

A Unification Framework for Tree *and* Block Wavelet Encoders

X.-W. Yin, M. Fleury, and A. C. Downton

Department of Electronic Systems Engineering, University of Essex,
Wivenhoe Park, Colchester, CO4 3SQ, United Kingdom

Abstract

Wavelet transform coefficient encoders are broadly categorized as depending on correlations: across sub-bands through a zerotree, or within sub-bands through a block quad-tree. This paper proposes a unification framework that allows a suitable algorithm to be crafted according to need. An example block-tree algorithm is described in detail, and results show that the performance closely approaches a state-of-the-art encoder, without needing the complexity of arithmetic coding.

1. Introduction

The wavelet transform [5], using the common 9-7 tap biorthogonal filter, produces a pyramidal subband structure. Due to the localized nature of the wavelet transform, wavelet coefficients are correlated *both* spatially across subbands and in frequency within subbands. Therefore, the possibility exists of introducing compression by removing these correlations. This paper sets out to remedy an apparently surprising omission: there does not appear to exist an algorithm that exploits both types of correlation.

The first-generation algorithm, zero-tree encoding [8], took advantage of spatial coefficient magnitude correlation: if a parent is insignificant, then its children are likely to be insignificant. Second-generation improvements to this algorithm tended to bifurcate between hierarchical tree encoding [7], and wavelet sub-band block-based methods [4], using quad-trees. Unification of group testing mechanisms [1] may be considered as a third-generation improvement, and certainly the group testing concept has aided the understanding of these algorithms. [9] involves both a hierarchical tree phase followed by a block phase (to reduce memory usage), though there is no fusion of the algorithms. It was realized by us (and no doubt by others) that the two types of encoder are in fact, parameterized versions of the same algorithm. Following from this insight, a way of tuning an encoder to a required performance by varying block size has evolved.

2. Tree and Block Encoders

Figure 1 shows a three-level zero-tree with parent at sub-band HL_2 , as well as a pixel-level tree with parent at sub-band LH_1 (zero trees can occur at various granularities). Alternatively, (in)significant coefficients tend to be clustered within a subband. In Figure 2, only one of the typical subband's quadrants has significant pixels. This is first encoded by '0100' to identify the position of that quadrant. Recursively splitting the significant quadrant, results in the bit stream sequence illustrated in Figure 2.

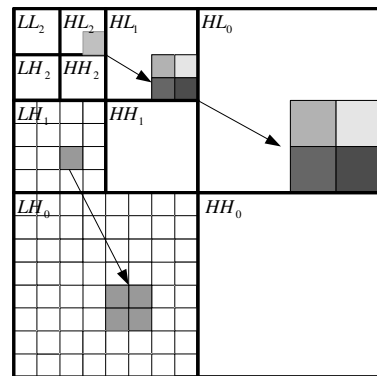


Figure 1. Zerotree Encoding

We consider an example of each of these types of encoder:

SPIHT [7] descends hierarchically from a parent pixel in the coarsest subband, to progressively quantize coefficient magnitude sets, bit-plane by bit-plane. All the descendants of a parent pixel at the coarsest layer might be collectively encoded by a single symbol as a zerotree, *i.e.* no member is significant at that quantization level. In SPIHT, trees may be further partitioned into three sets: 1) the root of the tree, 2) the immediate descendants, and 3) the remaining descendants. This is because the statistical characteristics of the coefficient sets differ as the tree is ascended. In fact, the partition-

ing is recursive, once the pixel at the coarsest level has been encoded. Whenever the finest level or a zerotree is encountered then the recursion halts.

SPECK [4] is an example of block set encoding, whereby a transformed image is recursively partitioned into sub-blocks, achieving compression by single-symbol encoding of insignificant blocks. 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. In Figure 3, after initial blocking of the pixelated image (with some significant pixels shaded in), set S_0 is initially block encoded in image I_0 . The block encoding order then proceeds, I_1 through the remaining subbands at the same level, $S_{1,2,3}$, before turning to the remaining subbands, $S_{4,5,6,\dots}$, in I_2 .

It is apparent that SPIHT does not exploit correlations between adjacent sets, whereas SPECK does not exploit spatial correlations in subbands at different levels of the fine-to-coarse pyramid. The contribution of this paper is to propose a unifying framework for the encoding of transform coefficients by these low-complexity methods. One advantage of low-complexity is removal of delay: in [6] tests are reported showing SPECK compared to JPEG2000's VM 3.2A to be between 4.6 and 15.7 times faster in encoding, and 8.1 to 12.1 times faster in decoding on the average over a set of four images and a set of four rates, 0.25, 0.50, 1.0 and 2.0 bits/pixel.

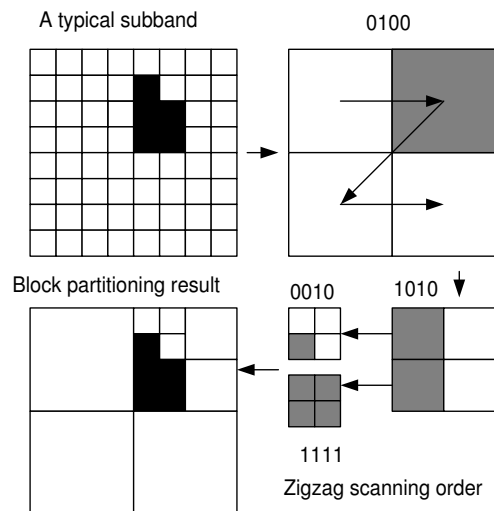


Figure 2. Zero Block Encoding

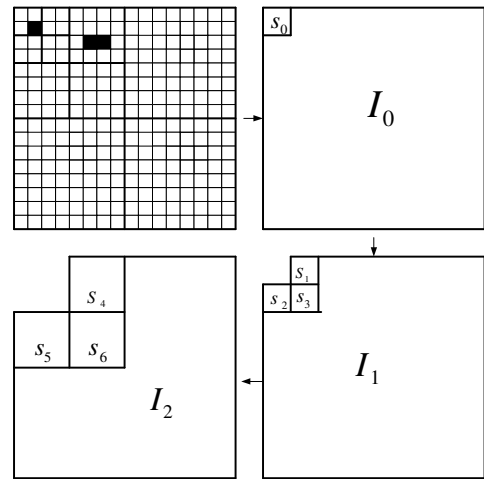


Figure 3. SPECK partitioning method

3. Unification of Zero-tree and Block Encoding

In this section, SPIHT and SPECK, representing the two forms of coefficient coder are integrated in a unified structure. In Figure 4, the original image pixels have been partitioned into blocks, which align with the subband structure of the coefficient image. For the image of Figure 4, three block trees are formed by identifying the head of the tree as the coarsest subband. Having identified a significant block tree, then individual blocks within the tree are encoded in the manner of block encoding. However, there is no necessity to identify the head of the tree as the coarsest subband, as other block sizes can be chosen. In fact, the block size can become a parameter, as the results of Section 4 explore. More formally, the unification al-

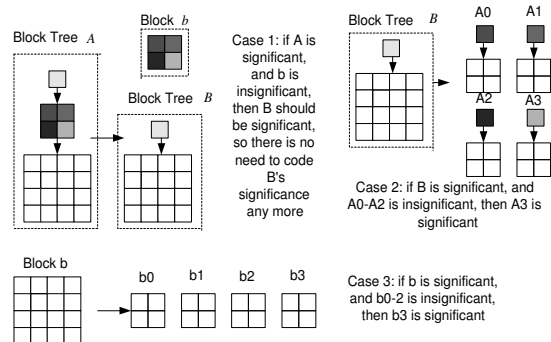


Figure 4. Conditional block encoding

gorithm is described as follows. Given an image, χ , of size $p \times q$ with $p = 2^x$, $q = 2^y$ for some integers x, y , which has been transformed to level N , then the top-left

low-low frequency subband can be taken to form a block, $B_{0,0}^{m,n}$ of size $m \times n$, $m = 2^{x-N}$, $n = 2^{y-N}$. Note carefully that in this description, the values of m, n are defined for convenience to coincide with the subband boundary, while in fact according to the earlier remark, these values can be treated as parameters. In general, an arbitrary block $B_{k,l}^{m,n}$ consists of set of indexed wavelet coefficients $\{c_{i,j} | k \leq i < k+m, l \leq j < l+m\}$, (k, l being the top left corner image coordinates of the block) and its direct descendant blocks are

$$\{B_{2k,2l}^{m,n}, B_{2k+m,2l}^{m,n}, B_{2k,2l+n}^{m,n}, B_{2k+m,2l+n}^{m,n}\}. \quad (1)$$

In a similar manner to SPIHT, define sets $D(B)$ as all descendants for block B and $L(B)$ as all descendants except the direct descendants. For some integer r define the significance function, $S(\cdot)$ applied to set R as

$$S_r(R) = \begin{cases} 1, & \max(\{|c_{i,j}|\}) \geq 2^r, c_{i,j} \in R \\ 0, & \text{otherwise.} \end{cases} \quad (2)$$

Also like SPIHT, define three ordered auxiliary lists:

1. LIB (List of Insignificant Blocks) to contain insignificant blocks of varying sizes with one or more member;
2. LIS (List of Insignificant Sets) to contain insignificant sets (with more than one member); and
3. LSP to contain significant pixels/coefficients.

Entries in LIS can be of type A or B (corresponding to D or L type descendants), receiving different processing in each case.

After initialization, the encoder algorithm creates a tree of blocks (in SPIHT fashion) before recursive partitioning at the block level (in SPECK fashion), until significant coefficients are found in a block. Therefore, compared to SPIHT, the unified algorithm, having identified significant block-trees, delays output to the LSP until after a further block partitioning step. The notional root of the tree is $B_{0,0}^{m,n}$, which has three direct descendants. Thereafter, block quad-trees are formed, rooted at the three direct descendants.

A complete description of the uniform quantizer is as follows:

Algorithm overview

1. Initialization pass:

Output $r = \lfloor \log_2(\max(\{|c_{i,j}|\})) \rfloor$, $c_{i,j} \in X$; set LSP= NULL; add $B_{0,0}^{m,n}$ to LIB; and put $B_{0,m/2}^{m,n/2}$, $B_{n/2,0}^{m,n/2}$, $B_{m/2,n/2}^{m,n/2}$ in LIS, as type A entries.

2. Block Splitting pass: $Codec_B()$

3. Block Tree Partitioning pass: $Codec_T()$

4. Refinement pass:

For each entry (i, j) in the LSP, except for those in the previous pass, output the r^{th} most significant bit of $|c_{i,j}|$.

Procedure: $Codec_B()$

1. For each element $B_{k,l}^{m,n}$ in the LIB, output $S_r(B_{k,l}^{m,n})$.
 2. If $S_r(B_{k,l}^{m,n}) = 1$
 - If $(m \neq 1) \&\& (n \neq 1)$, for $B_{k,l}^{m/2,n/2}$, $B_{k+m/2,l}^{m/2,n/2}$, $B_{k,l+n/2}^{m/2,n/2}$, $B_{k+m/2,l+n/2}^{m/2,n/2}$ call $Codec_B()$ in each case; else, as $B_{k,l}^{m,n}$ is a single pixel/coefficient, add (k, l) to the LSP and output the sign of $c_{k,l}$.
 - Remove $B_{k,l}^{m,n}$ from the LIB.
- else retain $B_{k,l}^{m,n}$ in the LIB.

Until all the $S_r(B) = 0$.

Procedure: $Codec_T()$

For each entry $B_{k,l}^{m,n}$ in the LIS do:

- If the entry is of type A
 - Output $S_r(D(B_{k,l}^{m,n}))$.
 - If $S_r(D(B_{k,l}^{m,n})) = 1$ then
 - * For each of $B_{2k,2l}^{m,n}$, $B_{2k+m,2l}^{m,n}$, $B_{2k,2l+n}^{m,n}$, $B_{2k+m,2l+n}^{m,n}$ call $Codec_B()$.
 - * If $L(B_{k,l}^{m,n}) \neq \phi$ then move $B_{k,l}^{m,n}$ to the end of LIS, as an entry of type B.
 - * Remove entry $B_{k,l}^{m,n}$ from the LIS.
- If the entry is type B then,
 - Output $S_r(L(B_{k,l}^{m,n}))$.
 - If $S_r(L(B_{k,l}^{m,n})) = 1$; then
 - * Add $B_{2k,2l}^{m,n}$, $B_{2k+m,2l}^{m,n}$, $B_{2k,2l+n}^{m,n}$, $B_{2k+m,2l+n}^{m,n}$ to the end of LIS as type A entries.
 - * Remove $B_{k,l}^{m,n}$ from the LIS.

In this algorithm, a simple form of rate-distortion control has been employed by placing the call to $Codec_B()$ before the call to $Codec_T()$. The motivation behind this ordering, is that only significant pixels contribute to the PSNR (Peak Signal-to-Noise) performance. As $Codec_T()$ only generates block significance information, whereas $Codec_B()$ generates pixel-level significance, $Codec_B()$ should come first.

	Lenna			Goldhill		
bpp	0.25	0.50	1.00	0.25	0.50	1.00
SPECK	33.37(dB)	36.49(dB)	39.65(dB)	30.21(dB)	32.58(dB)	35.67(dB)
SPIHT	33.70	36.85	39.99	30.22	32.71	36.00
EZBC	33.80	36.91	40.06	30.29	32.83	36.16
BT-1	33.68	36.83	39.98	30.20	32.67	35.99
BT-2	33.75	36.88	40.02	30.25	32.75	36.08
BT-4	33.76	36.88	40.03	30.26	32.77	36.10
BT-8	33.78	36.89	40.03	30.27	32.78	36.10
BT-64	33.79	36.89	40.03	30.27	32.78	36.10

Table 1. PSNR comparison for ‘Lenna’ and ‘Goldhill’ images.

4. Results

In Table 1, for two example 512×512 standard test natural images, the result of encoding by the block-tree (BT) method of Section 3 is compared with results for testing by well-known encoders. EZBC [3] is a block-based encoder with adaptive arithmetic encoding. The figures for SPECK, SPIHT, and EZBC are taken from [2]. Notice that the version of SPECK tested in [2] did not include arithmetic coding, unlike the original [7]. This version of SPECK also employed a fixed block (zig-zag) scanning order with no encoding of the final sub-block. All other algorithms in Table 1 employed a fixed scanning order with simple encoding of sub-blocks. The PSNRs are compared against a range of bits-per-pixel (bpp) communication rates. There is an approximate equivalence in algorithm between SPIHT and BT-1. However, generally SPIHT is slightly better in PSNR performance. Increasing the block size to 2×2 in the BT family of encoders, makes BT more like SPECK, but its performance is consistently better over the test images.

It is evident that in terms of total rate distortion, the more complex EZBC algorithm in PSNR terms always outperforms SPIHT and SPECK. However, by increasing the block size the BT family of encoders can approach the performance of EZBC, without its complexity. In general, block methods outperform zerotree methods. As the BT algorithm becomes less of a tree algorithm, and more of a block algorithm, the PSNR performance is increased. As previously noted, there is a limit to block sizes in the sense that one would want the block size to fit into cache memory. On the other hand, the BT method moves towards a parameterizable compression system, which can be crafted according to needs.

5. Conclusion

This paper has shown that it is possible to amalgamate the ‘zerotree’ and ‘quad-tree block’ algorithms, and that the resulting algorithm can be adjusted according to block size

to be more zerotree like, or more quad-tree like. The amalgamation also helps make clear the relationship between the two algorithmic types. By increasing the block size the PSNR performance is incrementally improved, though memory cache usage will be different. Remaining characteristics such as the possibility of decomposition into sub-images, and intermediate rate-distortion optimization remain available.

References

- [1] E. S. Hong and R. E. Ladner. Group testing for image compression. In *Data Compression Conference, DCC'00*, pages 3–12, 2000.
- [2] S.-T. Hsiang. *Highly Scalable Subband/Wavelet Image and Video Coding*. PhD thesis, Rensselaer Polytechnic Institute, 2002.
- [3] S.-T. Hsiang and J. T. Woods. Embedded image coding using zeroblocks of subband/wavelet coefficients and context modeling. In *ISCAS'00*, volume 3, pages 662–665, 2000.
- [4] A. Islam and W. A. Pearlman. An embedded and efficient low-complexity hierarchical image coder. In *Visual Coding and Image Processing, VCIP'99*, pages 294–305, 1999. SPIE vol. 3653.
- [5] S. Mallat. A theory for multiresolution signal decomposition: The wavelet transform. *IEEE Transactions on Pattern and Machine Intelligence*, 11(7):674–693, 1989.
- [6] W. A. Pearlman. Presentation on core experiment codeff 08: Set Partitioned Embedded Block Coding (SPECK), 1999. ISO/IEC/JTC1/SC29 WG1 N1245.
- [7] A. Said and W. A. Pearlman. A new fast and efficient image codec based on Set Partitioning in Hierarchical Trees. *IEEE Transactions on Circuits and Systems for Video Technology*, 6(3):243–250, 1996.
- [8] J. M. Shapiro. Embedded image coding using zerotrees of wavelet coefficients. *IEEE Transactions on Signal Processing*, 41(12):3445–3462, 1993.
- [9] F. W. Wheeler and W. A. Pearlman. Combined spatial and subband block coding of images. In *International Conference on Image Processing, ICIP'00*, volume 3, pages 861–864, 2000.