CONCURRENT ENCODING IN HIERACHICAL TREES FOR WAVELET BASED IMAGE COMPRESSION

Jing-Xin Wang, F.H. Cheng, and Alvin W.Y. Su*

*Dept. of CSIE. Chung-Hwa University, HsinChu, Taiwan
Dept. of CSIE. National Cheng-Kung University, Tainan, Taiwan

ABSTRACT

Wavelet based compression approaches becomes very popular in the last few years. In this paper, a method called CEIHT (Concurrent Encoding in Hierarchical Trees) rooted from SPIHT is proposed. This method tries to explore the relationship between adjacent resolution levels and encodes multiple coefficients with three fixed Huffman tables at the same time. Consistent coding efficiency improvement is achieved compared to the original SPIHT. Though wavelet based compression methods usually encode on a whole image, block based approaches have to be applied in many low-cost embedded applications because their memory space and computation power are usually limited. The proposed method is especially efficient in such cases. The performance of the proposed method is compared favorably to the highly acclaimed EBCOT based J2K when small coding block sizes are necessary.

1. INTRODUCTION

Developing efficient methods for encoding wavelet coefficients is an active research area. Among the existing coding techniques, EZW (Embedded Zerotree Wavelet), SPIHT (Set Partitioning in Hierarchical Trees), and EBCOT (Embedded Block Coding with Optimized Truncation) are the most famous ones [1]-[3].

Embedded Zerotree Wavelet (EZW), introduced by Shapiro [1], is based on the discrete wavelet transform. Zerotree is used to determine whether the wavelet coefficient is significant or not. It uses four symbols that are "zerotree root", "isolated zero", "positive significant", and "negative significant" to save wavelet coefficients and tree node types. This technique proposed embedded bit streams. It means that the job of bit coding can be stopped at any point and the image can be decompressed and reconstructed. Moreover, this technique has low complexly. Said and Pearlman [2] proposed the Set Partitioning in Hierarchical Trees (SPIHT) that is based on EZW. This principal concept uses three lists to replace the four symbols adopted in EZW. These three lists are "list of insignificant sets (LIS)", "list of insignificant pixels (LIP)", and "list of significant pixels (LSP)". Taubman proposed Embedded Block Coding with Optimized Truncation (EBCOT) [3] that is of J2K image coding standard. This algorithm is more efficient in coding performance compared with SPIHT. However, EBCOT has higher computational complexity than SPIHT, hence there are some papers focusing on improving the basic SPIHT algorithm. In [4], joint coding of wavelet coefficient, and significances and priority change of type-B set coding were used to improve the SPIHT. In [5], Cai and Zeng proposed a variable sorting threshold to economize the bits of the first sorting pass of LIP, in which a smaller threshold was used to improve performance in the sorting pass of LIS. In [6], the half initial threshold scheme was used.

In the sorting pass of LIS, the SPIHT algorithm uses four-bits, which can be regarded as symbols, to record whether the offspring sets and descendant sets are significant or not. Because the wavelet coefficients in the hierarchical trees are highly related, it is possible to improve the coding efficiency using this property. Based on our observation, these symbols have distinct probabilities, and the probabilities of some symbols are larger than those of others. In this paper, we use three small fixed Huffman table to when different conditions in the encoding process. It is known that EBCOT is not efficient enough when the coding tile sizes are small. The improvement of the proposed method is very significant in such situations. This is quite attractive for many embedded appliances whose computation power and memory spaces are both limited.

The rest of the paper is organized as follow. Section 2 briefly introduces the SPIHT algorithm. The CEIHT algorithm and the difference between SPIHT and CEIHT are also provided. Simulation results are shown in section 3. Conclusions are given in Section 4.

2. CONCURRENT ENCODING IN HIERARCHICAL TREES

The SPIHT method explores the self-similarity of wavelet coefficients across different resolution levels and provides an effective and computationally simple coding

0-7803-8554-3/04/$20.00 ©2004 IEEE.

procedure [2]. It contains three coding processes: initialization pass, sorting pass, and refinement pass. Three ordered lists are used during the coding process: LIS, LIP, and LSP. In the initialization pass, it outputs the maximum threshold and initiating entries of LIP and LIS. The maximum threshold is defined as:

$$n = \lfloor \log(\max_{(i,j)} \{ |C_{i,j}| \}) \rfloor \quad (1)$$

$C_{i,j}$ is called transform coefficient at coordinate (i, j).

The sorting pass tests each entry at a coordinate (i, j) of LIP and LIS whether it is significant or not. A significant entry will be attached to the end of LSP and is removed from LIS or LIP at the same time. Significance test is made by:

$$S_n(\tau) = \begin{cases} 1, & \max_{(i,j)\in\tau} \{ |C_{i,j}| \} \geq 2^n \\ 0, & otherwise \end{cases} \quad (2)$$

where τ is a set of coordinates. The information of the above significance test is outputted. The following sets of coordinates are defined for presenting the algorithm:

- $O(i, j)$: set of coordinates of all offspring of node (i,j).
- $D(i, j)$: set of coordinates of all descendants of the node (i,j).
- $L(i, j)$: $D(i, j) – O(i,j)$.

To differentiate the entries in LIS, the type-A and type-B ones are used. If it represents $D(i,j)$, it is of type-A. If it represents only $L(i,j)$, it is of type-B. Finally, the refinement pass outputs the n-th most significant bit of $|c_{i,j}|$ for entry-(i,j) within the LSP. The total encoding flow will not be reiterated. Those who are interested in it should refer to [2]. We will only explain the relationship between SPIHT and CEIHT here.

When entry-(i,j) in the LIS is of type-A, the corresponding four offspring are tested with the current threshold if the $S_n(D(i,j))$= 1. They are moved to LIP or LSP depending on whether it is significant or not. Since there are four direct offspring, we need four bits to represent this information in SPIHT codec. However, it is found that the probability model of these four symbols is skewed and the entropy coding can be used to save some bits. Let No_SO be the number of significant offspring for a newly found significant entry in LIS. If No_SO is equal to 3, there are three significant direct offspring stemmed from a parent node. Table 1 shows the statistics for different No_SO's. The table is derived by using four 512×512 images. They are Lena, Baboon, Peppers, and Barbara. A five-level dyadic DWT decomposition with the bi-orthogonal 9/7-filter suggested in J2K is used [7]. Based on the information theory, entropy of random events can be calculated as follow:

$$H = -\sum P(A_i) \log_2 P(A_i) \ , \quad (3)$$

where A_i represents the i-th outcome and $P(A_i)$ represents its probability. The probabilities for a parent node with different numbers of significant offspring are different. When No_SO is equal to 3, four bits are outputted in original SPIHT codec. They are either "0111", "1011", "1101" or "1110". Based on Table 1, some bits can be saved. By calculating the entropy of Table 1 with Eq. (3), it is approximately 3.60 when using 4-bit symbols for No_SO. A simple Huffman table can be used to encode these 4-bit symbols.

When entry-(i,j) in LIS is of type-B and $S_n(L(i,j))$=1, its four offspring are added to the end of LIS as type-A entries and entry-(i,j) is removed from LIS. For each offspring in $O(i,j)$ denoted by entry-(k,l), continue the sorting pass in the LIS until we meet entry-(k,l). Because entry-(k,l) is of type-A, $S_n(D(k,l))$ has to be outputted. Because there are four offspring to be processed, we need four more bits to provide the information of $S_n(D(k,l))$. All four offspring of entry-(i,j) are spatially related. Similar to what we do in Table 1, Table 2 shows the statistics for the number of significant descendants (No_SD) of entry-(i,j). It is noted that No_SO and No_SD have different meanings. The images used to derive Table 2 are the same as the ones to derive Table 1. The entropy using 4-bit symbols for No_SD is equal to 3.54. Therefore, a simple Huffman table can be used to improve the coding efficiency. By using Table 1 and Table 2, the combined entropy is 7.12.

For every type-A entry-(i,j) in LIS, if $S_n(D(i,j))$ = 1 and $S_n(L(i,j))$ = 1, then entry-(i,j) is moved from LIS and its four offspring are added to the end of LIS as type-A entries. In this case, it needs four bits to record the information whether its four offspring are significant pixels or not because these four pixels will be moved to LIP or LSP. The situation is the same as Table 1. We need another four bits to record the information whether its four offspring are significant sets or not because the four offspring are all type-A entries in LIS. The situation is the same as Table 2. Table 3 shows only part of the new statistics for (No_SO, No_SD) if these eight bits are to be encoded at the same time. We call the proposed method as Concurrent Encoding In Hierarchical Trees (CEIHT). The new entropy using 8-bit symbols is 6.81. This is less than that of the case if CEIHT isn't used. The difference between CEIDT and SPIHT algorithms lies in the sorting pass when processing each type-A entry-(i,j) in LIS. To implement CEIHT, type-C entries in LIS are added. The implementation of the initialization pass and the refinement pass are identical to original SPIHT. Fig.1 shows the sorting pass of the CEIHT algorithm.

3. SIMULATION RESULTS

512×512 8 bpp standard test images are used: Lena, Baboon, Boat, and Butterfly. Boat and Butterfly are not used in generating the Huffman tables. Bi-orthogonal 9/7 filters used in J2K are employed [7]. DWT based image compression methods usually encode on a whole image because the coding efficiency is higher in this case. This usually requires large memory space and high computation power in order to have higher throughput. This is certainly not suitable for some embedded applications and block based approaches are usually preferred. Hence, three different block sizes are used: 512×512, 64×64, and 32×32. No boundary overlap and de-blocking filter are used. The Original SPIHT and J2K codec are also used under identical simulation configurations for direct comparison. SPIHT and J2K codec's are downloaded at [8],[9]. Fig. 2 shows the rate-distortion curves In Fig.2, 64/5 means that block size is 64×64 and the 5-level dyadic DWT decomposition is applied. No other entropy coding method is used. The proposed CEIHT outperforms SPIHT by 0.1~0.3dB when 512×512 blocks are used. When smaller block sizes are used, the improvement of CEIHT can be as large as 5dB! When 512×512 blocks are used, EBCOT based J2K is better. When coding blocks are small, CEIHT starts to outperform J2K, too. CEIHT is also better than J2K at many lower bit-rate conditions.

4. CONCLUSION

A new wavelet based image compression technique called the Concurrent Encoding In Hierarchical Trees (CEIHT) are presented. CEIHT is modified from SPIHT. The proposed algorithm explores the relationship of the hierarchical trees and encodes multiple symbols at the same time. Three simple Huffman tables are used and better coding efficiency is achieved. The increased complexity is negligible. CEIHT is very efficient when block based coding schemes have to be used. This feature is attractive because memory and computation power are usually limited in most low-cost embedded appliances. The coding efficiency may be improved if arithmetic coding methods are also applied though it was reported that arithmetic coding is not as efficient in low bit-rate conditions [2] [10]. It is possible to consider the proposed approach on joint color image compression because the three color components are usually highly related [11].

5. REFERENCES

[1] J. M. Shapiro,, "Embedded image coding using zeroes of wavelet coefficients," *IEEE Trans. Signal Processing*, vol. 41, pp. 3445-3462, Dec. 1993.

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

[3] D. Taubman, "High Performance Scalable Image Compression with EBCOT," *IEEE Trans. Image and Processing,* vol. 9, no. 7, pp. 1158-1170, July 2000.

[4] U. Bayazit and W.A. Pearlman, "Algorithm modifications to SPIHT," Int. Conf. on Image Processing, ICIP 2001, Thessaloniki, Greece, pp. 800-803, 2001.

[5] H. Cai and B. Zeng, "A new SPIHT algorithm based on variable sorting thresholds," *Circuits and Systems, 2001. ISCAS 2001. The 2001 IEEE International Symposium on*, vol. 5, pp. 231-234, May 2001.

[6] X-W. Yin, M. Fleury, and A.C. Downton, "Reduced bit rate uniform quantization for SPIHT encoding," Electronices Letters, vol. 39, pp. 902-903, Jun 2003.

[7] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies, "Image coding using wavelet transform," *IEEE Trans. Image Processing,* vol. 1, pp.205-550, Apr. 1992.

[8] URL : http://www.kakadusoftware.com/

[9] URL : http://www.cipr.rpi.edu/research/SPIHT/

[10] I. H. Witten, R. M. Neal, and J. G. Cleary, "Arithmetic coding for data compression," Commun ACM, vol. 30, pp.520-540, June 1987.

[11] K. Shen and E. J. Delp, "Color image compression using an embedded rate scalable approach," *in Proc. IEEE Int. Conf. Image Processing*, Santa Barbara, CA, vol. 3, pp. 34-37, Oct. 1997.

No_SO	Lena	Baboon	Peppers	Barbara	Avg.
0	0.108	0.136	0.201	0.174	0.155
1	0.562	0.493	0.532	0.421	0.502
2	0.255	0.277	0.212	0.210	0.239
3	0.064	0.082	0.046	0.130	0.080
4	0.008	0.010	0.006	0.062	0.02

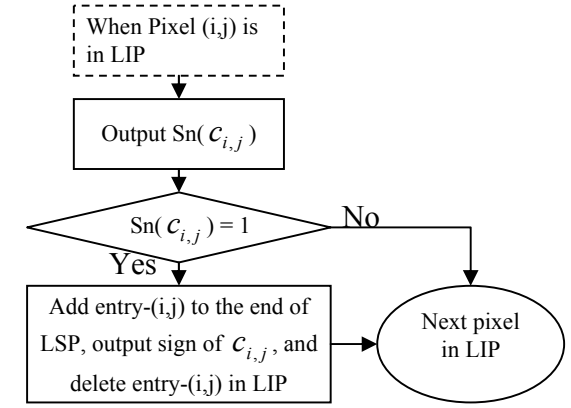
Table.1 Statistics of number of significant off-springs

No_SD	Lena	Baboon	Peppers	Barbara	Avg.
1	0.693	0.544	0.655	0.487	0.595
2	0.215	0.278	0.229	0.258	0.245
3	0.072	0.128	0.084	0.136	0.105
4	0.019	0.048	0.030	0.117	0.053

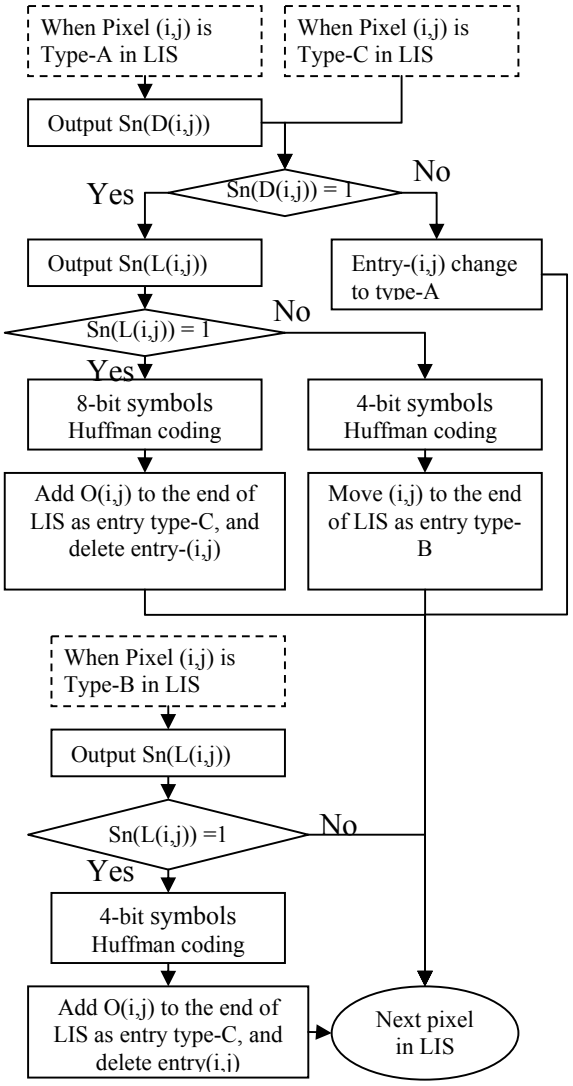
Table.2 Statistics of number of significant descendants

(No_SO, No_SD)	Lena	Baboon	Peppers	Barbara	Avg.
(0,1)	0.267	0.155	0.304	0.232	0.239
(1,1)	0.244	0.213	0.228	0.150	0.209
(2,1)	0.139	0.136	0.098	0.072	0.110
(1,2)	0.080	0.097	0.088	0.072	0.084
…	...	…	…	…	…
(4,4)	0.0003	0.0007	0.0008	0.0017	0.0009

Table.3 Statistics of number of significant off-springs and significant descendants

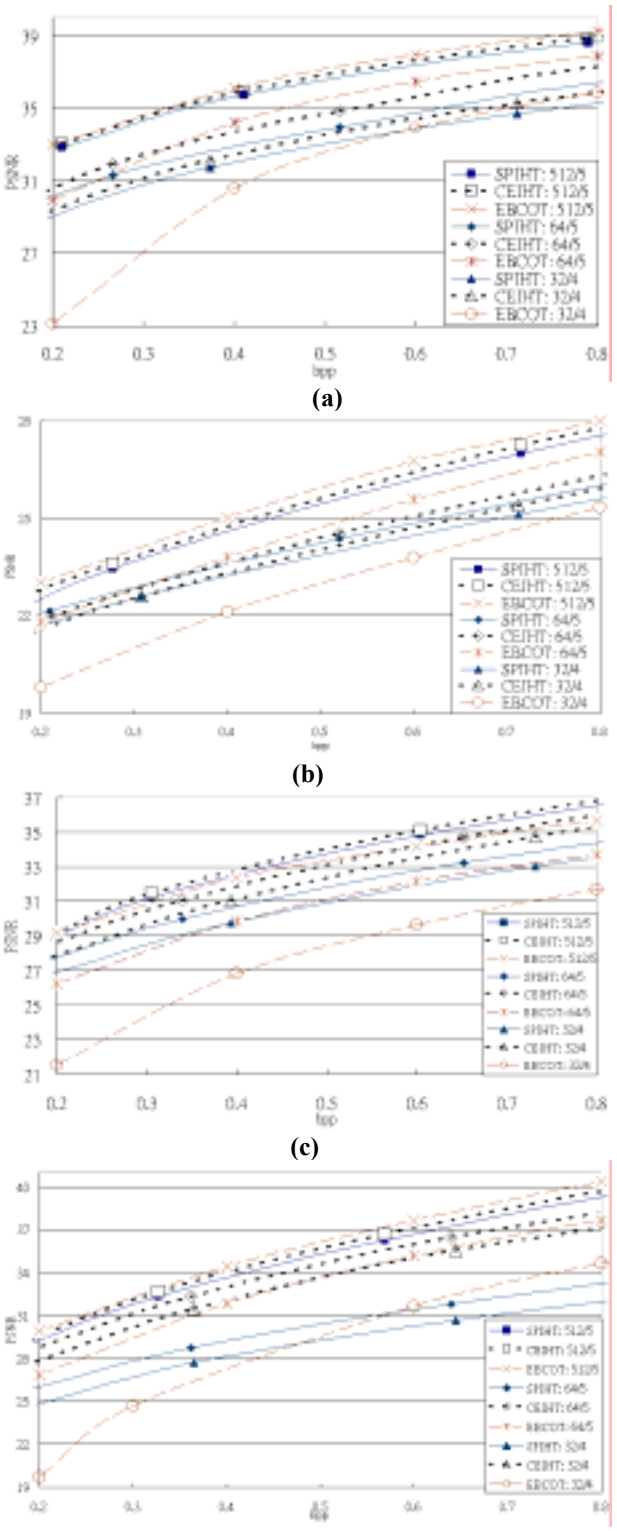


(a) LIP Part



(b) LIS Part

Fig.1 CEIHT sorting pass : (a)LIP ;(b)LIS



(a) **(b)** **(c)** **(d)**

Fig. 2. Comparisons of SPIHT, CEIDT and J2K (a)Lena(b)Barbara(c)Boat(d)Butterfly