ELSEVIER

Signal Processing: *Image Communication* 16 (2000) 401–411

SIGNAL PROCESSING: IMAGE COMMUNICATION

www.elsevier.nl/locate/image

Low computational complexity enhanced zerotree coding for wavelet-based image compression

Bing-Fei Wu[a,]*, Chorng-Yann Su[b]

[a]*Department of Electrical and Control Engineering, National Chiao Tung University, Hsinchu 300, Taiwan*
[b]*Department of Industrial Education, National Taiwan Normal University, Taipei 106, Taiwan*

Received 16 June 1998

Abstract

The embedded zerotree wavelet (EZW) algorithm, introduced by J.M. Shapiro and extented by A. Said and W.A. Pearlman, has proven to be a computationally simple and efficient method for image compression. In the current study, we propose a novel algorithm to improve the performance of EZW coding. The proposed method, called enhanced zerotree coding (EZC), is based on two new techniques: adaptive multi-subband decomposition (AMSD) and band flag scheme (BFS). The purpose of AMSD is to change the statistics of transformed coefficients so that the coding performance in peak signal-to-noise ratio (PSNR) can be elevated at a lower bit rate. In addition, BFS is used to reduce execution time in finding zerotrees. In BFS the tree depths are controlled, therefore, many unnecessary comparison operations can be skipped. Experimental results show that the proposed algorithm improves the performance of EZW coding and requires low computational complexity. In addition, the property of embedded coding is preserved, which enables a progressive transmission. © 2000 Elsevier Science B.V. All rights reserved.

Keywords: Image coding; Zerotree coding; Adaptive multi-subband decomposition; Band flag scheme

1. Introduction

Transform coding is a well-known and widely used technique in image compression. The purpose of the transformation is to produce decorrelated coefficients and remove redundancy. In the past decade, the discrete cosine transform (DCT) has been the most popular because it provides an almost optimal performance and can be implemented at a reasonable cost. However, discrete wavelet transform (DWT) has been widely used recently because of its ability to solve the blocking effect introduced by DCT and its suitability in multi-resolution analysis. Moreover, in a pyramidal decomposition scheme, DWT also yields some degree of the self-similarity across different scales, which is helpful in image compression.

By taking advantage of DWT, the embedded zerotree wavelet (EZW) coding algorithm, introduced by Shapiro [12], has shown that the EZW was not only competitive in its performance with the most complex techniques but also extremely fast in execution. In addition, it also has several other merits: it requires no pre-stored tables, no training, and produces fully embedded codes, i.e., codes corresponding to a lower rate R bits always

* Corresponding author. Tel.: + 886-3-5712121, ext. 54313; fax: + 886-3-5712385.
E-mail address: bwu@cc.nctu.edu.tw (B.-F. Wu).

forms the first R bits of those codes with a higher rate. With embedded coding, it is possible to stop the decoding process at any point of the compressed file, and to allow coding/decoding to the exact desired rate or distortion.

Said and Pearlman extended EZW by presenting a different implementation called set partitioning in hierarchical trees (SPIHT) [11]. The performance of SPIHT not only surpasses the original EZW in both the peak signal-to-noise ratio (PSNR) and the execution time, but also shows that SPIHT is almost the best coder in related studies. Because EZW-like coder can provide an efficient coding for still images and visual textures, the MPEG-4 standard also uses it in the visual texture mode [7]. In addition, EZW-like coder can also provide spatial and quality scalabilities, which are the desired functionalities of the MPEG-4 standard [3,5,13]. Another extended version of EZW, called zerotree wavelet video coder, is applied to video coding [9].

EZW's excellent performance is based on three concepts [11]: (1) partial ordering of the transformed coefficients by magnitude, (2) ordered bit plane transmission, and (3) exploitation of self-similarity across different scales of an image wavelet transformation. Because the distribution of the magnitudes of transformed coefficients will affect EZW coding, one method to improve the EZW coding performance is to change the statistics of the magnitudes. Demirciler and Freeman [4] suggested a fixed type of multi-bandwidth wavelet decomposition for all images. They applied distinct wavelet functions to the decomposition of different scales and performed better than the original EZW. However, for different images, an adaptive type of decomposition is more effective than a fixed type. To obtain a better transforming efficiency, some researchers proposed adaptive types of wavelet decomposition, such as the wavelet packet [10] and the space-frequency decomposition [16,6]. However, these methods are generally based on a rate-distortion sense, and therefore high computational complexity is required to find the optimal operating point in the rate-distortion curve. In addition, they could not give their best performance with a single embedded file, and they required, for each rate, the optimization of a certain parameter.

To retain all the merits of EZW coding, we present another adaptive type of wavelet decomposition. The proposed method, called adaptive multi-subband decomposition (AMSD), is a type of wavelet packet, but it has a lower computational complexity. The idea of AMSD arises from the observation in Mallat's work [8]. In that work, he used a two-dimensional (2-D) DWT to decompose an image with a rectangular region, and produced two vertical bars in the band HL and two horizontal bars in the band LH. The first letter of HL or LH denotes that the image is either filtered by a lowpass filter (L) or a highpass filter (H) in the horizontal orientation. Meanwhile, the second letter is the state of the vertical orientation. The coefficients in the bars are of high correlation. Therefore, if an image consists of many rectangular regions, we expect to further remove the correlation of the coefficients in the bars by using one-dimensional (1-D) DWT to decompose the band HL in the vertical orientation, and the band LH in the horizontal orientation. This idea is instinctive but later we will show that the determination for the decomposition is based on the feature of EZW coding.

In addition to AMSD scheme, we also propose a band flag scheme (BFS) to speed up a zerotree coding. Generally, a tree consists of all the nodes with a parent–child relationship. A node (i.e. transformed coefficient) $C(i,j)$ is considered insignificant with respect to a given threshold T_n, if $|C(i,j)| < T_n$. Otherwise, it is significant. For convenience, we define a significant test as

$$S_{T_n}(C(i,j)) = \begin{cases} 1 & \text{if } |C(i,j)| \geqslant T_n, \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

A tree is called a zerotree if all its nodes are insignificant. To find out a zerotree, the regular approach is to perform (1) for each node of the tree from root to leaf nodes in breadth-first approach. If one results in 1, the tree is not a zerotree and the subsequent tests can be waived. Otherwise, the tree is a zerotree and we will not get the answer until the last test is performed. In fact, we can use BFS to omit many unnecessary tests. Observing the average magnitude of transformed coefficients in each band, we find that the magnitudes generally decay

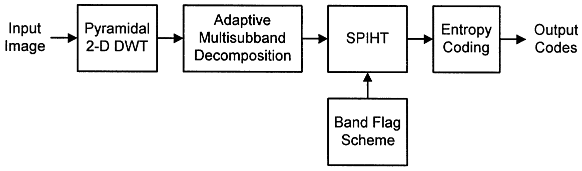


Fig. 1. Block diagram of enhanced zerotree coding.

from the coarsest scale to the finest scale. Therefore, for some given thresholds it is possible for all the nodes, located on the bands of finer scale, to be insignificant. For these bands, we can temporarily disable them for coding to save running time. Later, we will show that, by using BFS, we can speed up a zerotree coding without increasing computational complexity.

Fig. 1 illustrates the block diagram of our proposed enhanced zerotree coding (EZC). The coding algorithm of EZC is based on SPIHT [11]. The difference is that EZC is supplemented with AMSD and BFS. It is worth mentioning that the use of AMSD does not change the parent–child dependency of tree structure, and, by using BFS, it only changes the number of nodes to be tested. Thus, it is easy to apply the algorithm of SPIHT to EZC. Notably, the entropy coding used herein is an adaptive arithmetic coding referring to Witten et al. [15]. For simplicity, the following sections explain the designs of AMSD and BFS in binary bit plane excluding the entropy coding. We want to emphasize that the proposed coder preserves the property of the embedded coding. In other words, the transmitted codes or compressed file is completely embedded, so that a single file for an image at a given bit rate can be truncated at various points and decoded to give a series of reconstructed images at lower bit rates.

The organization of this paper is listed as follows. Section 2 briefly introduces the zerotree algorithm of SPIHT. For more details refer to [11]. Section 3 discusses the design of AMSD. Section 4 describes BFS and how to store the significant information of each band for decoding. Section 5 gives some experimental results to show the efficiency of proposed EZC algorithm. The performance is compared with SPIHT. The conclusions are reported in Section 6.

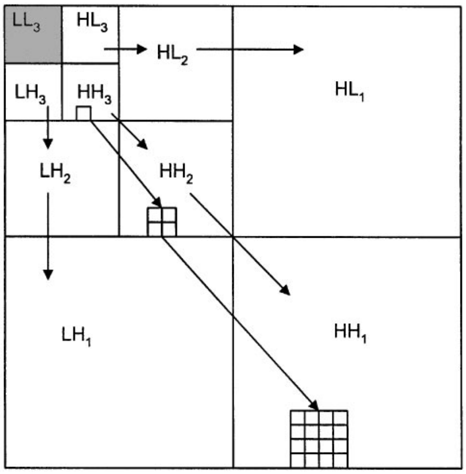


Fig. 2. Pyramidal wavelet decomposition and parent–child dependency for tree structures.

2. Brief review of SPIHT

Fig. 2 illustrates a typical three-scale pyramidal decomposition of an image. The image is generated by three stages of two-dimensional (2-D) DWT [1]. The notations LL_i, HL_i, LH_i and HH_i denote the output channels from the ith stage. The parent–child dependency for tree structures is also demonstrated. In Fig. 2, except the nodes located on bands HL_1, LH_1 and HH_1, each node has four children nodes. The nodes in LL_3 are isolated, and they will be coded independently.

We call a node (i.e. transformed coefficient) $C(i,j)$ at a coarse scale a parent. All nodes at the next finer scale with the same spatial location, and of similar orientation are called children, this set denoted $O(i,j)$. More precisely, $O(i,j) = \{C(2i,2j), C(2i,2j+1), C(2i+1,2j), C(2i+1,2j+1)\}$. All nodes at all finer scales with the same spatial location, and of similar orientation are called descendents, denoted $D(i,j)$. A set $L(i,j)$ is defined as $L(i,j) = D(i,j) - O(i,j)$, and the set $\boldsymbol{H}$ is the group of coordinates of all the tree roots (HL_3, LH_3 and HH_3 in Fig. 2). We also refer to a set as significant if one of its nodes is tested by (1) and the result is 1. If $D(i,j)$ is significant, then it is partitioned into $O(i,j)$ and $L(i,j)$. If not, it is a zerotree of type A. If $L(i,j)$ is significant, then it is partitioned into $\{D(2i,2j),$

$D(2i, 2j+1)$, $D(2i+1, 2j)$, $D(2i+1, 2j+1)\}$. Otherwise, it is a zerotree of type B. If we encounter a zerotree, we code such tree as a zerotree symbol, and avoid to code all its nodes. The nodes are scanned by the order of importance. It is performed so that no child is scanned before its parent. Therefore, one starts scanning the nodes $C(i,j)$ for $(i,j) \in \boldsymbol{H}$ and the sets $D(i,j)$ for $(i,j) \in \boldsymbol{H}$. The result of significant test for a node or for a set is coded. In addition, for each node $C(i,j)$, if it is significant, its sign bit is also coded.

The process begins with deciding the maximal value of n of threshold T_n. In a uniform successive approximation quantization (USAQ), the value of n can be obtained by using

$$n = \left\lfloor \log_2 \max_{(i,j)} \{|C(i,j)|\} \right\rfloor, \tag{2}$$

where $\lfloor x \rfloor$ is to truncate x near to zero. Then the following two passes, the sorting pass and refinement pass, are used for every n value. In the sorting pass, we scan $C(i,j)$, $D(i,j)$ and $L(i,j)$ in the scanning order described above, extract significant nodes, and put them into a list of significant pixels (LSP). In the refinement pass, however, another bit of precision is added to the magnitudes of nodes in the LSP. We decrease n by one, i.e., cut the threshold in half, and use these two passes for each n in the order of the refinement pass first until some terminating condition is met, such as that the bit budget is exhausted.

3. Adaptive multi-subband decomposition

The purpose of AMSD is to further decompose the bands LH_i and HL_i as shown in Fig. 2. We use only 1-D DWT to the horizontal orientation of the bands LH_i, and to the vertical orientation of the bands HL_i. To preserve self-similarity across different scales, we take all the bands LH_i into consideration. This means, if LH_1 is decomposed, all the bands LH_i for $i > 1$ are decomposed as well. Likewise, all the bands HL_i are considered together. Fig. 3 illustrates such decomposition. In this illustration, all the bands LH_i are further decomposed at once by using 1-D DWT. Notably, we limit the

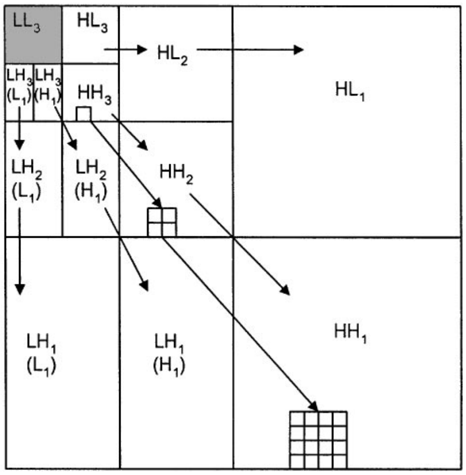


Fig. 3. An example of adaptive multi-subband decomposition. Herein, the bands LH_i for all i are further decomposed at once by using 1-D DWT.

times of the 1-D decomposition at the same orientation to two. In addition, the parent–child dependency of a tree structure remains unchanged no matter how many times the 1-D decomposition is performed.

The algorithm for AMSD is as follows. First of all, check the LH_1 subband, and partition the entire region of LH_1 subband into many nonoverlapped microblocks, where each microblock M_k is of size 2×2 and is composed with four nodes. Calculate the number of bits N_k for each M_k by using

$$N_k = 4 \left\lfloor \log_2 \max_{(i,j)\in M_k} |C(i,j)| \right\rfloor. \tag{3}$$

Let $N_{\mathrm{old}} = \sum_k N_k$. Next, decompose LH_1 into low-frequency containing subband $\mathrm{LH}_1(\mathrm{L}_1)$ and high-frequency with subband $\mathrm{LH}_1(\mathrm{H}_1)$. Calculate the number of bits $\tilde{N}_k$ for each $\tilde{M}_k$ by using a similar formula as (3), where $\tilde{M}_k$ locates at the same place as M_k but the magnitudes of the nodes it contains have been changed. Let $N_{\mathrm{new}} = \sum_k \tilde{N}_k$. Finally, if $N_{\mathrm{new}} > N_{\mathrm{old}}$, recover to the state of the band LH_1 and exit. Otherwise, check the subband $\mathrm{LH}_1(\mathrm{L}_1)$ by using this algorithm. Notably, when $\mathrm{LH}_1(\mathrm{L}_1)$ is checked, the number of microblocks is reduced to half of the original. Assuming that decomposing $\mathrm{LH}_1(\mathrm{L}_1)$ generates $\mathrm{LH}_1(\mathrm{L}_2)$ and $\mathrm{LH}_1(\mathrm{H}_2)$, we will

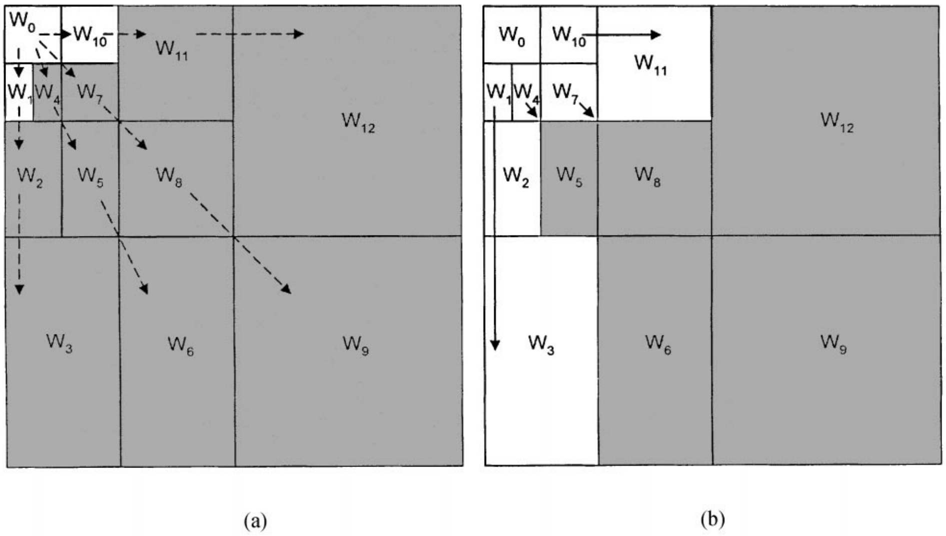


Fig. 4. Example of band flag scheme. The shadow regions representing these subbands are disabled for coding with respect to the current threshold (a) for T_n; (b) for $T_{n-1} = T_n/2$. The parent–child dependency of bands is also displayed in (a). In (b) the solid arrows indicate how far the tree depth goes.

exit this algorithm before decomposing $LH_1(L_2)$. Since the 1-D decomposition is done at most two times, we can use 2 bits to record how many times the decomposition occurs. Likewise, this algorithm is applied to the HL_1 subband.

Now, we will explain why we use (3) to calculate the number of bits cost for M_k. Consider a $D(i,j)$ with (i,j) located on LH_2. Then the nodes of $O(i,j)$ are located on LH_1. Assuming that $D(i,j)$ is significant (not a zerotree) and corresponds to the current threshold, it will be partitioned into $O(i,j)$ (i.e. four single nodes) according to the algorithm of SPIHT. For each node, if it is insignificant, it will be moved to a list of insignificant pixels (LIP). Otherwise, it will be moved to LSP. To identify the state of each node, one bit should be sent out even if the node is insignificant. At the next threshold, one bit is sent out for each node in LIP to denote whether it is significant or not, and one bit is sent out for each node in LSP to refine its magnitude. Hence, once one of the four nodes is significant, the other three nodes will cost the same number of bits as the significant one. This accounts for why we use (3) to calculate the number of bits cost for M_k.

By using the proposed algorithm, a different image may produce a different decomposition on the bands HL_i and LH_i for all i, which accounts for why the proposed decomposition is an adaptive multi-subband decomposition.

4. Band flag scheme

BFS is used to reduce the time spent on zerotree searching. In BFS, each band W_k is associated with a flag F_k to denote its state (enabled or disabled). If F_k is true, W_k is enabled for encoding and decoding. Otherwise, W_k is disabled.

The algorithm of BFS in encoding is as follows. Firstly, set F_k for all k to be false and evaluate the band information B_k for each W_k by using

$$B_k = \left\lfloor \log_2 \max_{(i,j)\in W_k} |C(i,j)| \right\rfloor. \tag{4}$$

Next, calculate the modified band information $\tilde{B}_k$. If W_k–W_l has a parent-child relationship (for example W_4–W_5 and W_0–W_{10} in Fig. 4(a)), then let $\tilde{B}_k = \max(B_k, B_l)$. In addition, let $\tilde{B}_k = B_k$ if W_k has no child (W_3, W_6, W_9 and W_{12} in Fig. 4(a)). Now, the values of $\tilde{B}_k$'s are descending from the coarsest scale to the finest scale. Therefore, for all k (excluding $k = 0$), we can differentially code $\tilde{B}_k$ by using the relation of parent–child dependency of bands. Finally, for each threshold T_n ($T_n = 2^n$), set F_k to be true if $\tilde{B}_k \geqslant n$ where n is the index of the threshold T_n. Notably, the value of n is descended in USAQ. Thus, if F_k is true at current threshold, the state of F_k keeps unchanged at the following thresholds.

Fig. 4 illustrates an example of BFS. The shadow regions representing these bands are disabled with respect to the current threshold. In this example, only the bands W_0, W_1 and W_{10} are enabled at T_n (Fig. 4(a)), and so are the bands W_2, W_3, W_4, W_7 and W_{11} at T_{n-1} (Fig. 4(b)). For larger value of n, more bands are disabled and it results in a shorter tree depth. This can help us save running time if we encounter zerotrees. For example, if a $D(i,j)$ has coordinate (i,j) located on W_1 and is a zerotree (of type A), by disabling W_2 and W_3 (as shown in Fig. 3(b)) we can save 20 tests of (1), four for the nodes in W_2 and 16 for the nodes in W_3, for this zerotree. In Fig. 4(a) we also indicate the parent–child dependency of bands. In addition, the arrow in Fig. 4(b) denotes how far the tree depth goes.

Note that we use $\tilde{B}_k$ to serve as band enabled information instead of B_k. This arrangement is made for two purposes. First, if some subbands violate the decaying spectrum, we can avert the case of a tree with many coefficients in the disabled subbands. Second, since the value of $\tilde{B}_k$ is descending from the coarsest scale to the finest scale, they can be differentially encoded to save bits.

In addition, it is worth mentioning that by evaluating B_k, it does not increase computational load because finding the maximal value of n of T_n is the first step of EZW-like coding, and the value of n can be obtained by using $n = \max_k\{B_k\}$.

The algorithm of BFS in decoding is quite simple. Begin with setting F_k to be false for all k and decoding the values $\tilde{B}_k$ for all k. Then for each T_n, set F_k to be true if $\tilde{B}_k \geqslant n$.

5. Experimental results

Four images, Lena, Barbara, Goldhill and Mandrill, are used for testing. All the images are monochrome with size 512 × 512. Each pixel consists of 8 bits. The image “Lena” is composed of large smooth skin areas. The image “Barbara” consists of many rectangular regions in the background and the table. The image “Goldhill” is characterized by occupying houses and farms that are like rectangles. As for the image “Mandrill”, it is full of texture regions and is used to show the power of AMSD.

The biorthogonal wavelet filter pair with lengths 9 and 7 is used [1]. Symmetric extension is applied to the edges of image for filtering [2]. For a 512 × 512 image, we perform six-scale pyramidal decomposition. The coefficients in LL_6 subband are coded independently. To simplify the coding algorithm, the LL_6 subband is decomposed by Haar transform and encoded by original EZW coding.

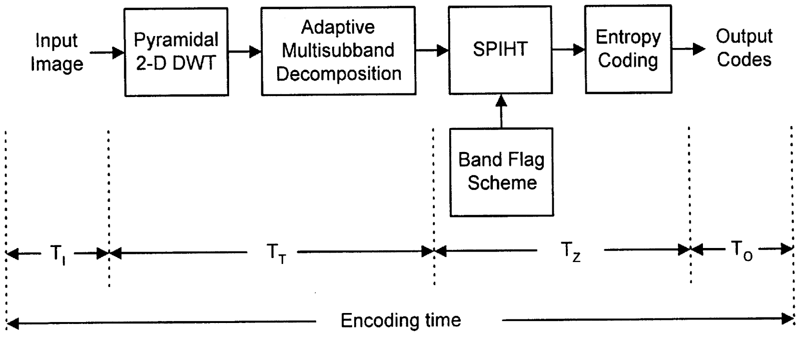


Fig. 5. Definition of encoding time T_E. T_I: input time; T_T: transform time; T_Z: zerotree-encoding time; T_O: output time. $T_E = T_I + T_T + T_Z + T_O$.

The performance of our image coder is measured by the PSNR, which is defined as

$$\mathrm{PSNR} = 10\log_{10}\left(\frac{255^2}{\mathrm{MSE}}\right)\mathrm{dB}. \tag{5}$$

Herein, MSE represents the mean-squared error between the original images and reconstructed ones. We emphasize that the bit rates are calculated from the actual size of the compressed files rather than the entropy estimates.

In the following statements the encoding time T_E is defined as the sum of input time T_I (for reading image), transform time T_T, zerotree-encoding time T_Z (entropy coding included), and output time T_O (for writing codes). Fig. 5 illustrates this definition. The original encoding time T_W is the result of coding without AMSD and without BFS. At bit rate 0.5 bpp, for Lena image T_W is about 3.56 s ($T_I + T_O = 0.27$ s, $T_T = 0.77$ s and $T_Z = 2.52$ s).

Fig. 6 shows the efficiency of AMSD. The results of coding with AMSD in PSNR outperform those of coding without AMSD over a wide range of bit-rates. Especially, the improvement is obvious for lower bit-rates. At the bit-rate 0.5 bits per pixel (bpp), adding AMSD will elevate the PSNR by 0.5 dB for Barbara, 0.22 dB for Goldhill, 0.07 dB for Lena, and 0.11 dB for Mandrill. The difference is caused because the Barbara and Goldhill images occupy a lot of rectangular regions. To reveal the potential of AMSD, we also test a special image, with pure rectangular structures (see Fig. 7(a)). The experimental results are shown in Fig. 7(b). On average, 1.46 dB improvement can be obtained. We emphasize that AMSD is not optimal for all bit-rates, but it performs better than the original (no AMSD) at lower bit-rate because it forces more the magnitudes of coefficients near to zero. The cost of adding AMSD to further decompose the HL$_i$ and LH$_i$ subbands is about 0.7 s, corresponding to 19.7% (0.7/3.56) of T_W. The running time is calculated by an IBM-compatible PC with AMD K6-166 MHz CPU, 32 MB RAM, and the platform WinNT 4.0.

Fig. 8 illustrates the comparison in T_E between coding with and coding without BFS. On average, by the use of BFS, the encoding time can be reduced by 1.4 s, corresponding to 39.3% (1.4/3.56) of T_W. The results of the running time can be further reduced if we develop the algorithm by using the optimal codes. The reason BFS can speed up a zerotree coding is that many unnecessary tests of (1) are omitted. For the Barbara image, at the threshold T_{10} (i.e. 1024) the number of tests required for searching zerotrees with BFS and that without BFS are 648 and 1,572,744, respectively. Consequently the number of saved tests is 99.9% (1,572,096/1,572,744) of the number of tests without BFS. When at the threshold T_6 (i.e. 64), the number of tests required for searching zerotrees with BFS

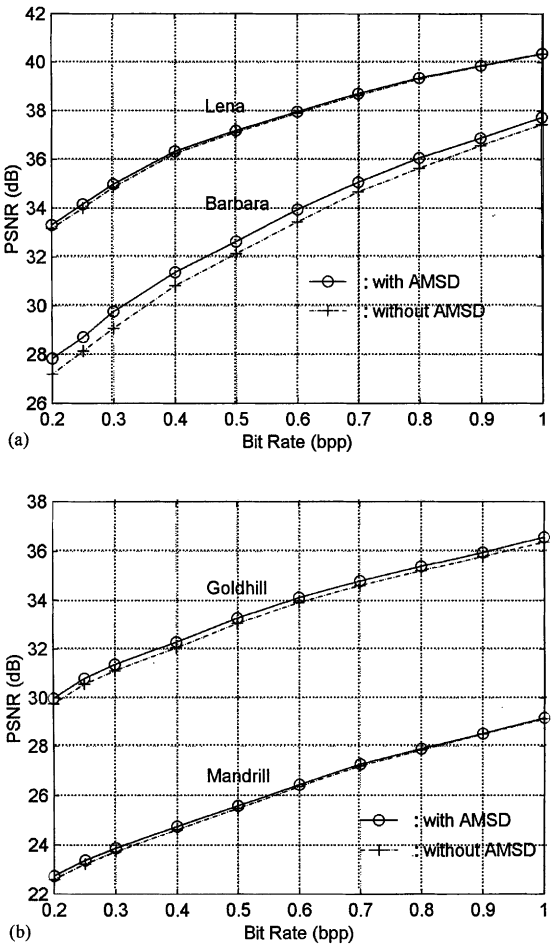
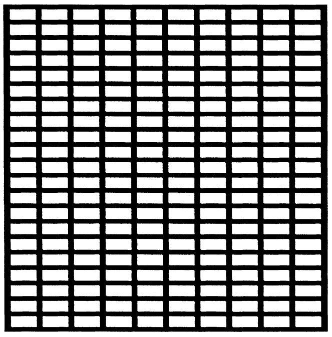
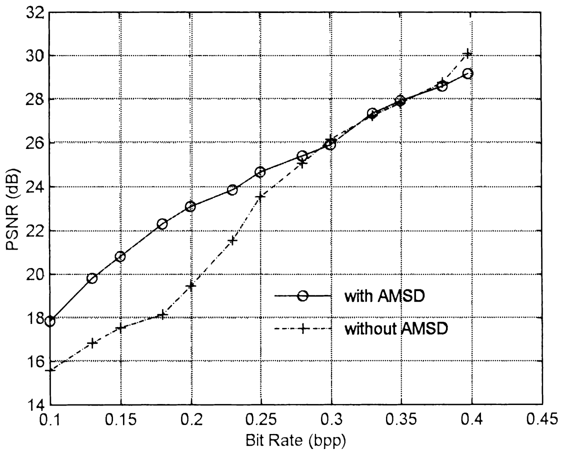


Fig. 6. Comparison of performance in PSNR between coding with and coding without AMSD: (a) for Lena and Barbara images; (b) for Goldhill and Mandrill images.


(a)



(b)

Fig. 7. The potential of AMSD: (a) an image with pure rectangular structures; (b) the comparison of performance in PSNR between coding with and coding without AMSD.

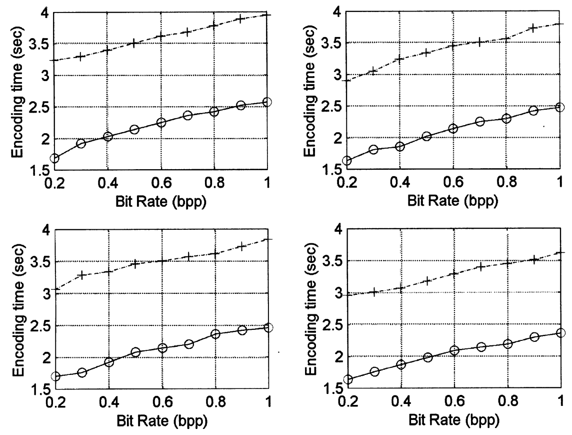


Fig. 8. Comparison of performance in encoding time between coding with and coding without BFS. The solid line denotes the results of coding with BFS, while dashed-dot line is the results without BFS. Left-top for Lena; right-top for Barbara; left-bottom for Goldhill; right-bottom for Mandrill.

and that without BFS are separately 286,005 and 2,629,173, corresponding to that the number of saved tests is 89.1% of the number of tests without BFS. If threshold T_n is small enough, all the bands are enabled, leading to that the number of saved tests stops increasing and that the percentage reduces. However, for general applications the improvement of using BFS is still significant because the coder stops at a threshold T_n for $n \geqslant 2$. For the Barbara image, the percentage is 67.2% at the threshold T_2. Notably, in these applications the reconstructed image shows good visual quality and no subject difference from the original one.

Notably, the use of BFS will not reduce decoding time because the purpose of BFS is to skip many unnecessary comparisons in finding zerotrees, while in decoding algorithm no such comparisons are needed because a zerotree is known directly from the decoded bit stream.

As for the bit rate affected by BFS, there is very little. When using BFS, we have to encode the band enabled information $\tilde{B}_k$ and require a few bits. However, the benefit here is that we need not encode the nodes located on the disabled subbands, and it can save a few bits. The zerotree symbol resembles the EOB symbol in JPEG coding, while BFS tends to group many zerotrees together. Our experimental results show that the use of BFS can elevate at most 0.01 dB in PSNR for the four tested images.

Table 1 tabulates the results of our EZC coder and SPIHT coder. For a fair comparison, the results are generated by practically running these two coders on the four images. The execution file of SPIHT coder is acquired from the internet site mentioned in [11]. The results of SPIHT reported here are using the arithmetic entropy-coding version. Experimental results show that the performance of our EZC coder surpasses that of SPIHT coder over a wide range of bit rates for different images. This is especially true of the images Barbara and Goldhill. At bit rate 0.2 bpp,

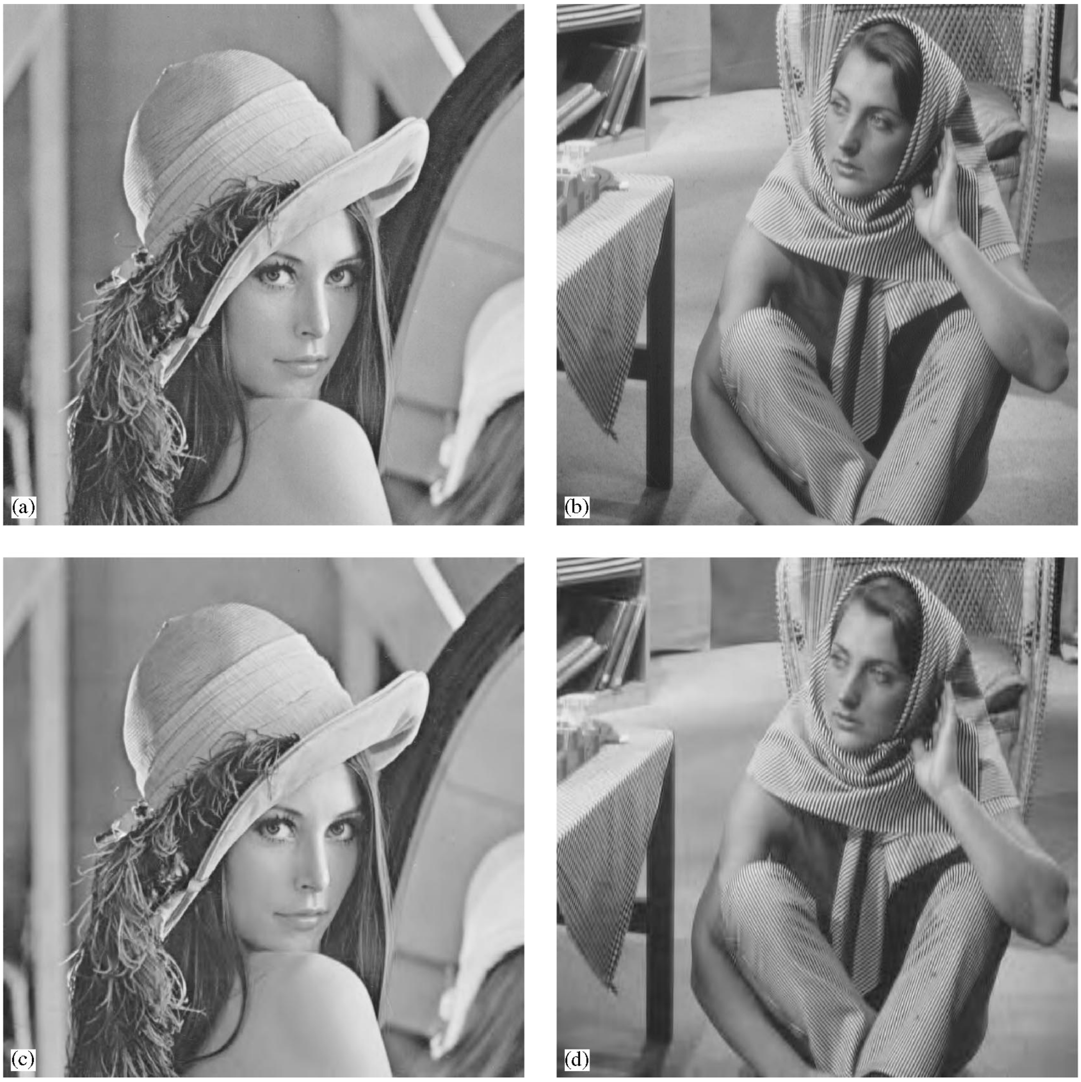


Fig. 9. Some test images and their corresponding reconstructed images at bit rate 0.5 bpp with the EZC coder: (a) original Lena; (b) original Barbara; (c) reconstructed Lena, PSNR = 37.18 dB; (d) reconstructed Barbara, PSNR = 32.63 dB.

EZC outperforms SPIHT by 0.61 dB for Barbara and 0.12 dB for Goldhill. The reason EZC performs better than SPIHT is that EZC uses AMSD to further reduce the correlation of all the transformed coefficients. Meanwhile the self-similarity across different scales is preserved, which facilitates a zerotree coding. It is necessary to point out that the coding results obtained here for both the coders are using progressive transmission ability, that is, the sets of distortion measures are acquired from the same file. In other words, the decoder reads the first bytes of the file up to the desired bit rate, performing the decoding and inverse transformation, and then compare the recovered image with the original. Some original images and their reconstructed images at bit rate 0.5 bpp (compression ratio 16) with the EZC coder are shown in Fig. 9. These reconstructed images show good visual

Table 1
Comparison of coding performance in PSNR (dB) between our EZC coder and SPIHT coder

Bit-rate (bpp)	Lena		Barbara		Goldhill		Mandrill	
	SPIHT	EZC	SPIHT	EZC	SPIHT	EZC	SPIHT	EZC
0.20	33.15	**33.30**	27.22	**27.83**	29.85	**29.97**	22.70	**22.74**
0.25	34.11	**34.15**	28.13	**28.72**	30.56	**30.79**	23.27	**23.39**
0.30	34.94	**34.97**	29.18	**29.77**	31.15	**31.36**	23.77	**23.88**
0.40	36.23	**36.32**	30.76	**31.38**	32.18	**32.29**	24.66	**24.75**
0.50	37.21	37.18	32.11	**32.63**	33.13	**33.26**	25.65	25.58
0.60	38.01	37.96	33.41	**33.94**	33.93	**34.11**	26.51	26.45
0.70	38.73	38.70	34.60	**35.06**	34.64	**34.77**	27.20	**27.27**
0.80	39.33	**39.34**	35.54	**36.05**	35.27	**35.37**	27.85	**27.90**
0.90	39.87	39.86	36.55	**36.87**	35.91	**35.94**	28.50	**28.52**
1.00	40.41	40.36	37.45	**37.73**	36.55	**36.56**	29.17	**29.18**

Table 2
Comparison of coding performance in PSNR (dB) among P0, P1 and P2. P0 represents no further partition on LH_i and HL_i bands; P1 represents partitioning LH_i and HL_i bands once; P2 represents partitioning LH_i and HL_i bands twice

Bit-rate (bpp)	Lena			Barbara		
	P0	P1	P2	P0	P1	P2
0.20	33.15	33.23	33.09	27.21	27.68	27.78
0.25	33.99	34.08	33.94	28.14	28.58	28.66
0.30	34.84	34.89	34.73	29.08	29.61	29.69
0.40	36.23	36.27	36.13	30.85	31.28	31.30
0.50	37.11	37.12	37.01	32.13	32.57	32.55
0.60	37.91	37.91	37.80	33.42	33.89	33.85
0.70	38.64	38.63	38.49	34.66	35.04	35.00
0.80	39.30	39.29	39.18	35.63	36.07	35.98
0.90	39.83	39.82	39.72	36.56	36.90	36.78
1.00	40.34	40.31	40.21	37.43	37.76	37.63

quality and no subjective difference from the original ones.

The coding results of using AMSD are also compared with that of using three fixed types of decomposition. We label P0 as no further partition on LH_i and HL_i bands, P1 as partitioning all the LH_i and HL_i bands once, and P2 as partitioning all the LH_i and HL_i bands twice. The experimental results for Lena and Barbara are shown in Table 2. In general, the results of using AMSD are better than those of the three types. In addition, the adaptive ability of AMSD can prevent that the coding performance drops. Note that the results in column P0 are slightly different from those obtained by the internet SPIHT coder (see Table 1) because they are generated by different implementation.

6. Conclusion

In this paper, we have proposed two new techniques, AMSD and BFS, to enhance a zerotree coding. AMSD provides a simple and efficient method to further partition the low-frequency contained bands, and results in an improved performance in the PSNR. At the bit rate 0.5 bpp, adding AMSD will elevate the PSNR by 0.5 dB for Barbara, 0.22 dB for Goldhill, 0.07 dB for Lena, and 0.11 dB for Mandrill. BFS gives us a way to speed up a zerotree coding. When using BFS, we can reduce the encoding time up to 1.4 s or approximate 39.3% reduction compared to the original encoding time. The proposed coder requires no training, no pre-known statistics of the coded images, and is simple and rapid in execution. In addition, the property of embedded coding is preserved which enables the progressive transmission. The experimental results are reported and compared with the results generated by SPIHT. It is shown that the proposed coder outperforms SPIHT in the PSNR over a wide range of bit rates. The interested reader can obtain EZC coder by anonymous ftp to haeshiuh.cn.nctu.edu.tw with the path pub/EZC.

It is possible to select distinct wavelet functions [14] to decompose different subbands. This is the next topic we will investigate in the near future. No matter how the wavelet function is selected, the proposed methods provide a simple way to enhance a zerotree coding.

References

[1] M. Antonini, M. Barlaud, P. Mathieu, I. Daubechies, Image coding using wavelet transform, IEEE Trans. Image Process. 1 (April 1992) 205–220.
[2] H.J. Barnard, J.H. Weber, J. Biemond, Efficient signal extension for subband/wavelet decomposition of arbitrary length signals, in: Proc. SPIE VCIP'93, Vol. 2094, November 1993, pp. 966–975.
[3] L. Chiariglione, MPEG and multimedia communications, IEEE Trans. Circuits Systems Video Technol. 7 (1) (February 1997) 5–18.
[4] E.S. Demirciler, G.H. Freeman, Multibandwidth wavelet-transform image coding using zerotrees, in: IEEE Canadian Conf. on Electrical and Computer Engineering, Vol. 1, 1995, pp. 148–151.
[5] T. Ebrahimi, MPEG-4 video verification model: A video encoding/decoding algorithm based on content representation, Signal Processing: Image Communication 9 (1997) 367–384.
[6] C. Herley, Z. Xiong, K. Ramchandran, M.T. Orchard, Joint space-frequency segmentation using balanced wavelet packet trees for least-cost image representation, IEEE Trans. Image Process. 6 (9) (September 1997) 1213–1230.
[7] R. Koenen (Ed.), Overview of the MPEG-4 Version 1 Standard, Doc. ISO/IEC JTC1/SC29/WG11 N1909, MPEG97, October 1997.
[8] S.G. Mallat, A theory for multiresolution signal decomposition: the wavelet representation, IEEE Trans. Pattern Anal. Machine Intell. 11 (7) (July 1989) 674–693.
[9] S.A. Martucci, I. Sodagar, T. Chiang, Y.-Q. Zhang, A zerotree wavelet video coder, IEEE Trans. Circuits Systems Video Techn. 7 (1) (February 1997) 109–118.
[10] K. Ramchandran, M. Vetterli, Best wavelet packet bases in a rate-distortion sense, IEEE Trans. Image Process. 2 (2) (April 1993) 160–175.
[11] A. Said, W.A. Pearlman, A new, fast, and efficient image codec based on set partitioning in hierarchical trees, IEEE Trans. Circuits Systems Video Techn. 6 (3) (June 1996) 243–250.
[12] J.M. Shapiro, Embedded image coding using zerotrees of wavelets coefficients, IEEE Trans. Signal Process. 41 (December 1993) 3445–3462.
[13] T. Sikora, The MPEG-4 video standard verification model, IEEE Trans. Circuits Systems Video Techn. 7 (1) (February 1997) 19–31.
[14] J.D. Villasenor, B. Belzer, J. Liao, Wavelet filter evaluation for image compression, IEEE Trans. Image Process. 4 (8) (August 1995) pp. 1053–1060.
[15] I.H. Witten, R.M. Neal, J.G. Cleary, Arithmetic coding for data compression, Commun. ACM 30 (June 1987) 520–540.
[16] Z. Xiong, K. Ramchandran, M.T. Orchard, Space-frequency quantization for wavelet image coding, IEEE Trans. Image Process. 6 (5) (May 1997) 677–693.