design of wavelet-based image codec in memory-constrained environment, IEEE Trans. Circ. Syst. Video Technol. 11 (5) (2001) 642–650.
[29] C.-K. Hu, W.-M. Yan, K.-L. Chung, Efficient cache-based spatial combinative lifting algorithm for wavelet transform, Signal Process. 84 (9) (2004) 1689–1699.
[30] V. Ratnakar, TROBIC: two-row buffer image compression, in: Proc. of the IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP), 1999, pp. 3133–3136.
[31] C.-H. Yang, J.-C. Wang, J.-F. Wang, C.-W. Chang, A block-based architecture for lifting scheme discrete wavelet transform, IEICE Trans. Fundam. E90-A (5) (2007) 1062–1071.
[32] Q. Lu, W. Luo, J. Wang, B. Chen, Low-complexity and energy efficient image compression scheme for wireless sensor networks, Comput. Netw. 52 (13) (2008) 2594–2603.
[33] C. Chrysafis, A. Ortega, Line-based, reduced memory, wavelet image compression, IEEE Trans. Image Process. 9 (3) (2000) 378–389.
[34] J. Oliver, M. Malumbres, On the design of fast wavelet transform algorithms with low memory requirements, IEEE Trans. Circ. Syst. Video Technol. 18 (2) (2008) 237–248.
[35] X. Zhang, L. Cheng, H. Lu, Low memory implementation of generic hierarchical transforms for parent children tree (PCT) production and its application in image compression, Signal Process.: Image Commun. 24 (5) (2009) 384–396.
[36] M.H. Eslami, M.Y. Tafti, ELHWT: efficient lowest to highest wavelet tree for image processing and compression, in: Proc. Int. MultiConference of Engineers and Computer Scientists, 2008, pp. 1–4.
[37] J. Guo, S. Mitra, B. Nutter, T. Karp, A fast and low complexity image codec based on backward coding of wavelet trees, in: Proc. of IEEE DCC, 2006, pp. 292–301.
[38] L. Ye, J. Guo, B. Nutter, S. Mitra, Memory-efficient image codec using line-based backward coding of wavelet trees, in: Proc. IEEE DCC, 2007, pp. 213–222.
[39] L. Ye, J. Guo, B. Nutter, S. Mitra, Low-memory-usage image coding with line-based wavelet transform, SPIE J. Opt. Eng. 50 (2) (2011) 027005-1–027005-11.
[40] M. Tausif, N. Kidwai, E. Khan, M. Reisslein, FrWF-based LMBTC: memory-efficient image coding for visual sensors, IEEE Sens. J. 15 (11) (2015) 6218–6228.
[41] J. Guo, S. Mitra, B. Nutter, T. Karp, Backward coding of wavelet trees with fine-grained bitrate control, J. Comput. 1 (4) (2006) 1–7.
[42] O. López, M. Martinez-Rach, J. Oliver, M. Malumbres, Impact of rate control tools on very fast non-embedded wavelet image encoders, in: Proc. SPIE Electronic Imaging, 2007, pp. 650829-1–650829-9.
[43] O.M.L. Granado, M.O. Martínez-Rach, P.P. Peral, J.O. Gil, M.P. Malumbres, Rate control algorithms for non-embedded wavelet-based image coding, J. Signal Process. Syst. 68 (2) (2012) 203–216.
[44] J. Guo, S. Mitra, T. Karp, B. Nutter, A resolution-and rate-scalable image subband coding scheme with backward coding of wavelet trees, in: Proc. APCCAS, 2006, pp. 442–445.
[45] S. Rein, S. Lehmann, C. Gühmann, Fractional wavelet filter for camera sensor node with external flash and extremely little RAM, in: Proc. of the ACM Mobile Multimedia Communications Conference (Mobimedia), 2008, pp. 1–7.
[46] S. Rein, M. Reisslein, Performance evaluation of the fractional wavelet filter: a low-memory image wavelet transform for multimedia sensor networks, Ad Hoc Netw. 9 (4) (2011) 482–496.
[47] S. Rein, M. Reisslein, Low-memory wavelet transforms for wireless sensor networks: a tutorial, IEEE Commun. Surv. Tutorials 13 (2) (2011) 291–307.
[48] A. Said, W. Pearlman, A new, fast, and efficient image codec based on set partitioning in hierarchical trees, IEEE Trans. Circ. Syst. Video Technol. 6 (3) (1996) 243–250.
[49] C.B. Margi, K. Obraczka, R. Manduchi, Characterizing system level energy consumption in mobile computing platforms, Proc. IEEE Int. Conf. on Wireless Networks, Communications and Mobile Computing, vol. 2, 2005, pp. 1142–1147.
[50] C.B. Margi, V. Petkov, K. Obraczka, R. Manduchi, Characterizing energy consumption in a visual sensor network testbed, in: Proc. IEEE Int. Conf. on Testbeds and Research Infrastructures for the Development of Networks and Communities (TRIDENTCOM), 2006, pp. 1–8.
[51] V. Shnayder, M. Hempstead, B.-R. Chen, G.W. Allen, M. Welsh, Simulating the power consumption of large-scale sensor network applications, in: Proc. ACM Int. Conf. on Embedded Networked Sensor Systems, 2004, pp. 188–200.

Please cite this article in press as: S. Rein, M. Reisslein, Scalable line-based wavelet image coding in wireless sensor networks, J. Vis. Commun. (2016), http://dx.doi.org/10.1016/j.jvcir.2016.07.006