WAVELET BASED EMBEDDED IMAGE CODING USING UNIFIED ZERO-BLOCK-ZERO-TREE APPROACH

Athar Ali Moinuddin and Ekram Khan

Department of Electronics Engineering
Aligarh Muslim University
Aligarh, 202002, India

ABSTRACT

In this paper, we propose a unified approach of exploiting both inter- and intra- band correlations among the wavelet coefficients in a single algorithm. Most of the existing wavelet based image coding techniques are either based on zero-tree approach or zero-block (quad-tree) approach. The zero-tree based coders exploit only inter-band correlations whereas zero-block based coders exploit only intra-band correlations among the wavelet coefficients. Thus this work is an attempt to fuse the features of both block- and tree-based coding algorithms into a single algorithm. The proposed algorithm generates fully embedded bit stream. Simulation results show that the proposed algorithm outperforms existing wavelet based image coders. Additionally, the memory requirements due to the auxiliary lists is also reduced as compared to the zero-tree based coders.

1. INTRODUCTION

The success of wavelet-based image coding algorithms is mainly due to the exploitation of energy clustering property of wavelet transform. Over the years, a number of successful wavelet-based image coding algorithms have been proposed. These algorithms can be broadly classified into two groups: zero-tree and zero-block based algorithms. The zero-tree coding algorithms exploit similarities among the coefficients across the subbands. Since the pioneer work of Shapiro's EZW (Embedded Zero-tree Wavelet) algorithm [1], many of its variants are being developed. The Set Partitioning In Hierarchical Trees (SPIHT) [2] is the most popular and efficient variant of EZW. An improved version of SPIHT known as Virtual SPIHT (VSPIHT) algorithm is proposed in [3]. The coding efficiency of these algorithms are mainly due to the exploitation of inter-band redundancies by grouping the insignificant coefficients in trees growing exponentially across the scale and by coding them with 'zero' symbol (zero-trees). The SPIHT (and also VSPIHT) algorithm has excellent rate-distortion performance with low computational complexity, while generating embedded bit stream. Its small dependency on entropy coding (usually 0.2-0.5 dB) enables one to avoid the use of arithmetic coding.

On the other hand, zero-block coding algorithms exploit only intra-band correlations in the form of zero-blocks. They work by dividing the transformed image into contiguous blocks and performing the significance test on the individual blocks. A zero-block is then a block with no significant coefficients with respect to a given threshold. The well-known algorithms in this category include Set Partitioning Embedded bloCK (SPECK) [4], SBPH [5], Embedded Block Coding with Optimized Truncation (EBCOT) [6] and Embedded Zero Block Coding (EZBC) [7][8]. SPECK is an efficient zero-block based embedded image coder. SBPH is a low complexity version of SPECK incorporated into JPEG2000 framework. It is faster but sacrifices coding efficiency. EBCOT is used in JPEG2000 standard and has improved performance over SPECK, but it is highly complex. EZBC is based on quad-tree partitioning of subbands. It is a higher complexity variant of SPECK and has superior performance over SPECK and EBCOT. The major performance gains of EBCOT and EZBC are due to the use of complicated context based arithmetic coding.

In this paper, we propose an efficient image coding algorithm that combines the feature of both zero-tree and zero-block coding algorithms. It is based on partitioning a wavelet-transformed image into coefficient blocks and using spatial orientation trees (SOT) of blocks having roots in top-most (LL) subband. We have used the SOT of blocks (called block-tree) as opposed to the SOT of coefficients in SPIHT. In a block-tree, significant blocks are found using tree partitioning concept of SPIHT, whereas significant coefficients within each significant block are found using quad-tree partitioning of SPECK. A significant block-tree is recursively partitioned (with combined tree and block partitioning) until significant coefficients are found. The proposed algorithm is termed as **Wavelet Block Tree Coding** (WBTC) due to obvious reasons. Earlier attempts to use the feature of both zero-tree and zero-block coding algorithms are made in [9] and [10]. The algorithm in [9] uses a single list and the emphasis was more on the memory reduction rather than coding efficiency, where as in the proposed algorithm, we have targeted the improved coding

efficiency and memory reduction is an added advantage. In [10], authors have divided transformed subbands into spatial and subband blocks, which were coded independently by SPIHT and SPECK. The drawback of this algorithm is that it no longer retains the progressiveness property and requires rate-distortion optimization. Whereas the proposed coder generates fully embedded bit stream without any rate-distortion optimizer.

The rest of the paper is organized as follows. In Section 2, the SPIHT and SPECK algorithms are briefly reviewed. The proposed algorithm is described in Section 3. Simulation results and discussions are presented in Section 4 and finally the paper is concluded in Section 5.

2. REVIEW OF SPIHT AND SPECK ALGORITHMS

Here we briefly review the SPIHT [2] and SPECK [4] algorithms highlighting their important features. These algorithms are the representatives of zero-tree and zero-block coding algorithms respectively. In SPIHT [2], which is a state-of-the-art zero-tree coding algorithm, significant information is stored in three ordered lists: a list of insignificant pixels (LIP), a list of insignificant sets (LIS), and a list of significant pixels (LSP). At the initialization step, the pixels in the LL-subband are added to LIP, and those with descendents also are added to LIS as type 'A' entries. The LSP starts as an empty list.

The algorithm starts with the most significant bit plane and proceeds toward the finest resolution. At every bit plane, the encoder goes through the three lists in order, starting with LIP followed by LIS and then LSP. For each pixel in LIP, one bit is used to describe its significance. If the pixel is not significance, it remains in LIP and no more bits will be generated; otherwise, the sign bit is produced and the pixel is moved to LSP. Similarly, each set in LIS requires one bit for significance information. Insignificant sets remain in LIS while significant set will be partitioned into subsets. A significant type 'A' set will be partitioned into four offsprings pixels and a type 'B' set (grand-descendants); the type 'B' set is added to the end of LIS while four pixels are immediately examined for their significance and added to LIP and LSP accordingly. A significant type 'B' set will be partitioned into four type 'A' sets (with offspring pixels as corresponding root nodes) and are added to the end of LIS. Since all newly generated insignificant sets are added to the end of LIS, they will be processed in the same manner at the same resolution until each significant subset has exactly one coefficient. Finally, each pixel in LSP, except those just added at current bit plane, is refined with one bit. The algorithm then repeats the above procedure for the next resolution.

On the other hand, zero-block coding algorithms, such as SPECK [4] and EZBC [7] divides the transformed image into contiguous blocks and performs the significance test on the individual blocks. If a block is significant then it is partitioned into four equal size subblocks (quad-tree partitioning). Each subblocks is then individually tested for its significance. A significant subblock again uses a quad-tree partitioning. In this way a significant block is recursively portioned into subblocks. Recursion ends when significant coefficients are found. The advantage of block encoding is that small blocks, representing high frequency areas, are coded separately from larger areas with low spatial-frequency content.

It is apparent that zero-tree coding algorithms like SPIHT [2] do not exploit intra-band correlations where as zero-block coding algorithms like SPECK [4] and EZBC [7][8] do not exploit inter-band correlations. Also, at very low encoding bit rates, SPIHT generates a lot of clustered zero-trees. This is because at very low encoding bit rates, coding terminates after few early passes when the threshold is very high, so a vast majority of wavelet coefficients fall below the threshold. This reduces the coding efficiency of SPIHT at very low bit rate. The virtual SPIHT algorithm combines the clustered zero-trees for very low bit rate video coding [3]. The contribution of this work is to combine the feature of both zero-block and zero-tree coding algorithms to get improved coding efficiency.

3. PROPOSED ALGORITHM

Consider an image χ of size $M \times N$ that after N_d levels of wavelet transformation exhibits a pyramidal subband structure. The transformed image is represented by an indexed set of transformed coefficients $\{c_{i,j}\}$ located at the i^{th} row and the j^{th} column. The coefficients are grouped together in blocks of size $m \times n$ and then block-trees are formed with roots in the LL-subband. A block-tree is a tree of all descendent blocks of a root block. This approach has three distinct advantages over SPIHT. First, it will combine many clustered zero-trees of the SPIHT, which are going to occur in the early passes, thus creating longer zero-trees. Second, intra-band correlations can also be partially exploited. Third, because of block-based, memory requirement for storing the lists will be less as compared to pixel-based techniques.

Except for the lowest and the highest resolution band, each block is having four offspring blocks that correspond to the same spatial orientation in higher frequency subbands. In the LL-subband, out of each group of 2×2 blocks, one (top-left) block has no descendents, and each of the other three blocks has four offspring blocks in high frequency subbands of their corresponding orientations. By creating block-tree, many of the SPIHT's SOTs are combined into a single and longer SOT. In particular, for a block size of 2×2, four SOTs of SPIHT are combined into a single WBTC's SOT.

Significance information is stored in three ordered lists: a list of insignificant blocks (LIB), a list of insignificant

block sets (LIBS), and a list of significant pixels (LSP). At the initialization step, the blocks in the LL-subband are added to LIB, and those with the descendents also are added to LIBS as type 'A' entries. The LSP starts as an empty list.

Like SPIHT and SPECK, WBTC is also a bit-plane based coding algorithm and comprise two main stages, sorting and refinement passes, within each bit-plane. The coding process starts with the most significant bit-plane and proceeds toward the finest resolution. During the sorting pass, the encoder first traverses through LIB, testing the significance of a block against the current threshold. For each block in LIB, one bit is used to describe its significance. If the block is not significant, then it is a zero-block and a '0' is send, it remains in LIB and no more bits will be generated. Here, insignificant information of $m \times n$ individual coefficients is conveyed using a single '0', where as in SPIHT it will generate $m \times n$ '0' bits. This is how WBTC exploits intra-band correlation partially. Otherwise, if the block is significant then it is a non-zero block and a '1' is sent. A significant block is partitioned into four adjacent blocks (quad-tree partitioning). The division operation is repeated recursively until no further division is needed or the smallest possible block size (individual coefficient) is attained. At this stage four coefficients and their significant is tested individually. If a coefficient is insignificant, then a '0' is send and it is moved to LIB as single coefficient block. On the other hand if a coefficient is significant, then a '1' is send and its sign bit is also coded and the coefficient is moved to the LSP. After testing all the four individual coefficients in the block, the current block is deleted from LIB. The encoder then examines the LIBS and performs significance test on each set. Insignificance sets remain in LIBS while significant ones are partitioned into subsets. A significant type 'A' set will be partitioned into a type 'B' set and four offspring blocks. The type 'B' set is added to the end of LIBS while the four blocks are immediately examined for significance as is done in LIB. Here also a zero-block will save the bits. A significant type 'B' set will be partitioned into four type 'A' sets; all of them are added to the end of LIBS. Since all newly generated insignificant sets are added to the end of LIBS, they will be processed in the same manner at the same threshold until each one of them is examined. After each sorting pass, each coefficient in LSP, except those just added at this bit plane, is refined with one bit. The algorithm then repeats the above procedure by decreasing the current threshold by a factor of two until the desired bit rate is achieved. It should be noted that for a block size of 1×1 (single pixel), WBTC leads to the SPIHT algorithm. Thus, SPIHT is a special case of the WBTC algorithm.

4. SIMULATION RESULTS

The rate-distortion (R-D) performance of the proposed algorithm is evaluated on three classical test images, *Lena, Goldhill* and *Barbara*. Each of these images are 256 gray-level image (8 bits/pixel) and of size 512×512. A 5-level wavelet decomposition using biorthogonal 9/7 tap filter is used. Test images are encoded at maximum rate of 1.0 bit/pixel (bpp) and are decoded at different bit rates from the same embedded bitstream. The objective quality of the reconstructed image is measured in terms of the *Peak-Signal-to-Noise-Ratio (PSNR)*, defined as

$$PSNR = 10 \log_{10} \frac{255^2}{mse} \quad (1)$$

where the mean square error, *mse* is calculated as follows:

$$mse = \frac{1}{n} \sum_{i=0}^{n-1} (x_i - x_i')^2 \quad (2)$$

where n is the number of pixels, and x_i and x_i' are the original and reconstructed pixels values at the same location.

The performance of the proposed algorithm is compared with SPIHT, SPECK and EZBC. For WBTC, we have taken block size of 2×2. The coding results are summarized in Tables 1, Table 2 and Table 3 for *Lena*, *Barbara* and *Goldhill* test mages respectively. All the results shown in these tables are for the binary coded (without arithmetic coding) version of the corresponding algorithm. The results for SPECK and EZBC are taken from Hsiang's thesis [8]. The best result at each bit rate is shown in the bold face. The results for the binary coded version of SPECK are only available for 0.25 and 0.5 bpp for all the test images. It can be seen from these tables that proposed WBTC algorithm has the best performance for *Lena* and *Goldhill* images whereas EZBC has better performance for *Barbara* image. It can be seen from Table 1 that WBTC outperforms SPECK by approximately 0.5 dB, SPIHT by 0.1-0.25 dB and EZBC by 0.02-0.16 dB. Similarly, for Barbara image in Table 2, WBTC has better performance than SPIHT and SPECK, and has slightly inferior but comparable performance to that of EZBC. One of the reasons for better performance of EZBC for Barabra image is that it has many high frequency components, and block based coder like EZBC exploit intra-band correlation in the better way. We believe that by increasing the block size in WBTC, we can improve the performance of WBTC for the Barbara image also. Also, the results of Goldhill image in Table 3 show that WBTC consistently outperform SPIHT and SPECK and slightly better than that of EZBC. It should be noted that performance of WBTC is much better compared to other coders at lower bit rates. The reason for this is that at lower bit rates, most of the coefficients are likely to be insignificant and WBTC combines a large area of insignificant coefficients together and hence increases the coding efficiency. Consistently better performance of WBTC as compared to SPIHT (zero-tree) and SPECK (zero-block) is attributed to the exploitation of both inter- and intra-band correlations of the wavelet coefficients.

Additionally, WBTC algorithm requires lesser memory to store the entries of the linked list as compared to that of SPIHT. Since instead of storing pixels as the element in the lists (as in SPIHT), WBTC stores block addresses that are always lesser than the number of pixels. In initialization phase, LIB and LIBS of WBTC contain block addresses, whereas LIP and LIS of SPIHT contain addresses of pixels from LL-subband. Thus number of initial entries in the lists of WBTC will always be less than the number of entries in the lists of SPIHT. For 2x2 initial block size, WBTC requires 75% lesser memory as compared to that of SPIHT.

bpp	SPECK	SPIHT	EZBC	WBTC
0.03125	-	25.36	-	**25.61**
0.0625	-	28.01	28.05	**28.21**
0.125	-	30.72	30.81	**30.91**
0.25	33.37	33.70	33.80	**33.82**
0.5	36.49	36.85	36.91	**36.95**

Table 1: PSNR (in dB) comparison of WBTC with SPECK, SPIHT and EZBC at various bit rates for test image *Lena.*

bpp	SPECK	SPIHT	EZBC	WBTC
0.03125	-	21.98	-	**22.11**
0.0625	-	23.12	**23.32**	23.28
0.125	-	24.47	**24.94**	24.81
0.25	26.96	27.22	**27.70**	**27.70**
0.5	30.79	30.94	**31.33**	31.21

Table 2: PSNR (in dB) comparison of WBTC with SPECK, SPIHT and EZBC at various bit rates for test image *Barbara.*

bpp	SPECK	SPIHT	EZBC	WBTC
0.03125	-	24.52	-	**24.66**
0.0625	-	26.54	26.61	**26.68**
0.125	-	28.27	28.32	**28.39**
0.25	30.21	30.22	**30.29**	**30.29**
0.5	32.58	32.71	32.83	**32.84**

Table 3: PSNR (in dB) comparison of WBTC with SPECK, SPIHT and EZBC at various bit rates for test image *Goldhill.*

5. CONCLUSIONS

In this paper, we have proposed a novel algorithm that fuses the feature of both zero-tree and zero-block based algorithms in a single algorithm. It is observed that the proposed coding algorithm has better performance than algorithms based on either zero-tree or zero-block concepts only. It can also be concluded that in wavelet based compression techniques, exploitation of both inter- and intra-band correlation is important. The proposed coding algorithm is especially attractive for very low bit-rate applications. Also, being block based, it reduces the memory requirement at the encoder and decoder.

6. REFERENCES

[1] J. Shapiro, "Embedded image coding using zerotree of wavelet coefficients," *IEEE Trans. Signal Processing*, Vol. 41, pp. 3445-3462, December 1993.

[2] A. Said, and W. A. Pearlman, "A new, fast and efficient image codec based on set partitioning in hierarchical trees", *IEEE Trans. Circuits and Systems for Video Technology*, Vol. 6, pp. 243-250, June 1996.

[3] E. Khan, and M. Ghanbari, "Very low bit rate video coding using Virtual SPIHT", *IEE Electronics Letters*, Vol. 1, pp. 40-42, Jan. 2001.

[4] W. A. Pearlman, A. Islam, N. Nagaraj, and A. Said, "Efficient, low-complexity image coding with a set partitioning embedded block coder", *IEEE Trans. Circuits and Systems for Video Technology*, Vol. 14, pp. 1219-1235, November 2004.

[5] C. Chrysafis, A. Said, A. Drukarev, A. Islam, and W. A. Pearlman, "SBPH- a low complexity wavelet coder," IEEE *Int. Conf. on Acous. Speech and Sig. Proc. (ICASSP'00),* Vol. IV, pp.2035-2038, June 2000.

[6] D. Taubman, "High performance scalable image compression with EBCOT", *IEEE Trans. Image Processing*, Vol. 9, pp. 1158-1170, July 2000.

[7] S. –T. Hsiang, and J. W. Woods, "Embedded image coding using zeroblocks of subband/wavelet coefficient and context modeling", *IEEE Int. Symposium on Circuit and Systems (ISCAS'00)*, Vol. III, pp. 662-665, May 2000.

[8] S. –T. Hsiang, "Highly scalable subband/wavelet image and video coding", *Ph.D. Thesis*, Rensselaer Polytechnic Institute, May 2002.

[9] H. Arora, P. Singh. E. Khan, and F. Ghani, "Memory efficient image coding with embedded zero block-tree coder", Proc. *IEEE Int. Conference on Multimedia and Expo2004 (ICME'04)*, Vol. 1, pp. 679-682, May 2004.

[10] F. W. Wheeler and W. A. Pearlman, "Combined spatial and subband block coding of images", *IEEE Int. Conference on Image Processing (ICIP'00)*, Vol. III, pp. 861-864, September 2000.