Prediction and Adaptive Scanning in a Block-Set Wavelet Image Coder

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

University of Essex,

Department of Electronic Systems Engineering,

Wivenhoe Park, Colchester, CO4 3SQ, United Kingdom

tel: +44 - 1206 - 872817

fax: +44 - 1206 - 872900

e-mail {xwyin,fleum,acd}@essex.ac.uk

Abstract

Conditional probability to predict sub-block significance improves the peak signal-to-noise ratio (PSNR) performance of a block-set wavelet image coder. Adaptive scanning of sub-blocks results in a further gain in PSNR performance. Together the related algorithmic changes introduced in this paper result in competitive PSNR performance without the need for an arithmetic coding stage. As a consequence, timing tests reveal that the critical decoder computational complexity is reduced.

1 Introduction

Recent still-image codecs, including the JPEG2000 standard codec [1], employ a wavelet transform to achieve space-frequency localization. The 2-D wavelet transform for images [2], using the almost invariable 9-7 tap bi-orthogonal filter [3], produces a pyramidal sub-band structure. Transform coefficients are correlated in frequency within sub-bands and spatially across sub-bands. Block-based coefficient coders take advantage of the fact that (in)significant coefficients tend to be clustered within a sub-band. Following coefficient coding, some form of entropic coding of the bit-stream may optionally be applied. For example, context-adaptive binary arithmetic coding (CABAC) [4][5] is considered to be optimal in terms of compression efficiency.[1] However, as CABAC's algorithmic complexity remains high, arithmetic coding may be omitted to reduce latency. Often, it is important to minimize the time taken to decode an image, whereas encoding time is less critical as images may be pre-encoded or high-performance processors can be used. Whereas the encoder is under the control of the image database or library administrator, the decoder is not and may even run on a mobile device, with limited processing power.

This paper refines block set coding by a more efficient treatment of sub-block scanning order and bit allocation. An adaptive scanning order is introduced and sub-block significance is predicted by means of conditional

[1] Multi-symbol arithmetic coding can further improve the compression ratio at a cost in computational complexity.

probabilities. Conditional probabilities are predicated on the significance of parent and neighbouring coefficients, as well as on the significance of child-blocks in the previous quantization round. Adaptive scanning results in an improved peak-signal-to-noise ratio (PSNR)[2]. As a result, arithmetic coding is no longer required, as PSNR measurements indicate a similar image quality at the decoder compared to that obtained by a similar coder with arithmetic coding. Timing experiments confirm that the modified decoder has reduced computational complexity compared to a similar unmodified coder with arithmetic coding.

In fact, quad-tree block-set coders are normally of lower complexity than other types of coder such as those based on a zero-tree structure [6] including SPIHT [7][8]. Block access also has the potential to provide beneficial memory access patterns, because of locality of reference, and because the dynamic memory requirements all lie within one sub-band. However, not all block algorithms are of low computational complexity; for example, ECECOW [9] achieves outstanding PSNR results but requires prior training on sample images to arrive at a prediction model that is embedded in the coefficient coding process. Although the JPEG2000 coder algorithm, EBCOT [10][11], restricts memory usage through coding of pixels within small code-blocks (16×16 pixels), it is also of high complexity as a result of: integrating CABAC into the codec; multiple coding passes; and application of rate-distortion optimization. In [8], timing tests comparing the Set Partitioning

[2]Defined as $10 \log_{10}(255^2/\epsilon)$, where ϵ is the mean of the squares of the pixel errors and 255 is the maximum grey-level value.

Embedded bloCK (SPECK) block-set coder to JPEG2000's [12][1] VM 3.2A showed SPECK to be between 4.6 and 15.7 times faster in encoding, and 8.1 to 12.1 times faster in decoding on average over a set of four images and a set of four rates, 0.25, 0.50, 1.0 and 2.0 bit/pixel.

Given that the block appears to be the natural basis of any low-complexity algorithm, we wondered whether full advantage had been taken of conditional probability, when constructing quad-trees. In fact, some existing algorithms appear inconsistent in their treatment of the quad-tree traversal once the primary block has been encoded. In our approach, bit savings are made at the scanning stage, and hence, unlike the case when entropic coding has been previously applied, it becomes unnecessary to employ variable length coding to identify redundant bit sequences.

The remainder of this paper is organized as follows. Section 2.1 briefly describes the role of block-set coefficient coders within a codec and Section 2.2 introduces block-set coding. Section 3 details the block-set coding techniques introduced in this paper to improve low-level scanning of blocks. Section 4 describes the prediction model implemented to test the adaptive scanning algorithm. Section 5 compares the PSNR results for a range of block-set coders including those designed by us, whereas timing tests show the speed-up in the decoder from our algorithmic modifications. Finally, Section 6 draws some conclusions.

2 Background

2.1 Wavelet codec structure

The canonical structure of a 2-D wavelet transform exploits separability by performing a series of 1-D transforms, Figure 1, at successively coarser granularity, in the manner of quadrature mirror filters. The quantization step is primarily responsible for compression through rejection of low-valued transform coefficients, with secondary compression resulting from entropy coding (if used). Figure 2 shows the generic structure of a codec, with image pixels, x, transformed to wavelet coefficients, w, which are uniformly quantized by thresholding, c, from the highest power of two present, integer n, downwards.

For each bit plane, some form of group significance test is performed on the coefficient stream. At each quantization level, if a group or set (formed from a coefficient's descendants or its spatial neighbors) is insignificant (contains no member above the current quantization level), only one bit is generated at the encoder, as the decoder mimics the decision structure of the encoder and hence infers the implication of the single bit.

Subsequently, some form of entropic coding (simple or adaptive Huffman, CABAC) is commonly performed to produce the output bitstream, b. The decoder reverses the process, to produce lossy output, $\hat{x}$. This paper proposes removal of entropic coding, substituting predictive and adaptive scanning within the group test.

2.2 Block-set coding

The authors have designed a hybrid block-tree (BT) algorithm which takes advantage of block-set coding, and also correlations across sub-bands in the manner of classic tree-coders such as SPIHT [7]. BT is *not* the subject of this paper and details of its algorithm can be found in [13]. When used with large-sized blocks, *e.g.* 64×64, BT's behaviour becomes block-like and is similar to the well-known block-set coder SPECK [14]. It is to the BT algorithm that the adaptive scanning and a block significance prediction model have been applied, with the results given in Section 5. Other block-set coders include NQS [15], SWEET [16], SBHP [17](a variant of SPECK which was proposed as a low-complexity alternative in the JPEG2000 still image standard), and AGP [18]. A comparison in Section 5 is made with EZBC [19], a block-set coder with an additional CABAC stage.

SPECK is a classic example of block-set coding, whereby a transformed image is recursively partitioned into sub-blocks, achieving compression by single-symbol coding of insignificant blocks[3]. Recursion ends when significant coefficients are found. The advantage of block-set coding 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 coded in image I_0. The block coding order then proceeds, I_1 through the remaining sub-bands at the same level, $S_{1,2,3}$, before turning

[3]See 'An example of SPECK Coding' W. A. Pearlman available from `http://www.cipr.rpi.edu/ pearlman/`

to the remaining sub-bands, $S_{4,5,6,\ldots}$, in I_2. In Figure 4, only one of the typical sub-band's quadrants has significant pixels. This block is first coded by '0100' to identify the position of that quadrant. Recursively splitting the significant quadrant, results in the bit stream sequence illustrated in Figure 4.

The resulting binary bit-stream is commonly coded using CABAC, *e.g.* [20]. Adaptive arithmetic coding suffer from the need to update after every symbol, leading to attempts to reduce complexity by periodic updates, pseudo-adaptive coders. Alternatively, using a binary arithmetic coder leads to some loss in coding precision compared to a multi-symbol coder. In [21], which is a comprehensive comparative study of entropic coding times, it is noted that static Huffman coding remains significantly faster than widely-used CABAC implementations, by as much as an order of two magnitudes on a 2 GHz Pentium 4 processor, though adaptation to current microarchitectures can narrow this performance gap to some extent. CABAC is suited to modelling unknown sources.

3 Block-set coding techniques

Quad-tree decomposition allows a block to be coded separately from its four sub-blocks. One bit is allocated to denote significance or insignificance. If a block is insignificant, then no further bit allocation is necessary, but otherwise a further four bits are required to identify the significant sub-blocks.

However, if the significance of each sub-block were to be tested directly then just four bits would be required, with no extra bit needed for the super-block. The trade-off between direct coding of sub-blocks and coding the super-block and then its component sub-blocks can be analyzed as follows. In general,

*total bit budget = probability of insignificance * insignificant bit budget + probability of significance * significant bit budget.*

Applying this principle, let t represent the probability that a sub-block is insignificant, and assume that the sub-block probabilities are taken from identical independent distributions (i.i.d.). Then,

$$1 \times t^4 + (1 - t^4) \times 5 \leq 4 \Rightarrow t \geq 0.71 \tag{1}$$

Therefore, if each sub-block has a more than 71% chance of being insignificant, then testing this block separately from its sub-blocks will outperform direct testing. 71% is a high assumed probability, and if true, implies that further advantage can be taken of this correlation.

It turns out that only limited advantage is taken of the assumed high probability in contemporary algorithms. If a number of blocks have been established as insignificant, but the block tree is established as significant, then by process of elimination a remaining block is significant. In Figure 5, three cases where coding of the final significant block becomes unnecessary are shown. In case 1, if block tree A has been identified as significant but child block a is known to be insignificant, then there is no need to code block tree B as significant, as it must be significant to justify its super-tree

A being identified as significant. In case 2, assuming block tree B is indeed significant but super-blocks a0–a2 are not significant, then it automatically follows that sub-block a3 is significant, and there is no point wasting a bit on coding this fact. Finally, in case 3, if block b, part of block tree B, is significant and sub-blocks b0-b2 are found to be insignificant, it also follows that b3 is automatically significant, and a bit need not be expended in coding it.

This is a simple form of conditional sub-block coding. Section 3.2 is a more formal treatment of simple sub-block coding, whereas Section 3.3 analyzes a more complex form of sub-block coding. However, before passing to that discussion, consider varying the sub-block scanning order.

The sub-block scanning order is normally a fixed (usually raster scan) order. The raster scan scan has some advantages in cache access. However, it is also possible to adapt the scanning order in order to locate insignificant blocks more quickly (and thus make coding of the remaining significant block unnecessary).

3.1 Adaptive scanning

In adaptive scanning, for each sub-group, a weight value is set, indicating the probability that this sub-group will become significant. The weighting value depends on three factors: parent significance, children's significance (from the previous coding round), and already scanned (or previous round) neighbours' significance ratings. The relative value of each of the three

contributions (which might be zero) is implementation dependent.

Once the total significance probability is calculated for each sub-group, the scanning order is rearranged according to the weighting value. As a result, the sub-groups that are most likely to be insignificant will be scanned first, whereas the sub-group that is most likely to be significant will be scanned last. In this way, a series of events E_0, E_1, E_2 can be used to predict event E_3. Adaptive scanning can be employed either with simple coding of subgroups or with complex coding.

3.2 Simple coding

Suppose there exists a block $K = \{K_0, K_1, K_2, K_3\}$, where K_0, K_1, K_2, K_3 are sub-blocks of K. Given a threshold n, the significance function is defined as:

$$S(K) = \begin{cases} 1, \max(K) \geq n \\ 0, \text{otherwise} \end{cases} \tag{2}$$

If K is insignificant, there is no need to test its sub-blocks any more. If K is significant, each of its sub-blocks need to be scanned whether they are significant or not.

Suppose further, for sake of illustration, that the scanning order is $K_0 \rightarrow K_1 \rightarrow K_2 \rightarrow K_3$(raster scan scanning order). If the first three sub-blocks are all insignificant, the last sub-block should be significant. This significance information is redundant, and can be discarded on the encoder side. As usual, on the decoder side, it is easy to recover this information by mimicking the encoder's behavior. The inference made by the decoder can be

represented by:

$$\{S(K) = 1, S(K_0) = 0, S(K_1) = 0, S(K_2) = 0\}$$
$$\Rightarrow S(K_3) = 1.$$

Further, let

$$E_n \text{ represent the event } \begin{cases} S(K_m) = 1, & m = n \\ S(K_m) = 0, & m \neq n \end{cases} \tag{3}$$

for $0 \leq m, n < 3$.

Assume that the significance probabilities of K_0, K_1, K_2, K_3 are equal, and taken from i.i.d's. Then obviously, $p(E_0) = p(E_1) = p(E_2) = p(E_3)$. In the above raster scanning order, if only E_3 is predicted, it means that just 25% conditional probability is exploited.

There is no condition in this analysis that requires a fixed scanning order, raster scan or otherwise, but, in any case, it is apparent that a more efficient set of tests can be applied.

3.3 Complex coding

Complex coding of sub-blocks can be employed either with a fixed or adaptive scanning order. The essence of the sub-block coding introduced is that a distinction is made between significant blocks with one significant sub-block and those with more than one significant sub-block.

If there is just one significant sub-block in a significant block, then only two bits are needed to identify its relative position within the block. This

implies that an extra bit is needed to indicate whether the block contains just one significant sub-block or more than one sub-block. As with the trade-off between coding separately block and sub-block significance vis-à-vis directly coding sub-blocks, the advantage of this procedure is related to the probability that a sub-block will be insignificant. However, inequality (1) implies that the average insignificant chance is more than 71% for each sub-block. In fact, by the following analysis, even if t is less than 71%, there is still a coding gain from complex prediction.

If the assumption that there is an advantage from coding the number of significant sub-blocks (either one, or more than one sub-block) is correct, then the following inequality should be satisfied:

$$\begin{aligned} t^4 + (1-t^4) \times 5 \quad \geq \quad & t^4 + (1-t^4) + (1-t^4) + \\ & 4 \times t^3(1-t) \times 2 + u \\ & \times [1 - t^4 - 4 \times t^3(1-t)] \end{aligned} \tag{4}$$

u represents how many bits are need to describe whether a block contains two or more than two significant sub-blocks using sequential scanning. The left-hand side (l.h.s.) of (4) is simply (1), the bit budget for no coding of the number of significant sub-blocks, while the right-hand side (r.h.s.) (the bit budget for coding the number of significant sub-blocks) is formed from: the probability of four insignificant sub-blocks with bit weighting of one; the probabilities of either having just one significant sub-block, or more than one significant sub-block, with bit weighting of one for each alternative; the four different probabilities of having just one significant sub-block, with bit

weighting of two bits for each; and finally the probability of two or more significant sub-blocks, with bit weighting of u.

Assume that just 3.5 bits on average are needed for u. (Refer to Section 3.4 for the justification of this assertion.) Solving inequality (4) for t:

$$\begin{aligned} \Rightarrow 5 - 4 \times t^4 \quad &\geq \quad 2 - t^4 - 8 \times t^4 + 8 \times t^3 \\ &\quad + 3.5 + 10.5 \times t^4 - 14 \times t^3 \end{aligned} \tag{5}$$

$$\Rightarrow t > 0.55 \tag{6}$$

As existing algorithms already assume a value of $t \geq 0.71$, then the implication is that a sub-block's chance of being insignificant will always satisfy this requirement.

3.4 Average number of bits

The value of u, the average number of bits employed to represent whether two or more than two sub-groups are significant, is justified in this Section.

In Table 1, assuming two or more of the sub-groups K_0–K_3 are significant, then the number of bits required to code the particular constellation of sub-groups, and the conditional probability of that constellation are tabulated. In Table 1, it is assumed that the scanning order is from K_0 to K_3 in sequential order. In the first row of Table 1, if, for example, K_0 and K_1 are already identified as insignificant, and one bit is employed for each identification, then, as it is already indicated that there are at least two significant sub-groups, there is no need to expend further bits.

The term in the denominator of the probability expressions is explained as follows. There are six different ways of assigning just two insignificant sub-groups, probability $6t^2$. These six ways are combined with the probability of finding the two remaining sub-groups to be significant, which is $(1-t)^2$. However, as this latter term is also found in the numerator, it has been cancelled out. The same cancellation has occurred for the term $4t(1-t)$, which represents the four different ways of combining a significant with insignificant sub-group, when choosing from all 11 possibilities. The final term represents the one way of finding all four sub-groups as significant (last row of the table), again after cancellation of the common term. The numerator terms represent, before cancellation of the common $(1-t)^2$ term, the probability of just one of the events out of the three types represented by the denominator terms. The complete probability expression is the probability of one of the events represented in the numerator, conditioned by the total probability represented in the denominator.

Assume $t = 0.71$ from (1), and substitute into each of the probabilities, then sum. Finally, scale by the number of bits allocated for each common probability expression, to find:

$$(20 \div 6) \times 0.7689 + (16 \div 4) \times 0.2094 + (4 \div 1) \times 0.0213) = 3.4862 \approx 3.5. \quad (7)$$

For example, there are twenty bits allocated to the first six rows of Table 1, and each has the same probability expression which evaluates to 0.7689, with $t = 0.71$.

In (7) u is found to be approximately 3.5, which is the value for u used in Section 3.3. As a fixed scanning order was used in the analysis, the implication is that for an adaptive scanning order, $u < 3.5$, and the estimate of Section 3.3 is conservative. However, the analysis has also been simplified by assuming all probabilities are drawn from i.i.d.'s.

4 Implementing predictive coding

This Section describes the form of the prediction model used in block coding by us. Section 5 then gives comparative results from applying this predictive model with either a fixed or adaptive scanning order.

The current heuristic prediction model employed in the block coding algorithm (Figure 6) uses twelve neighbouring sub-blocks, four parent sub-blocks, and four child sub-blocks. Where necessary these values are taken from a previous round of bit layer coding.

Assume firstly that, in Figure 6, all four sub-blocks, K_0 – K_3 and all twelve neighbours of the sub-blocks, NE_0–NE_{11}, lie within the boundary of a sub-band. Assume also that a sub-block under test has children, *i.e.* is not at the lowest sub-band layer (highest resolution). Then, vertical and horizontal neighbours if significant are weighted 6, and diagonal neighbours are weighted 3, as they are known to give less predictive information. If a neighbour is not significant then the weighting is 0. In Figure 6, NE_i $i = 1, 2, 4, 5, 7, 8, 10, 11$ are weighted 6, and NE_i $i = 0, 3, 6, 9$ are weighted 0. Parents and children contain more valuable information for blocks, so their

weighting is 9 if significant, and 0 otherwise.

Now assume that some of the neighbours lie outside the boundary of the sub-band, either falling across a neighbouring sub-band or completely outside the transform image. Weight any such sub-blocks with a weighting of one. The logic behind this decision is that if weighted zero then these external neighbours would bias the weighting, whereas if allocated a value of more than one the bias might be in a different direction.

Finally, if a sub-block is a single pixel then the parent weighting is zero. Likewise a child weighting will be zero, if the child has no further children.

5 Results

5.1 Compression performance

Table 2 contains comparative results using a variety of standard coders reported in the literature. The comparison was made with the following coders. EZBC is a block-based coder with adaptive arithmetic coding. EZBC employs a fixed scanning order with simple coding of sub-blocks. The version of SPECK in Table 2 employed a fixed block (raster) scanning order with no coding of the final sub-block. The row labelled SPECK-AC represents the original results from [14], in which simple sub-block coding with a fixed raster scanning order was employed, along with arithmetic coding. SPIHT [7] is a well-known coder that is included as a reference point. The PSNR figures for SPECK, EZBC, and SPIHT are taken from [22]. Notice that the version of SPECK tested in [22] did not include arithmetic

coding, unlike the original [7].

The comparison is with three variants of our hybrid BT block coder. 'BT' is the block-set, in-house coder, already introduced in Section 2.2. 'BT-P' uses a fixed scanning order, large blocks of 64×64, and complex coding of sub-blocks. BT-A, uses the adaptive scanning order of Section 3.1. Both BT-P and BT-A employ the prediction model of Section 4. Lenna and Goldhill are 512×512 standard test natural images[4].

When an adaptive scanning order with complex coding is applied, BT-A, then the performance approaches within 0.1 dB of SPECK-AC on average for these test images. Though there is extra complexity involved from forming the coding tests, this extra complexity does not approach that of an arithmetic coder.

5.2 Time complexity performance

Timings were taken on a PC with Pentium 4 processor, with CPU speed 1.7 GHz and 256 MB RAM. Table 3, for the standard test images and a range of bit-rates (similar to those in the SPECK comparison [8] mentioned in Section 1).

The results show that decoder time performance for the BT-A algorithm is consistently better than the SPECK-AC algorithm [14]. The common wavelet transform times are recorded separately in Table 3, but the 'total time' percentage improvement is calculated after adding the two components

[4]Images accessed iii 3 03 from http://www.sys.uea.ac.uk/Research/researchareas/imagevision/images_ftp/

(transform and coding) for each algorithm. For a bit-rate of 1.0 bpp there is over 10% overall improvement in execution speed for 512×512 pixel sized images. For 'coding time', *i.e.* coefficient and entropic coding combined, the improvement is about twice this. As operation of arithmetic coding and coefficient coder are interleaved in the SPECK-AC algorithm, it was not feasible to provide separate times for the two operations.

The encoder timings for BT-A are currently inferior to those of SPECK-AC, as only the decoder code has been converted from initial 'proof-of-concept' code to production code. For the reasons given in Section 1, this is not a particular problem for pre-encoded still images, as it is the decoder performance that is critical.

No special optimizations were applied to the decoder code and the coding level was consistent with that of SPECK-AC. The program code of SPECK-AC was that of the Quantization, Compression and Coding Library, QccPack version 0.47.1 available from `http://qccpack.sourceforge.net/`, and the coding of BT-A used the same program code as its basis. The code was run under the Linux operating system, with the Unix utility `gettimeofday` recording timings.

The underlying reasons for the improved decoder times are postulated as follows. The time complexity of a coder is approximately proportional to the number of comparisons made. The scanning method described herein makes less comparisons than the common quad-tree method so that its complexity would be expected to be reduced. There is, however, a small increase

in complexity from looking up neighbouring pixels. The method also bears some relationship to simple or static Huffman coding, with much lower computational complexity compared to the CABAC algorithm [21] as applied in SPECK-AC.

6 Conclusion

In some existing block-set coders, once the main block is established as significant, there is inconsistent and, indeed, inefficient use of conditional probability to indicate which of the sub-blocks were making the whole block significant. This paper has introduced a way of reducing the bit budget by taking advantage of the fact that more than one significant sub-group is increasingly unlikely as one descends the sub-band pyramid. Combining this insight with an adaptive sub-block scanning order has led to gains in PSNR performance over standard low-complexity coders, without the need for an entropic coding stage, such as Huffman or arithmetic coding. An alternative would be to employ simple Huffman coding, but that would require construction of the coding tree beforehand and generation of a redundant bit stream, before reducing the redundancy by variable length coding. The approach of the paper is to remove redundant bits in the scanning process, thus obviating the need for entropic coding. The result is PSNR performance close to that which has only previously achieved by including an arithmetic coding stage. Timing tests show that overall the decoder will be about 10% faster than the already fast SPECK-AC decoder for bit rates of 1.0 bpp.

References

[1] D. Santa-Cruz, T. Ebrahimi, J. Askelöf, M. Larsson, and C. A. Christopoulos. JPEG 2000 still image coding versus other standards. In *SPIE 45th Annual Meeting: Applications of Digital Image Processing XIII*, pages 446–454, 2000.

[2] S. Mallat. A theory for multiresolution signal decomposition: The wavelet transform. *IEEE Transactions on Pattern and Machine Intelligence*, 11(7):674–693, 1989.

[3] M. Antonini, M. Barland, P. Mathieu, and I. Daubechies. Image coding using the wavelet transform. *IEEE Transactions on Image Processing*, 1(12):205–220, 1992.

[4] I. H. Witten, R. M. Neal, and J. G. Cleary. Arithmetic coding for data compression. *Communications of the ACM*, 30:520–540, 1987.

[5] W. B. Pennebaker, J. L. Mitchell, G. G. Langdon Jr., and R. B. Arps. An overview of the basic principles of the Q-coder adaptive binary arithmetic coder. *IBM Journal of Research and Development*, 32(4):717–726, 1988.

[6] J. M. Shapiro. Embedded image coding using zerotrees of wavelet coefficients. *IEEE Transactions on Signal Processing*, 41(12):3445–3462, 1993.

[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] W. A. Pearlman. Presentation on core experiment codeff 08: Set Partitioned Embedded Block Coding (SPECK), 1999. ISO/IEC/JTC1/SC29 WG1 N1245.

[9] X. Wu. High-order context modeling and embedded conditional encoding of wavelet coefficients for compression. In *31st Asilomar Conference on Signals, Systems, and Computers*, volume 23, pages 1378–1382, 1998.

[10] D. Taubman and A. Zakhor. Multi-rate 3-subband coding of video. *IEEE Transactions on Image Processing*, 3(5):572–588, 1994.

[11] D. Taubman. High performance scalable image compression with EBCOT. *IEEE Transactions on Image Processing*, 9(7):1158–1170, 2000.

[12] C. Christopoulos, A. Skodras, and T. Ebrahimi. The JPEG2000 still image coding system: An overview. *IEEE Transactions on Consumer Electronics*, 46(4):1103–1127, 2000.

[13] X-W. Yin, M. Fleury, and A. C. Downton. A structure for tree *and* block encoders. In *International Conference on Pattern Recognition, ICPR 2004*, 2004.

[14] 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.

[15] J. Spring, J. Andrew, and F. Chebil. Nested quadratic splitting, 2000. ISO/IEC/JTC1/SC29 WG1N1191.

[16] J. Andrew. A simple and efficient hierarchical image coder. In *IEEE International Conference on Image Processing, ICIP-97*, volume 3, pages 658–661, 1997.

[17] C. Chrysafis, A. Said, A. Drukarev, A. Islam, and W. A. Pearlman. SBHP - a low complexity wavelet coder. In *International Conference on Acoustics, Speech, and Signal Processing, ICASSP'00*, volume 4, pages 2035–2038, 2000.

[18] A. Said and W. A. Pearlman. Low-complexity waveform coding via alphabet and sample-set partitioning. In *Visual Communications and Image Processing '97, SPIE 3024*, pages 234–246, 1997.

[19] S.-T. Hsiang and J. T. Woods. Embedded image coding using zeroblocks of subband/wavelet coefficients and contex modeling. In *ISCAS'00*, volume 3, pages 662–665, 2000.

[20] X-W. Yin, M. Fleury, and A. C. Downton. Archive image communication with improved compression. In *International Conference on Document Analysis and Recognition, ICDAR 2003*, volume 1, pages 92–96, 2003.

[21] A. Said. Comparative analysis of arithmetic coding computational complexity. In *IEEE Data Compression Conference*, page 562, 2004. Full report available from `http://www.hpl.hp.com/techreports/2004/HPL-2004-75.pdf`.

[22] S.-T. Hsiang. *Highly Scalable Subband/Wavelet Image and Video Coding*. PhD thesis, Renssaeler Polytechnic Institute, 2002.

List of Tables

List of Figures

K_0	K_1	K_2	K_3	bits-required	event probability
0	0	1	1	2	$\frac{t^2}{6t^2+4t(1-t)+(1-t)^2}$
0	1	1	0	4	$\frac{t^2}{6t^2+4t(1-t)+(1-t)^2}$
1	1	0	0	4	$\frac{t^2}{6t^2+4t(1-t)+(1-t)^2}$
1	0	0	1	3	$\frac{t^2}{6t^2+4t(1-t)+(1-t)^2}$
1	0	1	0	4	$\frac{t^2}{6t^2+4t(1-t)+(1-t)^2}$
0	1	0	0	3	$\frac{t^2}{6t^2+4t(1-t)+(1-t)^2}$
1	1	1	0	4	$\frac{t(1-t)}{6t^2+4t(1-t)+(1-t)^2}$
0	1	1	1	4	$\frac{t(1-t)}{6t^2+4t(1-t)+(1-t)^2}$
1	0	1	1	4	$\frac{t(1-t)}{6t^2+4t(1-t)+(1-t)^2}$
1	1	0	1	4	$\frac{t(1-t)}{6t^2+4t(1-t)+(1-t)^2}$
1	1	1	1	4	$\frac{(1-t)^2}{6t^2+4t(1-t)+(1-t)^2}$

Table 1: Average bit usage probabilities for two or more significant sub-blocks

Images:		Lenna			Goldhill		
	bpp	0.25	0.50	1.00	0.25	0.50	1.00
Standard coders	SPIHT	33.70(dB)	36.85(dB)	36.99(dB)	30.22(dB)	32.71(dB)	36.00(dB)
	EZBC	33.80	36.91	40.06	30.29	32.83	36.16
	SPECK	33.37	36.49	39.65	30.21	32.58	35.67
	SPECK-AC	34.03	37.10	40.25	30.50	33.03	36.36
In-house coder(s)	BT	33.79	36.89	40.03	30.27	32.78	36.10
	BT-P	33.92	37.03	40.17	30.40	32.92	36.27
	BT-A	33.99	37.08	40.21	30.44	32.98	36.32

Table 2: PSNR comparison for 'Lenna' and 'Goldhill' images for a variety of block-set and other coders

Image	Bitrate (bpp)	Wavelet transform (s)	SPECK-AC (s)	BT-A (s)	Coding time % improvement	Total time % improvement
Lenna	0.25	0.094	0.145	0.135	6.9	4.2
	0.50		0.198	0.185	6.6	4.5
	1.00		0.304	0.260	14.5	11.1
Barbara	0.25	0.090	0.146	0.114	21.9	13.6
	0.50		0.200	0.161	19.5	13.4
	1.00		0.301	0.242	19.6	15.1
Goldhill	0.25	0.099	0.153	0.134	12.4	7.5
	0.50		0.213	0.188	11.7	8.0
	1.00		0.322	0.269	16.5	15.1

Table 3: Decoder timing comparisons for 'Lenna', 'Barbara', and 'Goldhill' images for the SPECK algorithm with arithmetic coding and the BT-A block-set algorithm

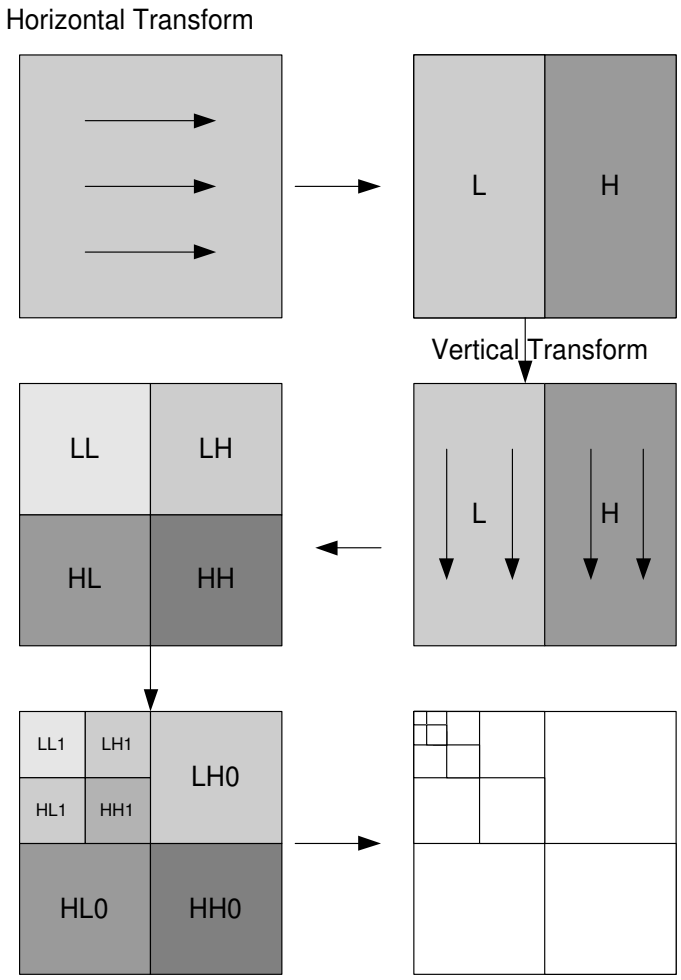

Figure 1: 2-D recursive pyramidal wavelet transform

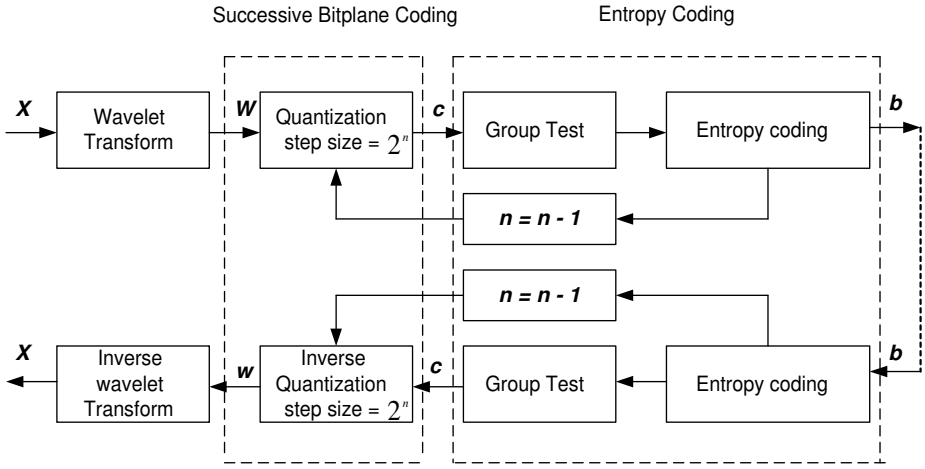

Figure 2: Block diagram of a generic still-image codec

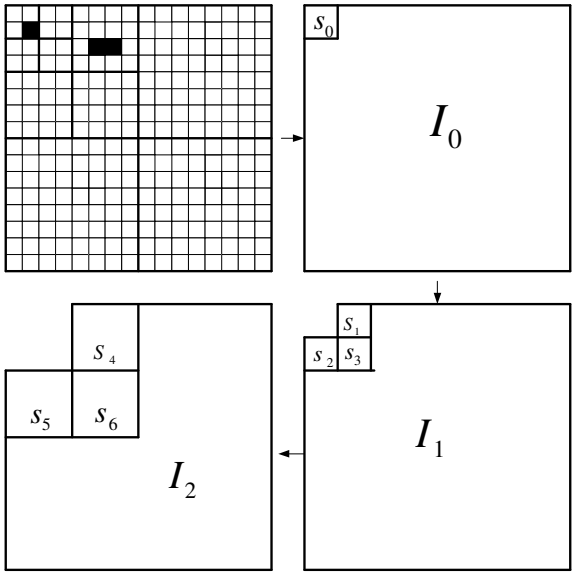

Figure 3: SPECK partitioning scheme for wavelet images

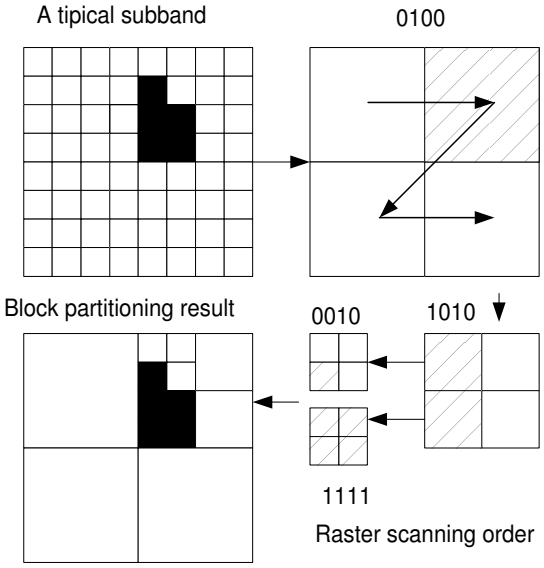

Figure 5: Block Encoding of a typical sub-band containing an area of significant pixels. Also shown is the standard raster scanning order

Figure 4: Block coding of a typical sub-band containing an area of significant pixels. The standard raster scanning order is also shown.

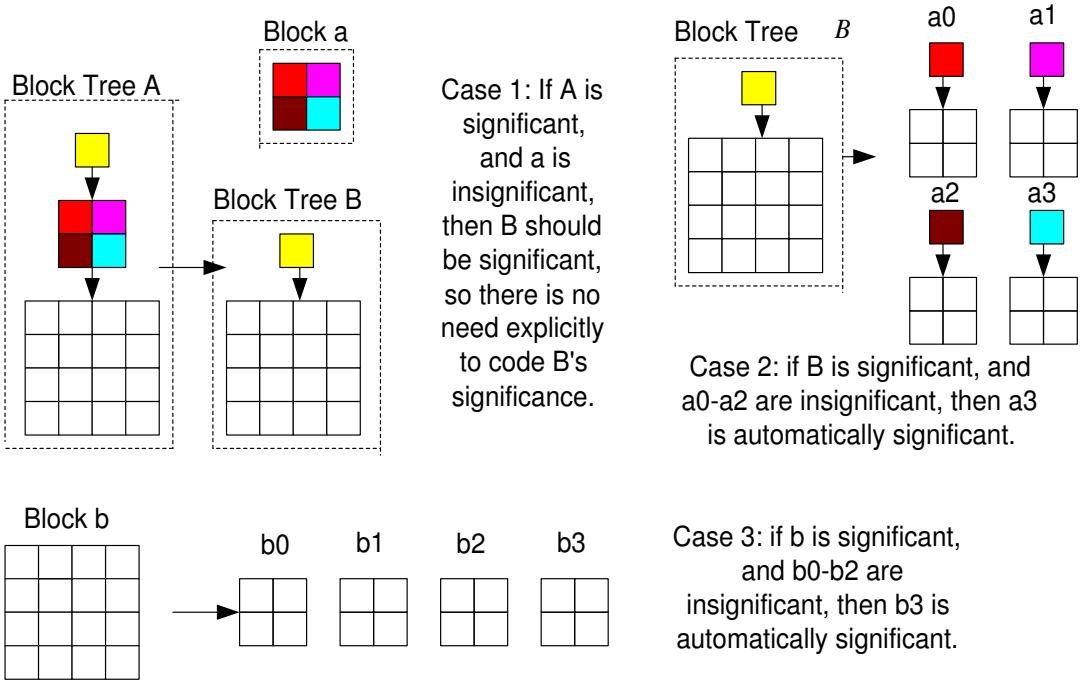

Figure 5: Conditional block coding, showing three cases in which bit savings are possible

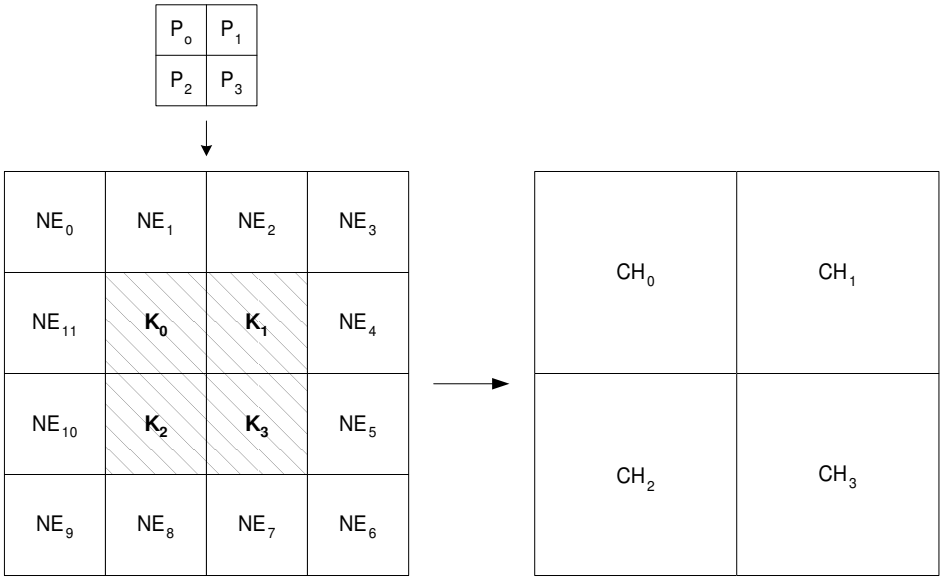

Figure 6: Weighting value calculation by parent, P, neighbour, NE, and child, CH, sub-block