PACKED INTEGER WAVELET TRANSFORM CONSTRUCTED BY LIFTING SCHEME

Chengjiang Lin, Bo Zhang, Yuan F. Zheng

Department of Electrical Engineering
The Ohio State University
Columbus, OH 43210, USA
zheng@ee.eng.ohio-state.edu

ABSTRACT

A new method for speeding up the integer reversible 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, the proposed method can save the decomposition/reconstruction time by up to 37 percent on 32-bit machines in comparison with the original wavelet transform algorithms. Furthermore, the packed integer wavelet transform requires much less working memory.

1. INTRODUCTION

The wavelet transform has received much attention in the field of image compression [1, 2, 3]. It has lower distortion than the DCT based JPEG approach when the compression ratio is high. 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 biorthogonal wavelets [4]. Recently, biorthogonal wavelets constructed by the lifting scheme have been identified as very promising filters for lossless/lossy image compression applications [3, 5]. Since the lifting scheme makes optimal use of similarities between the high and low pass filters, the computation complexity can be reduced by a factor of two compared with traditional fast wavelet transform algorithms. With certain modifications, the corresponding wavelet transform can be calculated even 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.

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 completely new approach for further speeding up the computation. 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 not affected by the packed transform method.

This paper is organized as follows: Section 2 describes the basic idea of the wavelet transform and the integer wavelet transform based on the lifting scheme. Section 3 introduces the packed computation and gives the packed integer wavelet transform algorithm. Section 4 presents the experimental results of the new method for grayscale and color image compression. Conclusions are offered in Section 5.

2. INTEGER WAVELET TRANSFORM

2.1. Wavelet Transform and the Lifting Scheme

The wavelet transform can be considered as a subband transform and implemented with a filter bank [6]. Figure 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 the synthesis filters are added together to reconstruct the original signal. The conditions for perfect reconstruction are given by [7]

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

When $\tilde{h} = h$ and $\tilde{g} = g$, $\{\tilde{h}, \tilde{g}, h, g\}$ forms an orthogonal wavelet transform.

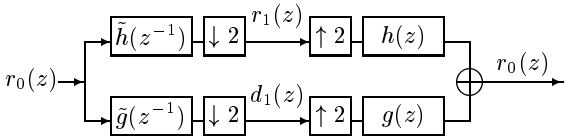

Figure 1: Basic filter bank for biorthogonal wavelet transform

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 Figure 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 $x(z)$. For the Lazy wavelet transform [4], its corresponding polyphase matrix is a 2×2 unit matrix. The readers are referred to [6, 7] for details on wavelet and subband transforms.

The lifting scheme [4, 5] provides a new approach for construction of biorthogonal wavelets with compact support. Based on an Euclidean algorithm for Laurent polynomials, 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 $\tilde{h}^{new}$ complementary to $\tilde{g}$ is created:

$$P^{new}(z) = \begin{bmatrix} \tilde{h}_e^{new}(z) & \tilde{h}_o^{new}(z) \\ \tilde{g}_e(z) & \tilde{g}_o(z) \end{bmatrix} = \begin{bmatrix} 1 & s(z) \\ 0 & 1 \end{bmatrix} P(z).$$

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

$$P^{new}(z) = \begin{bmatrix} \tilde{h}_e(z) & \tilde{h}_o(z) \\ \tilde{g}_e^{new}(z) & \tilde{g}_o^{new}(z) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ t(z) & 1 \end{bmatrix} P(z).$$

Both $s(z)$ and $t(z)$ are Laurent polynomials. 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]. We are not going to discuss how to choose $s_i(z)$ and $t_i(z)$, but one can refer to references [3, 5] for clarification of this process.

2.2. Integer Wavelet Transform

One of the most interesting parts of the lifting scheme is that it can be used to create integer wavelet transformations. Since we can obtain every wavelet transform by using lifting, it follows that we can build an integer version of every wavelet transform. It has been proved 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 $\{\tilde{h}^{new}, \tilde{g}, h, g^{new}\}$ 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.

- Forward transform:
  1. 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.
  2. Update of the low subband $r_1^o(k)$ by applying the s filter on the high subband $d_1(k) = d_1^o(k)$.

$$r_1(k) = r_1^o(k) + Int\left(\sum_n d_1(k-n)s(n)\right), \quad (1)$$

  or, update of the high subband $d_1^o(k)$ by applying the t filter on the low subband $r_1(k) = r_1^o(k)$.

$$d_1(k) = d_1^o(k) + Int\left(\sum_n r_1(k-n)t(n)\right). \quad (2)$$

- Inverse transform:
  1. Undo 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 (1) and (2). In practice, this comes down to simply changing each $+$ into a $-$ and vice versa.
  2. Inverse transform using the synthesis filters (h, g) on the low pass and high pass subbands $r_1^o(k)$ and $d_1^o(k)$ and get back the original signal $r_0(n)$.

Notice that if we carefully choose the parameters $\{s, t\}$ in (1) and (2) 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 best biorthogonal filter banks for image compression, the (2, 6) wavelet that corresponds to the TS transform [2, 8], the decomposition can be done via the following lifting steps [3]:

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

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-bit or 64-bit computational capability of modern computers.

3. 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.

3.1. Packed Computation

Let $A = (a_{n-1}, \cdots, a_1, a_0)$ and $B = (b_{n-1}, \cdots, b_1, b_0)$ be the packed integer of $a_i, b_i \in [-2^{15}, 2^{15} - 1]$ (i=0, $\cdots$, 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 $S = (s_{n-1}, \cdots, s_1, s_0)$, $D = (d_{n-1}, \cdots, d_1, d_0)$, $s_i, d_i \in Z$. 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 result of packed addition and subtraction may be different from its unpacked version, the difference is quite neglectable (limited to ± 1). Based on the definition of s_i and d_i ($0 \leq i \leq n-1$), we have:

$$s_i = \begin{cases} a_i + b_i & \text{if } s_{i-1}a_{i-1}b_{i-1} \geq 0 \text{ or } i = 0, \\ a_i + b_i + 1 & \text{if } s_{i-1}a_{i-1}b_{i-1} < 0, \end{cases} \quad (3)$$

$$d_i = \begin{cases} a_i - b_i & \text{if } d_{i-1}a_{i-1}b_{i-1} \geq 0 \text{ or } i = 0, \\ a_i - b_i - 1 & \text{if } d_{i-1}a_{i-1}b_{i-1} < 0. \end{cases} \quad (4)$$

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: $\{Int(a_i/2^l)\}$, $i = 0, \cdots, n-1$.

3.2. Packed Integer Wavelet Transform

In general, multiple coefficients can be packed into one integer provided that the width of the datapath 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 h and $\tilde{g}$. The algorithms are the same as (1)(2) except that we use $\overline{r_0(n)}$ as the input instead of $r_0(n)$. The 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
   $$r_1(2n, k) = HW(\overline{r_1(k)}), \qquad r_1(2n+1, k) = LW(\overline{r_1(k)}),$$
   $$d_1(2n, k) = HW(\overline{d_1(k)}), \qquad d_1(2n+1, k) = LW(\overline{d_1(k)})$$
   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
   $$r_0(2n, k) = HW(\overline{r_0(k)}), \qquad r_0(2n+1, k) = LW(\overline{r_0(k)}).$$

3.3. 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 neglectable 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 (3)(4), alternating addition and subtraction can cancel out the 1 bit difference; (3) Right shift operations can further reduce the effect of bit propagation. For example, the maximum difference between the coefficients in a 3-level packed and unpacked TS transform of the *Peppers* image is only 3 and its population is limited to 0.58% of the entire coefficient set. 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.

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 if the dynamic range of its coefficients is within $[-2^{15}, 2^{15} - 1]$.

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

4. 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 [9]. No further entropy coding was used after the stack-run coding. 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

Table 1: Comparison of decomposition time (msec)

		Girl	Lena	Peppers	Man
S transform	unpack	46	226	318	978
	pack	33	150	213	674
TS transform	unpack	51	246	348	1061
	pack	35	156	222	712
(5,3) transform	unpack	62	268	397	1103
	pack	42	176	257	811
(S+P) transform	unpack	52	251	357	1063
	pack	36	165	238	734

on four images: *Girl* (256×256), *Lena* (512×512), *Peppers* (color, $512\times512\times3$), and *Man* (1024×1024), respectively. Table 1 shows the decomposition time savings for the four packed wavelets v.s. 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. Figure 2 shows the original *Peppers* image (grayscale version) and the reconstructed images by the TS transform and the packed-TS transform (3-level), respectively.

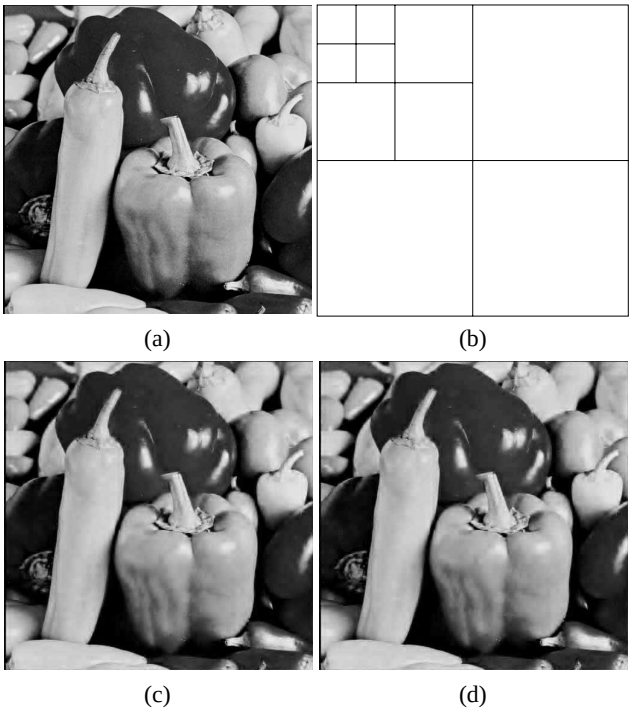

Figure 2: Comparison of reconstructed image quality. (a) The original image, (b) 3-level multiresolution decomposition, (c) Reconstructed image by TS transform, compression ratio=21.92, PSNR=31.14dB, (d) Reconstructed image by packed-TS transform, compression ratio=21.73, PSNR=31.15dB

5. CONCLUSIONS

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 percent 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, such as software based videoconferencing and real-time image compression systems. 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 datapath. More speed-up can be achieved if the software is tested on 64-bit machines, or future computers with even wider datapath. 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. This paper offers four biorthogonal wavelet transforms which fall into this category. Increasing the membership of this category presents a challenging task to future research.

6. REFERENCES

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

[2] E. Schwartz, A. Zandi, and M. Boliek, "Implementation of compression with reversible embedded wavelets," in *Proc. of SPIE 40th Annual Meeting*, San Diego, CA, 1995.

[3] H. Chao and P. Fisher, "An approach to fast integer reversible wavelet transforms for image compression," 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: Technical report, Bell Laboratories, 1996.

[6] S. Mallat, "A theory for multiresolution signal decomposition: the wavelet representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, no. 7, pp. 674–693, 1989.

[7] I. Daubechies, "Ten lectures on wavelets," in *CBMS-NSF Regional Conference Series in Applied Mathematics, SIAM*, Philadelphia, 1992.

[8] J. Villasenor, B. Belzer, and J. Liao, "Wavelet filter evaluation for image compression," *IEEE Transactions of Image Procession*, vol. 4, no. 8, pp. 1053–1060, 1995.

[9] M. Tsai, J. Villasenor, and F. Chen, "Stack-run image coding," *IEEE Transactions on Circuits and System for Video Technology*, vol. 6, no. 5, pp. 519–521, 1996.