Packed Integer Wavelet Transform Constructed by Lifting Scheme

Chengjiang Lin, Bo Zhang, and Yuan F. Zheng

***Abstract*—A new method for speeding up the integer wavelet transforms constructed by the lifting scheme is proposed. The proposed method packs multiple pixels (wavelet coefficients) in a single word; therefore, it can make use of the 32-bit or 64-bit computational capability of modern computers to accomplish multiple addition/subtraction operations in one instruction cycle. As a result, our method can save the decomposition/reconstruction time by up to 37% on 32-bit machines and require much less working memory in comparison with the original wavelet transform algorithms.**

***Index Terms*—Biorthogonal wavelets, image compression, integer wavelet ransform, lifting scheme, packed computation.**

I. Introduction

THE WAVELET transform has received much attention in the field of image compression [1]–[3]. It provides great potential of achieving better rate-distortion performance than conventional DCT-based approach (e.g., JPEG). One problem associated with the wavelet image compression technology is the high computational complexity. Although floating-point arithmetic is nearly as fast as integer arithmetic when their operands have the same data length, the integer wavelet transform can be implemented much faster than the floating point wavelet transform in almost all general purpose computers because the floating point wavelet transform demands for longer data length than the integer wavelet transform does. Another benefit of using integer wavelets is the reversibility. That is, the image can be reconstructed losslessly because all the coefficients are integers and can be stored without rounding-off errors.

The lifting scheme is a new method for constructing integer wavelet transform [4]. Recently, biorthogonal wavelets constructed by the lifting scheme have been identified as very promising filters for lossless/lossy image compression applications [3], [5]. By making use of similarities between the high- and low-pass filters, the lifting scheme reduces the computation complexity by a factor of two compared with traditional wavelet transform algorithms. With certain modifications, the corresponding wavelet transform can even be calculated with only integer addition and shift operations which make the computation even faster [3]. Besides, the transform is reversible which means that it can be used for both lossless and lossy image compression. Furthermore, the inverse wavelet transform can be immediately found by undoing the operations of the forward transform.

Manuscript received August 27, 1998; revised June 12, 2000. This paper was recommended by Associate Editor C. W. Chen.

The authors are with the Department of Electrical Engineering, The Ohio State University, Columbus, OH 43210 USA (e-mail: zheng@ee.eng.ohio-state.edu).

Publisher Item Identifier S 1051-8215(00)10625-1.

Modern wavelet-based image compression systems [1], [2] contain three building elements: 1) wavelet transform; 2) successive quantization; and 3) adaptive entropy coding. Typically, more than 60% of the time used in image compression is consumed by the wavelet transform. It is very crucial to speed up the computation of the wavelet transform for real-time image and video-compression applications, especially for large-scale and color images. While integer wavelets using the lifting scheme significantly reduce the computation time [5], we propose a new approach to further speed up the computation of the wavelet transform.

The method is based on the fact that the 16-bit integer arithmetic has the same speed as the 32-bit integer arithmetic in contemporary computers while a 16-bit data unit is sufficient for most integer wavelets. We can therefore pack multiple pixels (wavelet coefficients) in a single long word during the computation of the reversible wavelet transform. As a result, operations on multiple pixels (wavelet coefficients) can be performed at once. Thus, the computation time as well as the working memory space can be dramatically reduced. Furthermore, we observed that for reversible integer wavelets constructed by the lifting scheme, if the dynamic range of the coefficients is within $[-2^{15}, 2^{15}-1]$, their corresponding packed version is also a reversible transform. Consequently, the quality of the reconstructed images is the same as an unpacked transform method. Performing two logical arithmetic operations with one physical operation was proposed earlier for DCT-based JPEG compression and decompression [6]. This paper uses the same approach, but applies it to the integer wavelet transform that needs different considerations in the design of multiple arithmetic operations and analysis of errors.

This paper is organized as follows: Section II describes the basic idea of the wavelet transform and the integer wavelet transform based on the lifting scheme. Section III introduces the packed computation and gives the packed integer wavelet transform algorithm. Section IV presents the experimental results of the new method for grayscale and color image compression. Conclusions are offered in Section V.

II. Integer Wavelet Transform

The wavelet transform can be considered as a subband transform and implemented with a filter bank [7]. Fig. 1 describes the general block scheme of a one-dimensional biorthogonal wavelet transform. The forward transform uses two analysis filters $\tilde{h}$ (low pass) and $\tilde{g}$ (high pass) followed by subsampling, while the inverse transform first upsamples and then uses two synthesis filters h (low pass) and g (high pass). The outputs of

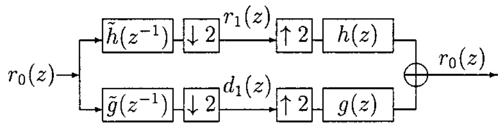

Fig. 1. Basic filter bank for biorthogonal wavelet transform.

the synthesis filters are added together to reconstruct the original signal. The conditions for perfect reconstruction are given by [8]

$$\begin{cases} h(z)\tilde{h}(z^{-1}) + g(z)\tilde{g}(z^{-1}) = 2 \\ h(z)\tilde{h}(-z^{-1}) + g(z)\tilde{g}(-z^{-1}) = 0. \end{cases}$$

When $\tilde{h} = h$ and $\tilde{g} = g$, $\{\tilde{h}, \tilde{g}, h, g\}$ forms an orthogonal wavelet transform. By using the polyphase representation of a filter h: $h(z) = h_e(z^2) + z^{-1}h_o(z^2)$, where $h_e(z) = \sum_k h_{2k}z^{-k}$ contains the even coefficients, and $h_o(z) = \sum_k h_{2k+1}z^{-k}$ contains the odd coefficients, we can assemble the *polyphase matrix* $P(z)$ [5] to represent the filter pair $(\tilde{h}, \tilde{g})$

$$P(z) = \begin{bmatrix} \tilde{h}_e(z) & \tilde{h}_o(z) \\ \tilde{g}_e(z) & \tilde{g}_o(z) \end{bmatrix}.$$

Let $x(z) = r_0(z)$. The corresponding wavelet transform and subsampling in Fig. 1 can be written as

$$\begin{bmatrix} r_1(z) \\ d_1(z) \end{bmatrix} = P(z) \begin{bmatrix} x_e(z) \\ z^{-1}x_o(z) \end{bmatrix}$$

where $x_e(z)$ and $x_o(z)$ are the even and odd components of $x(z)$.

The lifting scheme [4], [5] provides a new approach for construction of biorthogonal wavelets with compact support. Daubechies and Sweldens [5] proved that any polyphase matrix $P(z)$ representing a wavelet transform with finite filters can be obtained by performing a Lazy wavelet transform followed by alternating *primal* and/or *dual lifting* steps.

With *primal lifting,* starting from two complementary finite filters $\tilde{h}$ and $\tilde{g}$, a new finite filter $\tilde{h}^{\text{new}}$ complementary to $\tilde{g}$ is created

$$\tilde{h}^{\text{new}}(z) = \tilde{h}(z) + s(z^2)\tilde{g}(z) \tag{1}$$

where $s(z)$ is a Laurent polynomial. Using the polyphase representation, (1) can be written as

$$P^{\text{new}}(z) = \begin{bmatrix} 1 & s(z) \\ 0 & 1 \end{bmatrix} P(z).$$

With *dual lifting,* starting from $\tilde{h}$ and $\tilde{g}$, a new finite filter $\tilde{g}^{\text{new}}$ complementary to $\tilde{h}$ is created

$$\tilde{g}^{\text{new}}(z) = \tilde{g}(z) + t(z^2)\tilde{h}(z) \tag{2}$$

where $t(z)$ is a Laurent polynomial. Using the polyphase representation, (2) can be written as

$$P^{\text{new}}(z) = \begin{bmatrix} 1 & 0 \\ t(z) & 1 \end{bmatrix} P(z).$$

By iteratively repeating this process, $P(z)$ can be factored into a product of unit upper and lower triangular 2×2 matrices, and a diagonal normalization matrix [5]. Alternatively, one can obtain any wavelet transform $P(z)$ by starting with a simple wavelet transform represented by the diagonal normalization matrix and using the lifting scheme. One of the most interesting advantages of the lifting scheme is that it can be used to create integer wavelet transforms.

It has been proven that if the decomposition and reconstruction by filters $\{\tilde{h}, \tilde{g}, h, g\}$ can be accomplished with only integer arithmetic, we can also create a corresponding integer wavelet transform by filters $\{\tilde{h}^{\text{new}}, \tilde{g}, h, g^{\text{new}}\}$ which are generated based on $\{\tilde{h}, \tilde{g}, h, g\}$ and $\{s, t\}$. In fact one can, in each lifting step, round off the result of the filter right before the addition or subtraction. Thus, the forward and inverse integer wavelet transforms are as follows.

1) Forward transform:
   a) Classical subband splitting by applying the analysis filters $(\tilde{h}, \tilde{g})$ to $r_0(n)$, the corresponding low-pass and high-pass subbands are $r_1^o(k)$ and $d_1^o(k)$, respectively.
   b) Updating the low subband $r_1^o(k)$ by applying the s filter on the high subband $d_1(k) = d_1^o(k)$

$$r_1(k) = r_1^o(k) + \text{Int}\left(\sum_n d_1(k-n)s(n)\right) \tag{3}$$

   or updating the high subband $d_1^o(k)$ by applying the t filter on the low subband $r_1(k) = r_1^o(k)$

$$d_1(k) = d_1^o(k) + \text{Int}\left(\sum_n r_1(k-n)t(n)\right). \tag{4}$$

2) Inverse transform:
   a) Undoing the primal lifting with $d_1^o(k) = d_1(k)$, or undo the dual lifting with $r_1^o(k) = r_1(k)$. This is exactly the "backward" version of (3) and (4). In practice, this comes down to simply changing each $+$ into a $-$ and vice versa.
   b) Inverse transforming using the synthesis filters (h, g) on the low-pass and high-pass subbands $r_1^o(k)$ and $d_1^o(k)$ to get back the original signal $r_0(n)$.

Notice that if we carefully choose the parameters $\{s, t\}$ in (3) and (4) so that only integer addition, subtraction, and shift operations are required in the computation, the wavelet transform can be performed directly by integer arithmetic [3]. For example, for one of the biorthogonal filter banks for image compression, the (2, 6) wavelet that corresponds to the TS transform [2], [9]

$$\begin{cases} \tilde{h} = \dfrac{1}{\sqrt{2}}(1+z) \\ \tilde{g} = \dfrac{1}{8\sqrt{2}}(-z^{-2} - z^{-1} + 8 - 8z + z^2 + z^3) \end{cases}$$

the decomposition can be done via the following lifting steps [3]:

$$\begin{cases} d_1^o(k) &= r_0(2k) - r_0(2k+1), \\ r_1(k) &= \mathrm{Int}\left(\dfrac{d_1^o(k)}{2}\right) + r_0(2k+1) \\ d_1(k) &= \mathrm{Int}\left(\dfrac{r_1(k-1) - r_1(k+1)}{4}\right) - d_1^o(k). \end{cases}$$

Its reconstruction algorithm immediately follows as we undo the decomposition operations. Based on the work of [3] and [5], several other integer wavelet transforms, such as S transform, (5,3) transform, and $(S+P)$ transform, etc. can be modified and only need integer addition, subtraction and shift operations. This property gives rise to the possibility of our proposed packed reversible integer wavelet transform, which takes advantage of the 32- or 64-bit computational capability of modern computers.

III. Packed Integer Wavelet Transform

The basic idea of packed integer wavelet transform is to pack multiple pixels (wavelet coefficients) in one integer word; therefore, multiple additions/subtractions can be accomplished in one instruction cycle. There are two issues associated with the packed approach. One is overflow of the magnitude, and the other is carry over of the sign bit. When either occurs, the result of the transform will be affected. It can be shown that overflow is not a concern in most applications of the wavelet transform.

The packed addition and subtraction will not overflow if the wavelet transform is limited to a few levels of the multi-band decomposition. This is because in the wavelet transform, the low-pass filters $h(n)$ and $\tilde{h}(n)$ must satisfy [10]

$$\sum_n h(n) = \sum_n \tilde{h}(n) = \sqrt{2}. \tag{5}$$

As a result, the magnitude of the coefficient is increased by $\sqrt{2}$ for every level of decomposition. For the 2-D case, the increase will be 2. This will enlarge the range of the coefficient by 1 bit in each level of decomposition. If the decomposition is limited to three to four levels, an additional 3–4 bits will be sufficient to hold the dynamic range of the coefficients. The above is based on a condition that every $h(n)$ or $\tilde{h}(n)$ is positive. If the coefficients are not all positive, one may have $\sum_n |h(n)|$ or $\sum_n |\tilde{h}(n)|$ be greater than $\sqrt{2}$. And the dynamic range may be increased by more than 1 bit in each level of decomposition. In reality, however, it is still not much greater than $\sqrt{2}$. For example, in the S, TS, (5,3), and $(S+P)$ transforms, which we have implemented using the integer computation, the limit is $2.5/\sqrt{2}$. Consequently, four levels of decomposition will introduce at most 7 bits in the worst case. If the pixels are assigned with 8 bits in the first place, the dynamic range will not exceed 16 bits.

In the integer computation, the magnitude does not even increase (see the lifting steps shown in the previous section). Reference [3] also confirms this property and calls it the *Precision Preservation* property. Consequently, overflow of the magnitude is not a concern in our applications. On the other hand, the most significant bit of a binary number in the computation is the sign bit; when multiple numbers packed in a long word, the carry over of the sign bit from the low word to the high word will alternate the magnitude of the high word. This issue needs a careful consideration and will be discussed in the following sub-section.

A. Packed Computation

Let $A = (a_{n-1}, \ldots, a_1, a_0)$ and $B = (b_{n-1}, \ldots, b_1, b_0)$ be the packed integer of $a_i, b_i \in [-2^{15}, 2^{15}-1]$ $(i = 0, \ldots, n-1)$. Notice that for 32-bit computers, $n = 2$; for 64-bit computers, $n = 4$, etc. The packed addition and subtraction of a_i and b_i are denoted as $S = A + B$ and $D = A - B$, respectively, where $S = (s_{n-1}, \ldots, s_1, s_0)$, $D = (d_{n-1}, \ldots, d_1, d_0)$, $s_i, d_i \in Z$. We observe that even when $s_{i-1}, d_{i-1} \in [-2^{15}, 2^{15}-1]$, it is not necessary that $a_i + b_i = s_i$ or $a_i - b_i = d_i$. For example, let $A = (4, -2)$ and $B = (5, 4)$. We have $S = (10, 2)$ instead of (9, 2). Although $s_0 = 2$ does not overflow, there is a carry bit generated by the low word addition that is propagated to the high word. As a result, $s_1 = 10 \neq 4 + 5$. Similar situations exist for the packed subtraction. For $A = (111, 72)$ and $B = (108, 82)$, we have $D = (2, -10)$, where $d_1 = 2 \neq 111 - 108$.

Although the carry over of the sign bit alters the result of the packed addition and subtraction, the effect is very small (limited to ± 1 for each level of wavelet transform). Consequently, we have the following possibilities for s_i and d_i $(0 \le i \le n-1)$:

$$s_i = \begin{cases} a_i + b_i, & \text{if } s_{i-1}a_{i-1}b_{i-1} \ge 0 \text{ or } i = 0 \\ a_i + b_i + 1, & \text{if } s_{i-1}a_{i-1}b_{i-1} < 0 \end{cases} \tag{6}$$

$$d_i = \begin{cases} a_i - b_i, & \text{if } d_{i-1}a_{i-1}b_{i-1} \ge 0 \text{ or } i = 0 \\ a_i - b_i - 1, & \text{if } d_{i-1}a_{i-1}b_{i-1} < 0. \end{cases} \tag{7}$$

In addition to the packed addition and subtraction operations, we further introduce the packed shift operation to avoid the interference on a_{i-1} from a_i when we apply a right shift to the packed integer A (i.e., divided by 2^l where l is the number of shifts). The functionality of the packed shift operation is equivalent to that of the following operations: $\{Int(a_i/2^l)\}$, $i = 0, \ldots, n-1$.

B. Packed Integer Wavelet Transform

In general, multiple coefficients can be packed into one integer provided that the width of the data-path is sufficient. Since 32-bit is a typical data width for the state-of-the-art computers, we will focus on packing two pixels/coefficients into one 32-bit integer in our implementation. This does not mean that our algorithms are limited to 32-bit machines. The general algorithm for the packed integer wavelet decomposition can be described as follows.

1) Pack two rows/columns

   Let $r_0(2n, k)$ and $r_0(2n+1, k)$ be the $(2n)$th and $(2n+1)$th row or column, respectively. Suppose the packed result is saved in $\overline{r_0(k)}$, we have

$$\overline{r_0(k)} = (r_0(2n, k), r_0(2n+1, k)).$$

2) Compute the packed wavelet transform using the simple filters $\tilde{h}$ and $\tilde{g}$. The algorithms are the same as (3), (4) except that we use $\overline{r_0(n)}$ as the input instead of $r_0(n)$. The

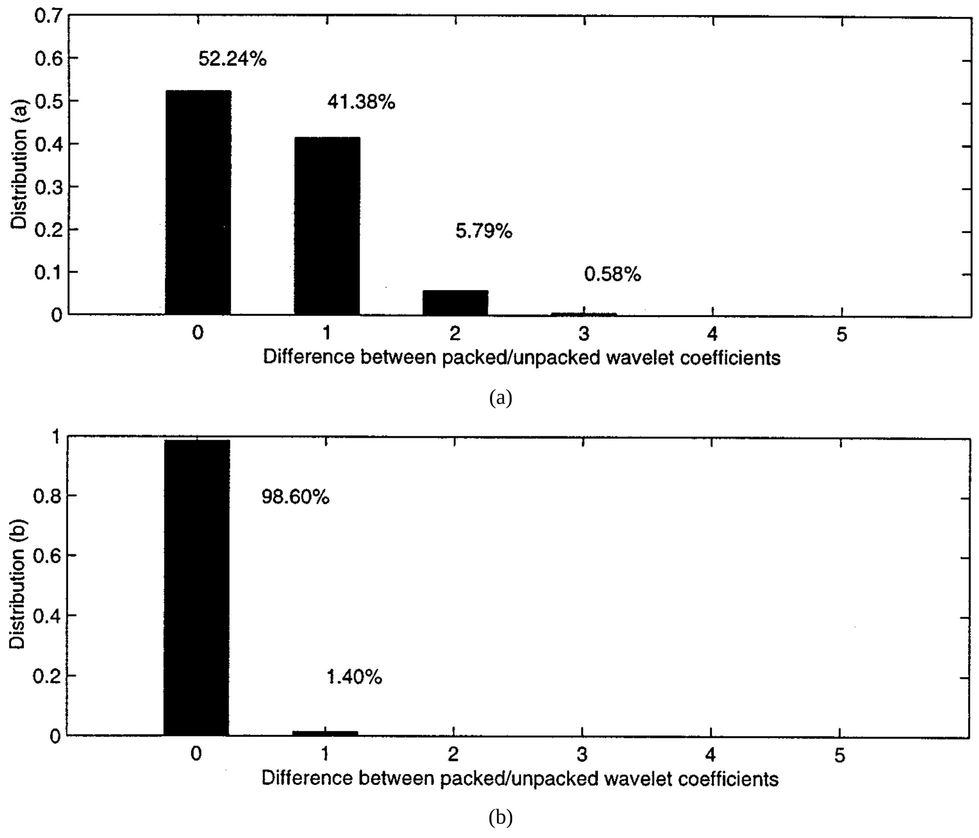

Fig. 2. Distribution of difference values between the packed and unpacked TS transforms applied to the *Peppers* image. (a) Before quantization. (b) After the scalar quantization (stepsize $= 16$).

intermediate results $\{r_1^o(k), d_1^o(k)\}$ and the final results $\{r_1(k), d_1(k)\}$ are also in packed format, which correspond to $\{\overline{r_1^o(k)}, \overline{d_1^o(k)}\}$ and $\{\overline{r_1(k)}, \overline{d_1(k)}\}$.

3) Unpack two rows/columns

$$r_1(2n, k) = HW(\overline{r_1(k)}), \quad r_1(2n+1, k) = LW(\overline{r_1(k)})$$
$$d_1(2n, k) = HW(\overline{d_1(k)}), \quad d_1(2n+1, k) = LW(\overline{d_1(k)})$$

where $HW(\cdot)$ and $LW(\cdot)$ represent the high word and low word of the packed integer, respectively.

Similarly, the algorithm for packed wavelet reconstruction can be described as follows.

1) Pack two rows/columns

$$\overline{r_1(k)} = (r_1(2n, k), r_1(2n+1, k))$$
$$\overline{d_1(k)} = (d_1(2n, k), d_1(2n+1, k)).$$

2) Undo the packed primal and/or dual lifting. Notice that the reconstructed signal is in packed format $\overline{r_0(k)}$.

3) Unpack two rows/columns

$$r_0(2n, k) = HW(\overline{r_0(k)}), \quad r_0(2n+1, k) = LW(\overline{r_0(k)}).$$

C. Performance Analysis

Although we may introduce a ± 1 difference in each packed addition or subtraction operation, the impact of this difference on image compression and reconstruction is rather negligible because: 1) it is not necessary that every packed addition or subtraction will contribute a $+1$ or -1 to s_i or d_i; 2) according to (6) and (7), alternating addition and subtraction can cancel out the 1-bit difference; and 3) right shift operations can further reduce the effect of bit propagation.

Consider to apply the TS transform to the *Peppers* image. In a three-level packed and unpacked TS transform, the maximum difference is only three, as expected, and its population is limited to 0.58% of the entire coefficient set. As is evident in Fig. 2, the majority of the coefficients is the same (52.24%) or the difference is only 1 (41.38%). The difference becomes even smaller and negligible after the scalar quantization, where 98.60% of the coefficients are the same and the difference for the rest 1.40% is limited to 1 (Fig. 2).

According to [5], every wavelet with finite filters can be obtained as the Lazy wavelet followed by a finite number of lifting steps and scaling. The Lazy wavelet is reversible for either packed or unpacked data because it does nothing but splitting the original signal into even and odd indexed samples. The lifting step is also reversible because the packed reconstruction is the exact reverse of the packed decomposition provided that there is no overflow. As a result, the entire packed wavelet transform is reversible when the dynamic range of its coefficients is within $[-2^{15}, 2^{15} - 1]$, which is true for most applications.

In our scheme, we have to unpack each row before we can pack the columns, and vice versa. This process will induce some overhead. However, in the practical implementation, pack and unpack operations can be done simultaneously with memory copy which is also required by the original wavelet transform

TABLE I
DIFFERENCE IN COMPRESSION RATIO

		girl 256 × 256	lena 512 × 512	peppers 512 × 512 × 3	man 1024 × 1024
S transform	unpacked	19.70	17.48	35.52	16.94
	packed	19.55	17.42	35.31	16.82
TS transform	unpacked	24.20	23.07	48.37	21.66
	packed	23.97	22.98	47.72	21.44
(5,3) transform	unpacked	22.18	21.69	46.43	19.13
	packed	22.25	21.65	46.27	19.09
(S+P) transform	unpacked	26.48	25.86	52.20	23.63
	packed	26.12	25.78	51.81	23.38

TABLE II
COMPARISON OF DECOMPOSITION TIME (MSEC) AND RECONSTRUCTED IMAGE QUALITY IN PSNR (dB)

		girl 256×256		lena 512×512		peppers 512×512×3		man 1024×1024	
		time	psnr	time	psnr	time	psnr	time	psnr
S transform	unpacked	46	29.70	226	29.80	318	28.45	978	28.89
	packed	33	29.94	150	30.06	213	28.68	674	29.08
TS transform	unpacked	51	30.74	246	30.96	348	29.57	1061	29.80
	packed	35	30.76	156	30.96	222	29.59	712	29.80
(5,3) transform	unpacked	62	32.05	268	32.39	397	30.41	1103	30.87
	packed	42	31.86	176	32.18	257	30.38	811	30.76
(S+P) transform	unpacked	52	30.71	251	30.90	357	29.43	1063	29.63
	packed	36	30.76	165	30.90	238	29.41	734	29.62

to move data between the image matrix and the working buffer for the wavelet transform. So the overhead of the pack/unpack operations is negligible.

IV. EXPERIMENT RESULTS

To verify the advantage of the packed integer wavelet transform, we compared its performance with that of the original integer wavelet transform for image compression. The coding algorithm used was a three-level wavelet transform, followed by a scalar quantization and stack-run coding [11]. No further entropy coding was used after the stack-run coding. The step-size used in the quantization is 16. We implemented four different integer wavelet transform algorithms [S, TS, (5, 3), and ($S+P$)] along with their packed versions. This experiment was done on a 166-MHz Pentium PC with 16-MB memory. We compared the performance of the packed and unpacked integer wavelet transforms on four images: *Girl* (256×256), *Lena* (512×512), *Peppers* (color, $512 \times 512 \times 3$), and *Man* (1024×1024), respectively. Table I shows the compression ratio for both the packed and unpacked computations in terms of the bitrate. One can see that the difference between the packed and unpacked transforms is very small. Table II shows the decomposition time savings and reconstructed image quality for the four packed wavelets versus the original ones. We can see that up to 37% savings in the decomposition time can be achieved by using our packed transform algorithms. We also observed that the speed-up factor is nearly image invariant and wavelet-type invariant. Since the packed wavelet transform is symmetric in terms of computation, it follows that we have very close performance in the reconstruction time as in the decomposition time. For the above experiment, the maximum difference in compression ratio between the packed and unpacked transforms is less than 0.25 and the difference in the reconstructed image quality (PSNR) is limited to 0.24 dB.

One may notice that in the S transform, the packed computation has consistently a better PSNR than the unpacked one, while in general it should be other way around. This is because the S transform is based on the Haar wavelet, which is the simplest among the four transforms shown. It produces many small high-frequency coefficients by the high-pass filter. Many of them are eliminated after the quantization. On the other hand, the minor errors generated by the packed computation move some of them to the nonzero range after the quantization. As a result, more high-frequency coefficients are kept than in the unpacked computation, which result in a better PSNR, but a lower compression ratio.

V. CONCLUSION

In this paper, an efficient mechanism for computing the integer wavelet transform, the packed integer wavelet transform, is described. The proposed packed transform can speed up the decomposition/reconstruction process up to 37% with a comparable performance in the compression ratio and reconstructed image quality. This approach is quite suitable for the applications in which the speed is a critical factor but no additional hardware such as vector processing devices or other embedded system is available. In reality, many types of special-purpose processors exist which can significantly speed up the computation of the wavelet transform. The current approach is not to replace them but to provide an alternative, especially when the extra hardware is not available to the users.

Although the current implementation of our proposed algorithms is based on 32-bit computers, the packed wavelet transform algorithm is not limited to 32-bit data path. More speed-up can be achieved if the software is tested on 64-bit machines, or future computers with even wider data path. The speed-up mechanism is made possible by using the lifting scheme. It is shown in [5] that the cost of the lifting algorithm for computing the wavelet transform is one half of the cost of the standard algorithm. For certain wavelet transforms, the lifting scheme can result in only addition, subtraction and shifting of integers. By using the packed integer wavelet transform, the computation cost can be further reduced. Finally, it should be pointed out that the current approach is not suitable for completely lossless image compression because it introduces a difference during the process of computation although the magnitude of the difference is very small.

REFERENCES

[1] A. Said and W. Pearlman, "A new fast and efficient image codec based on set partitioning in hierarchical tree," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, pp. 243–250, 1996.
[2] E. Schwartz, A. Zandi, and M. Boliek, "Implementation of compression with reversible embedded wavelets," in *Proc. SPIE 40th Annu. Meeting*, San Diego, CA, 1995.

[3] H. Chao and P. Fisher. An approach to fast integer reversible wavelet transforms for image compression. [Online]. Available: http://www.compsci.com/wchao/Publication/Pub no. 1.ps.gz

[4] W. Sweldens, "The lifting scheme: a custom-design construction of biorthogonal wavelets," *Appl. Comput. Harmon. Anal.*, vol. 3, no. 2, pp. 186–200, 1996.

[5] I. Daubechies and W. Sweldens, "Factoring Wavelet Transforms into Lifting Steps," Tech. Rep., Lucent Technologies, Bell Labs., Holmdel, NJ, 1996.

[6] J. Allen, "An approach to fast transform coding in software," *Signal Processing: Image Commun.*, vol. 8, pp. 3–11, 1996.

[7] S. Mallat, "A theory for multiresolution signal decomposition: the wavelet representation," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 11, pp. 674–693, 1989.

[8] I. Daubechies, "Ten lectures on wavelets," in *Proc. CBMS-NSF Regional Conf. Series Applied Mathematics*, 1992.

[9] J. Villasenor, B. Belzer, and J. Liao, "Wavelet filter evaluation for image compression," *IEEE Trans. Image Processing*, vol. 4, pp. 1053–1060, 1995.

[10] R. Gopinath and C. Burrus, *Introduction to Wavelets and Wavelet Transform, A Primer*. Englewood Cliffs, NJ: Prentice-Hall, 1998.

[11] M. Tsai, J. Villasenor, and F. Chen, "Stack-run image coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, pp. 519–521, Oct. 1996.