High Performance Scalable Image Compression with EBCOT

David Taubman, *Member, IEEE*

***Abstract*—A new image compression algorithm is proposed, based on independent Embedded Block Coding with Optimized Truncation of the embedded bit-streams (EBCOT). The algorithm exhibits state-of-the-art compression performance while producing a bit-stream with a rich set of features, including resolution and SNR scalability together with a "random access" property. The algorithm has modest complexity and is suitable for applications involving remote browsing of large compressed images. The algorithm lends itself to explicit optimization with respect to MSE as well as more realistic psychovisual metrics, capable of modeling the spatially varying visual masking phenomenon.**

***Index Terms*—Embedded coding, image compression, JPEG 2000, random access, rate-distortion optimization, scalability, visual masking.**

I. Introduction

THIS paper describes a novel image compression algorithm known as EBCOT. The acronym is derived from the description "embedded block coding with optimized truncation" (EBCOT) which identifies some of the major contributions of the algorithm. The EBCOT algorithm is related in various degrees to much earlier work on scalable image compression. Noteworthy among its early predecessors are Shapiro's EZW (embedded zero-tree wavelet compression) algorithm [14], Said and Pearlman's SPIHT (spatial partitioning of images into hierarchical trees) algorithm [13] and Taubman and Zakhor's LZC (layered zero coding) algorithm [16]. Like each of these, the EBCOT algorithm uses a wavelet transform to generate the subband samples which are to be quantized and coded, where the usual dyadic decomposition structure attributed to Mallat [5] is typical, but other "packet" decompositions are also supported and occasionally preferable. Fig. 1 illustrates two decomposition structures which are considered in this paper. In each case, the original image is represented in terms of a collection of subbands, b_0, $b_1, \cdots$, which may be organized into increasing resolution levels, $\mathcal{L}_0$, $\mathcal{L}_1, \cdots, \mathcal{L}_L$. The lowest resolution level consists of the single LL subband, $\mathcal{L}_0 = \{b_0\}$. Each successive resolution level, $\mathcal{L}_l$, contains the additional subbands, b_i, which are required to reconstruct the image with twice the horizontal and vertical resolution.

Scalable compression refers to the generation of a bit-stream which contains embedded subsets, each of which represents an efficient compression of the original image at a reduced resolution or increased distortion. The terms "resolution scalability" and "SNR scalability" are commonly used in connection with this idea. We say that a bit-stream is resolution scalable if it contains distinct subsets, $\mathcal{B}_l$, representing each successive resolution level, $\mathcal{L}_l$. We say that a bit-stream is SNR scalable if it contains distinct subsets, $\mathcal{B}_q$, such that $\cup_{k=0}^{q} \mathcal{B}_k$ together represent the samples from all subbands at some quality (SNR) level, q. A bit-stream may be both resolution and SNR scalable if it contains distinct subsets, $\mathcal{B}_{l,q}$, which hold the relevant quality refinement of only those subbands in resolution level $\mathcal{L}_l$. A key advantage of scalable compression is that the target bit-rate or reconstruction resolution need not be known at the time of compression. A related advantage of practical significance is that the image need not be compressed multiple times in order to achieve

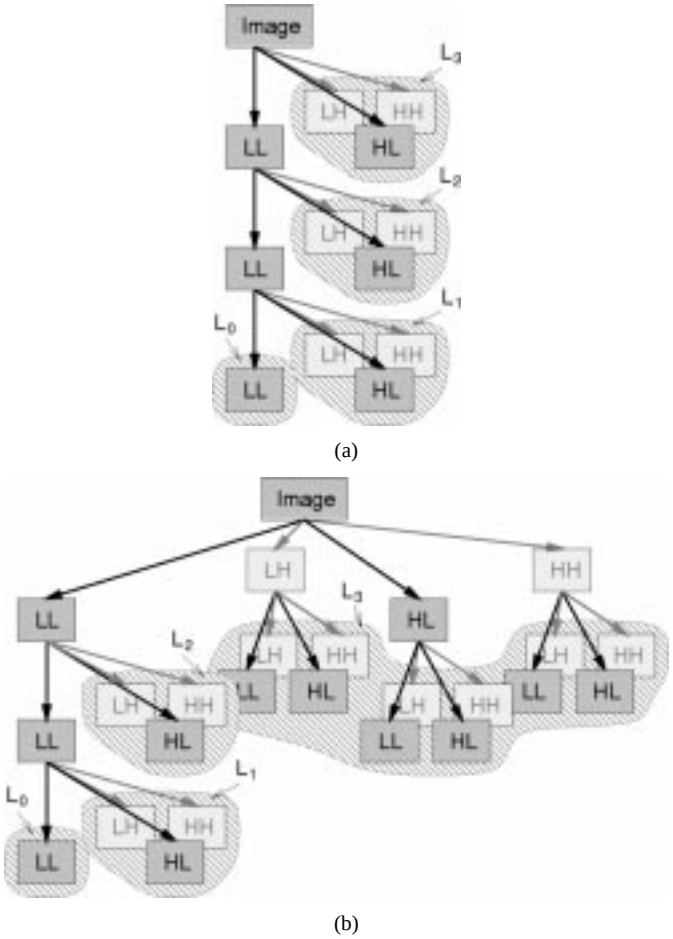


Fig. 1. (a) Mallat and (b) "Spacl" wavelet decomposition structure for three decomposition levels—five are used in the experiments.

Manuscript received May 4, 1999; revised October 13, 1999. This work was largely completed while the author was at Hewlett-Packard Research Laboratories, Palo Alto, CA. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Touradj Ebrahimi.

The author is with the School of Electrical Engineering, University of New South Wales, Sydney, Australia.

Publisher Item Identifier S 1057-7149(00)05436-1.

a target bit-rate, as is common with the existing JPEG compression standard.

In EBCOT, each subband is partitioned into relatively small blocks of samples, which we call *code-blocks*. EBCOT generates a separate highly scalable (or *embedded*) bit-stream for each code-block, B_i. The bit-stream associated with B_i may be independently truncated to any of a collection of different lengths, R_i^n, where the increase in reconstructed image distortion resulting from these truncations is modeled by D_i^n. An enabling observation leading to the development of the EBCOT algorithm is that it is possible to independently compress relative small code-blocks (say 32×32 or 64×64 samples each) with an embedded bit-stream consisting of a large number of truncation points, R_i^n, such that most of these truncation points lie on the convex hull of the corresponding rate-distortion curve. To achieve this efficient, fine embedding, the EBCOT block coding algorithm builds upon the fractional bit-plane coding ideas which have recently been introduced by Ordentlich *et al.* [10] and by Li and Lei [4]. The embedded block coding algorithm is developed in Section III.

A. Efficient One-Pass Rate Control

Given a target bit-rate, say $R^{\max}$, we can truncate each of the independent code-block bit-streams in an optimal way so as to minimize distortion subject to the bit-rate constraint. We refer to this as post-compression rate-distortion (PCRD) optimization, because the rate-distortion algorithm is applied after all the subband samples have been compressed. The PCRD optimization algorithm is described in Section II.

Although image compression schemes involving rate-distortion optimization abound in the literature, the advantage of PCRD optimization is its reduced complexity. The image need only be compressed once, after which the PCRD algorithm consumes negligible computational resources in passing over the embedded block bit-streams. Perhaps even more importantly, there is no need to buffer the entire image or indeed any quantity comparable to the size of the image. The wavelet transform and block coding operations may be implemented incrementally using a relatively small amount of memory which is proportional to one linear dimension of the image (say its width), as explained in [19]. Thus, the only representation of the image which must be buffered prior to PCRD optimization is the embedded block bit-streams, which are generally much smaller than the original image. In fact,it is also possible to perform the PCRD optimization step incrementally so that only a fraction of the compressed block bit-streams need be buffered. Earlier work on PCRD optimization may be found in [17]. The key features which distinguish EBCOT from this previous approach are the availability of more finely embedded bit-streams and the use of much smaller blocks of subband samples.

B. Feature-Rich Bit-Streams

The simplest incarnation of the concepts mentioned above is a bit-stream generated by concatenating the suitably truncated representations of each code-block, B_i, including sufficient auxiliary information to identify the truncation points, n_i, and the corresponding lengths, $R_i^{n_i}$. Such a bit-stream is clearly resolution scalable, because the information representing the individual code-blocks and hence the subbands and resolution levels is clearly delineated. Also, the bit-stream possesses a useful "random access" attribute: given any region of interest and a wavelet transform with finite support kernels, as is common, it is possible to identify the region within each subband and hence the code-blocks which are required to correctly reconstruct the region of interest [9].

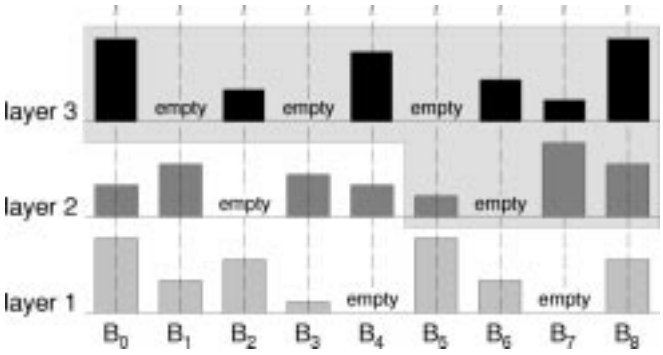


Fig. 2. Progressive appearance of embedded code-block bit-streams in quality layers. Only nine blocks and three layers shown for simplicity. The shaded region identifies the block contributions which are discarded by truncating the bit-stream between layers 1 and 2.

Interestingly, this simple bit-stream organization is not itself SNR scalable, despite the fact that it is composed of SNR scalable block bit-streams. This is because only a single truncation point and length are identified within the final bit-stream for each code-block. The EBCOT algorithm overcomes this difficulty by collecting incremental contributions from the various code-blocks into so-called quality layers, $\mathcal{Q}_q$, such that the code-block contributions represented by layers $\mathcal{Q}_1$ through $\mathcal{Q}_q$ form a rate-distortion optimal representation of the image, for each q. This is easily achieved with the aid of the PCRD algorithm described in Section II. In this way, truncating the bit-stream to any whole number of layers yields a rate-distortion optimal representation of the image, while truncating to an intermediate bit-rate yields a bit-stream which is approximately optimal provided the number of quality layers is relatively large. Fig. 2 illustrates the layered bit-stream concept; it also illustrates the effect of truncating the bit-stream between the first and second layers. Each quality layer must include auxiliary information to identify the size of each code-block's contribution to the layer. When the number of layers is large, only a subset of the code-blocks will contribute to any given layer, introducing substantial redundancy in this auxiliary information. To take advantage of this, EBCOT introduces a "second tier" coding engine to compress the auxiliary information for each quality layer.

EBCOT's layered bit-stream organization and two-tiered coding strategy represent a novel departure from current convention. Image compression algorithms previously described in the literature generate bit-streams whose organization is tied concretely to the structure of the embedded quantization and coding algorithm which is used. EBCOT, however, constructs abstract bit-stream layers, whose relationship to the truncation points offered by the underlying block coding engine is entirely arbitrary and is itself compressed. Fig. 3 illustrates

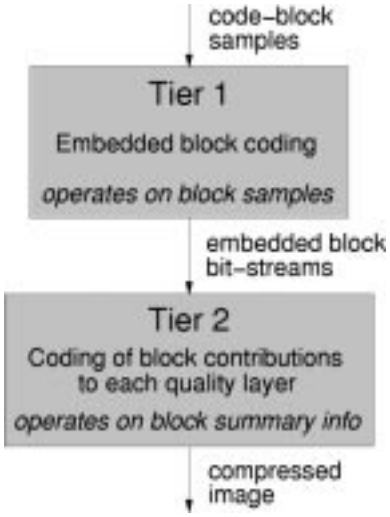


Fig. 3. Two-tiered coding structure of the EBCOT image compression algorithm.

this two-tiered compression paradigm. The layered abstraction and associated coding techniques are discussed further in Section IV. Useful bit-stream organizations range from single-layer streams which possess only the resolution scalable and random access attributes, through streams with a few layers targeted at specific bit-rates of interest, and ultimately to streams with a large number of layers, which offer excellent generic SNR scalability, in combination with the resolution scalable and random access attributes. EBCOT is able to achieve all of these features in a single bit-stream while exhibiting state-of-the-art compression performance, as demonstrated in Section V. By contrast, well known scalable image compression algorithms such as EZW [14] and SPIHT [13] offer only SNR scalability, to which the LZC algorithm [16] adds resolution scalability. The utility of the random access attribute is examined in Section VII. It is worth pointing out that the IW44 algorithm in AT&T's DjVu document compression system also achieves resolution and SNR scalability in combination with the random access attribute, using similar techniques to EBCOT; however, the abstract layering and PCRD optimization concepts are missing from IW44, which also has a less efficient embedded representation for each code-block. The DjVu specification is available at http://djvu.research.att.com/djvu/sci/djvuspec.

As a result of this rich set of features, modest implementation complexity, and excellent compression performance, the EBCOT algorithm was adopted for inclusion in the evolving JPEG2000 image compression standard at the Los Angeles international meeting of ISO/IEC JTC1/SC29/WG1 (JPEG working group) in November 1998. Most features of the algorithm were initially described in [18] and later in [19] as part of this standardization effort, but the work has not previously been published in the public arena. Since its original acceptance for JPEG2000, the algorithm has undergone several modifications to further reduce implementation complexity [6]; these are outlined in Section VIII for the benefit of readers who are interested in the relationship between EBCOT and JPEG2000.

II. Rate Distortion Optimization

Recall that EBCOT partitions the subbands representing the image into a collection of relatively small code-blocks, B_i, whose embedded bit-streams may be truncated to rates, R_i^n. The contribution from B_i to distortion in the reconstructed image is denoted D_i^n, for each truncation point, n. We are thus assuming that the relevant distortion metric is additive, i.e.,

$$D = \sum_i D_i^{n_i} \tag{1}$$

where D represents overall image distortion and n_i denotes the truncation point selected for code-block B_i. In our experimental work, two different distortion metrics are considered. An additive distortion metric which approximates Mean Squared Error (MSE) is obtained by setting

$$\hat{D}_i^n = w_{b_i}^2 \sum_{\mathbf{k} \in B_i} (\hat{s}_i^n[\mathbf{k}] - s_i[\mathbf{k}])^2.$$

Here, $s_i[\mathbf{k}]$ denotes the two-dimensional (2-D) sequence of subband samples in code-block B_i, $\hat{s}_i^n[\mathbf{k}]$ denotes the quantized representation of these samples associated with truncation point n, and w_{b_i} denotes the L2-norm of the wavelet basis functions for the subband, b_i, to which code-block B_i belongs. This approximation is valid provided the wavelet transform's basis functions are orthogonal or the quantization errors in each of the samples are uncorrelated. Neither of these requirements is strictly satisfied; however, the wavelet kernels used in our experimental investigations in Section V have nearly orthogonal basis functions. A second distortion metric which correlates more successfully with perceived visual distortion is investigated in Section VI.

We now briefly discuss the optimal selection of the truncation points, n_i, so as to minimize distortion subject to a constraint, $R^{\max}$, on the available bit-rate, i.e.,

$$R^{\max} \geq R = \sum_i R_i^{n_i}. \tag{2}$$

The procedure is not novel [2], but is summarized here for completeness.

It is easy to see that any set of truncation points, $\{n_i^\lambda\}$, which minimizes

$$(D(\lambda) + \lambda R(\lambda)) = \sum_i (D_i^{n_i^\lambda} + \lambda R_i^{n_i^\lambda}) \tag{3}$$

for some λ is optimal in the sense that the distortion cannot be reduced without also increasing the overall rate and vice-versa. Thus, if we can find a value of λ such that the truncation points which minimize (3) yield $R(\lambda) = R^{\max}$, then this set of truncation points must be an optimal solution to our R-D optimization problem. Since we have only a discrete set of truncation points, we will not generally be able to find a value of λ for which $R(\lambda)$ is exactly equal to $R^{\max}$. Nevertheless, since EBCOT's code-blocks are relatively small and there are many truncation points, it is sufficient in practice to find the smallest value of λ such that $R(\lambda) \leq R^{\max}$.

The determination of the optimal truncation points, n_i^λ, for any given λ, may be performed very efficiently, based on a small amount of summary information collected during the generation of each code-block's embedded bit-stream. It is clear that we have a separate minimization problem for each code-block,

B_i. A simple algorithm to find the truncation point, n_i^λ, which minimizes $(D_i^{n_i^\lambda} + \lambda R_i^{n_i^\lambda})$, is as follows:

- initialize $n_i^\lambda = 0$;
- for $j = 1, 2, 3, \cdots$
  set $\Delta R_i^j = R_i^j - R_i^{n_i^\lambda}$ and $\Delta D_i^j = D_i^{n_i^\lambda} - D_i^j$;
  if $\Delta D_i^j / \Delta R_i^j > \lambda$ then update $n_i^\lambda = j$.

Since this algorithm must be executed for many different values of λ, we first find the subset, $\mathcal{N}_i$, of feasible truncation points. Let $j_1 < j_2 < \cdots$ be an enumeration of these feasible truncation points and let the corresponding distortion-rate "slopes" be given by $S_i^{j_k} = (\Delta D_i^{j_k} / \Delta R_i^{j_k}$ where $\Delta R_i^{j_k} = R_i^{j_k} - R_i^{j_{k-1}}$ and $\Delta D_i^{j_k} = D_i^{j_{k-1}} - D_i^{j_k}$. Evidently, the slopes must be strictly decreasing, for if $S_i^{j_{k+1}} \geq S_i^{j_k}$ then the truncation point, j_k, could never be selected by the above algorithm, regardless of the value of λ, contradicting the fact that $\mathcal{N}_i$ is the set of feasible truncation points. When restricted to a set of truncation points whose slopes are strictly decreasing, the above algorithm reduces to the trivial selection $n_i^\lambda = \max\{j_k \in \mathcal{N}_i | S_i^{j_k} > \lambda\}$ so that each such point must be a valid candidate for some value of λ. It follows that $\mathcal{N}_i$ is the largest set of truncation points for which the corresponding distortion-rate slopes are strictly decreasing. This unique set may be determined using a conventional convex hull analysis.

In a typical implementation of the EBCOT algorithm, $\mathcal{N}_i$, is determined immediately after the bit-stream for B_i has been generated. The rates, $R_i^{j_k}$ and slopes, $S_i^{j_k}$, for each $j_k \in \mathcal{N}_i$, are kept in a compact form along with the embedded bit-stream until all code-blocks have been compressed, at which point the search for the optimal λ and n_i^λ proceeds in a straightforward manner. It is worth emphasizing that only rate and slope values must be stored, not the distortion. This requires only a fraction of the storage for the embedded bit-stream itself.

III. Block Coding

In this section, we describe the actual block coding algorithm, which generates a separate embedded bit-stream for each code-block, B_i. The algorithm relies upon the use of classical context adaptive arithmetic coding to efficiently represent a collection of binary symbols. The coder is essentially a bit-plane coder, using similar techniques to those of the LZC algorithm [16]. The key enhancements are: 1) the use of "fractional bit-planes," in which the quantization symbols for any given bit-plane are separated into multiple coding passes; 2) careful reduction of the number of model contexts for arithmetic coding; and 3) the code-block is further partitioned into "sub-blocks," with the significance of each sub-block coded explicitly prior to sample-by-sample coding in the significant sub-blocks. The use of fractional bit-planes is motivated by separate work by Ordentlich *et al.* [10] and by Li and Lei [4]; its purpose is to ensure a sufficiently fine embedding. Another variation on the fractional bit-plane concept was introduced into an early incarnation of the JPEG2000 verification model [15]. The introduction of sub-blocks, with explicit coding of whether or not each sub-block contains at least one significant sample in the relevant bit-plane, is a useful tool for reducing the model adaptation cost as well as implementation complexity. The assumption behind explicit sub-block significance coding is that significant samples tend to be clustered so that the opportunity frequently exists to dispose of a large number of samples by coding a single binary symbol. This is the same assumption which underlies quad-tree and zero-tree coding algorithms as in [14], [13]. In our case, however, we exploit the block-based clustering assumption only down to relatively large sub-blocks of size 16×16, rather than individual samples.

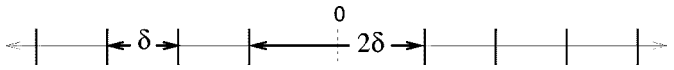


Fig. 4. Deadzone quantizer thresholds.

A. Quantization and Significance

Following the notation of Section II, let $s_i[\mathbf{k}] = s_i[k_1, k_2]$ denote the 2-D sequence of subband samples belonging to code-block B_i. For the LH (vertically high-pass) subband, as well as the HH and LL subbands, k_1 and k_2 denote horizontal and vertical position, respectively. For the HL (horizontally high-pass) subband, however, we first transpose the code-block so that k_1 and k_2 correspond to vertical and horizontal position, respectively, in the original image. This transposition allows us to treat subbands with LH and HL orientation in exactly the same way, thereby simplifying the ensuing description. Let $\chi_i[\mathbf{k}] \in \{1, -1\}$ denote the sign of $s_i[\mathbf{k}]$ and let $\nu_i[\mathbf{k}]$ denote the quantized magnitude, i.e.,

$$\nu_i[\mathbf{k}] = \left\lfloor \frac{|s_i[\mathbf{k}]|}{\delta_{\beta_i}} \right\rfloor$$

where δ_β is the step-size for subband β and β_i is the subband to which code-block B_i belongs. Fig. 4 depicts the thresholds for this so-called "deadzone" quantizer. Evidently the quantizer has uniformly spaced thresholds, except in the interval containing 0, which is twice as large.

Let $\nu_i^p[\mathbf{k}]$ denote the pth bit in the binary representation of $\nu_i[\mathbf{k}]$, where $p = 0$ corresponds to the least significant bit. Also, let $p_i^{\max}$ denote the maximum value of p (i.e. the most significant bit) such that $\nu_i^p[\mathbf{k}] \neq 0$ for at least one sample in the code-block. It turns out to be most efficient to encode the value of $p_i^{\max}$ in the second tier coding algorithm, as described in Section IV. The idea behind bit-plane coding is to encode first the most significant bits, $\nu_i^{p_i^{\max}}[\mathbf{k}]$, for all samples in the code-block, then the next most significant bits, $\nu_i^{p_i^{\max}-1}[\mathbf{k}]$, and so forth until all bit-planes have been encoded. If the bit-stream is truncated then some or all of the samples in the block may be missing one or more least significant bits, which is equivalent to having used a coarser dead-zone quantizer for the relevant samples, with step size $\delta_{\beta_i} 2^p$, where p is the index of the last available bit-plane for the relevant sample.

In order to efficiently encode $\nu_i^p[\mathbf{k}]$, it is important to exploit previously encoded information about the same sample and neighboring samples. We do this primarily by means of a binary-valued state variable, $\sigma_i[\mathbf{k}]$, which is initialized to 0, but transitions to 1 when the relevant sample's first nonzero bit-plane, $\nu_i^p[\mathbf{k}] = 1$, is encoded. We refer to the state, $\sigma_i[\mathbf{k}]$, as the sample's "significance." The point at which a sample

becomes significant depends intimately upon the sequence in which sample values are encoded, which is the subject of Section III-D.

B. Sub-Block Significance Coding Front-End

Each block, B_i, is partitioned into a 2-D sequence of sub-blocks, $B_i[\mathbf{j}]$, whose size is typically 16×16. For each bit-plane, $p_i^{\max} \geq p \geq 0$, we first encode information to identify those sub-blocks which contain one or more significant samples; all other sub-blocks are by-passed in the remaining coding phases for that bit-plane, which reduces complexity as well as the cost of adapting probability models to highly skewed statistics. Let $\sigma^p(B_i[\mathbf{j}])$ denote the significance of sub-block $B_i[\mathbf{j}]$, in bit-plane p. That is, $\sigma^p(B_i[\mathbf{j}]) = 1$ if and only if $\nu_i[\mathbf{k}] \geq 2^p$ for some $\mathbf{k} \in B_i[\mathbf{j}]$; otherwise, $\sigma^p(B_i[\mathbf{j}]) = 0$. There are a variety of ways to encode the values, $\sigma^p(B_i[\mathbf{j}])$, in each successive bit-plane, of which one of the most obvious is quad-tree coding.

In brief, we introduce a tree structure by identifying the sub-blocks with the leaf nodes, i.e. $B_i^0[\mathbf{j}] = B_i[\mathbf{j}]$, and defining higher levels in the tree according to $B_i^t[\mathbf{j}] = \cup_{\mathbf{z} \in \{0,1\}^2} B_i^{t-1}[2\mathbf{j} + \mathbf{z}]$, $0 \leq t \leq T$. At the root of the tree, we have $B_i^T[\mathbf{0}] = \cup_{\mathbf{j}} B_i[\mathbf{j}]$, representing the entire code-block. In any given bit-plane, p, the significance of the quads, $\sigma^p(B_i^t[\mathbf{j}])$, is identified one level at a time, starting from the root at $t = T$ and working to the leaves at $t = 0$. In our implementation, these binary significance symbols are sent to the arithmetic coding engine as uniformly distributed symbols without any adaptive model whatsoever; however redundant symbols are always skipped. A symbol, $\sigma^p(B_i^t[\mathbf{j}])$, is redundant if any of the following conditions holds: 1) the parent quad, if any, is insignificant, i.e., $\sigma^p(B_i^{t+1}[\lfloor j_1/2 \rfloor, \lfloor j_2/2 \rfloor]) = 0$; 2) the quad was significant in the previous bit-plane, i.e., $\sigma^{p+1}(B_i^t[\mathbf{j}]) = 1$; or 3) this is the last quad visited amongst those which share the same, significant parent and the other siblings are all insignificant.

C. Bit-Plane Coding Primitives

The purpose of this section is to describe the four different primitive coding operations which form the foundation of the embedded block coding strategy. The primitives are used to code new information for a single sample in some bit-plane, p. If the sample is not yet significant, i.e. $\sigma_i[\mathbf{k}] = 0$, a combination of the "zero coding" (ZC) and "run-length coding" (RLC) primitives is used to code whether or not the symbol becomes significant in the current bit-plane; if so, the "sign coding" (SC) primitive must also be invoked to identify the sign, $\chi_i[\mathbf{k}]$. If the sample is already significant, the "magnitude refinement" (MR) primitive is used to encode the value of $\nu_i^p[\mathbf{k}]$. In every case, a single binary-valued symbol must be coded using a common arithmetic coding engine. The probability models used by the arithmetic coder evolve within 18 different contexts: nine for the ZC primitive; one for the RLC primitive; five for the ZC primitive; and three for the MR primitive.

1) Zero Coding (ZC): The objective here is to code $\nu_i^p[\mathbf{k}]$, given that $\nu_i[\mathbf{k}] < 2^{p+1}$. Empirical evidence suggests that the sample statistics are approximately Markov: the significance of sample $s_i[\mathbf{k}]$ depends only upon the values of its immediate eight neighbors. In fact, almost all the relevant information appears to be captured by the significance of these neighbors, which we group into the following three categories:

horizontal: we write $h_i[\mathbf{k}] = \Sigma_{z \in \{1,-1\}} \sigma_i[k_1 + z, k_2]$ so that $0 \leq h_i[\mathbf{k}] \leq 2$;

vertical: we write $v_i[\mathbf{k}] = \Sigma_{z \in \{1,-1\}} \sigma_i[k_1, k_2 + z]$ so that $0 \leq v_i[\mathbf{k}] \leq 2$;

diagonal: we write $d_i[\mathbf{k}] = \Sigma_{z_1, z_2 \in \{1,-1\}} \sigma_i[k_1 + z_1, k_2 + z_2]$ so that $0 \leq d_i[\mathbf{k}] \leq 4$.

Neighbors which lie outside the code-block are interpreted as insignificant, so as to ensure that the block bit-streams are truly independent. No such assumption is imposed on neighbors which lie outside the relevant sub-block, however, so that sub-blocks are by no means independent.

TABLE I
ASSIGNMENT OF THE NINE ZC CONTEXTS BASED ON NEIGHBOURHOOD SIGNIFICANCE

LL, LH and HL bands				HH band		
$h_i[\mathbf{k}]$	$v_i[\mathbf{k}]$	$d_i[\mathbf{k}]$	Label	$d_i[\mathbf{k}]$	$h_i[\mathbf{k}] + v_i[\mathbf{k}]$	Label
0	0	0	0	0	0	0
0	0	1	1	0	1	1
0	0	>1	2	0	>1	2
0	1	x	3	1	0	3
0	2	x	4	1	1	4
1	0	0	5	1	>1	5
1	0	>0	6	2	0	6
1	>0	x	7	2	>0	7
2	x	x	8	>2	x	8

To minimize both model adaptation cost and implementation complexity, we quantize the 256 possible neighborhood configurations to nine distinct coding contexts, with the labels indicated in Table I. The context assignment for the LH and HL bands is identical, because the HL (horizontally high-pass) subband's code-blocks are transposed, as explained in Section III-A. The LH (vertically high-pass) subband responds most strongly to horizontal edges in the original image, so we expect strong correlation amongst horizontally adjacent samples; this explains the emphasis on horizontal neighbors in the first three rows of the table.

2) Run-Length Coding (RLC): The RLC primitive is used to reduce the average number of binary symbols which must be processed by the arithmetic coding engine. It is invoked in place of the ZC primitive when a horizontal run of insignificant samples is encountered whose immediate neighbors are also all insignificant. Specifically, each of the following conditions must hold: 1) four consecutive samples must all be insignificant, i.e., $\sigma[k_1 + z, k_2] = 0$, for $0 \leq z \leq 3$; 2) the samples must have insignificant neighbors, i.e., $h_i[k_1 + z, k_2] = v_i[k_1 + z, k_2] = d_i[k_1 + z, k_2] = 0$; 3) the samples must reside within the same sub-block; and 4) the horizontal index of the first sample, k_1, must be even. The last two conditions are enforced only to facilitate efficient implementations of the symbol grouping scheme.

When a group of four samples satisfies the above conditions, a single symbol is encoded to identify whether any sample in the group is significant in the current bit-plane. A separate context is used to model the distribution of this symbol. If any of the four samples becomes significant, i.e., $\nu_i^p[k_1 + z, k_2] \neq 0$, the zero-based index, z, of the first such sample is sent as a two-bit quantity with a nonadaptive, uniform probability model.

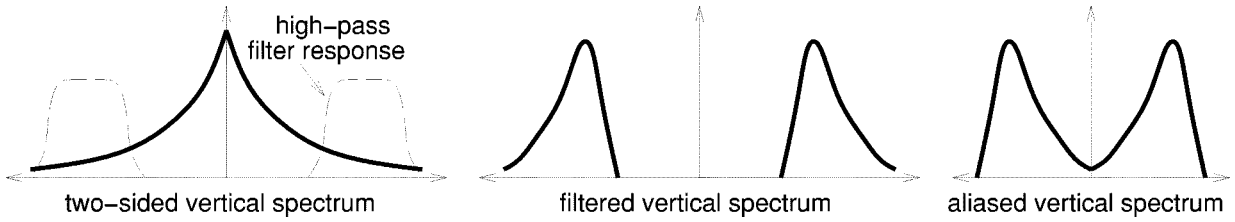


Fig. 5. Typical vertical spectra traced from the image domain to the vertically high-pass subbands.

This is reasonable subject to the assumption that the probability of any sample from the group becoming significant is very small; in this case, the conditional distribution of the run-length, $z \in \{0, 1, 2, 3\}$, given the significance of at least one sample in the group, is approximately uniform. Empirically, we observe that the introduction of the RLC primitive tends to slightly improve compression efficiency; its main role, however, is to substantially reduce the number of symbols which must be coded and hence implementation complexity.

3) Sign Coding (SC): The SC primitive is used at most once for each sample, immediately after a previously insignificant sample is first found to be significant during a ZC or RLC operation. It turns out that the sign bits, $\chi_i[\mathbf{k}]$, from adjacent samples, exhibit substantial statistical dependencies which can be effectively exploited to improve coding efficiency. To understand this, consider the LH (vertically high-pass) subbands. We claim that horizontally adjacent samples from LH subbands tend to have the same sign, whereas vertically adjacent samples tend to have opposite signs. Equivalently, the LH subband samples have predominantly low-pass horizontal power spectra and high-pass vertical power spectra. In the horizontal direction, this is entirely reasonable, since images typically have low-pass spectra which are preserved by the horizontal low-pass filtering and decimation operations used to generate the LH subbands.

In the vertical direction, the aliasing introduced by the high-pass filtering and decimation operations leads to the opposite conclusion. Fig. 5 illustrates the effect of these operations on the vertical spectrum of a typical image. Again, images typically have low-pass spectra; even sharp horizontal edges yield spectra whose amplitude decreases in inverse proportion to the vertical frequency. After high-pass filtering, then, the vertical spectrum typically exhibits more energy at the lower end of the pass band. Finally, the aliasing associated with vertical decimation reverses this trend, so that the actual subband samples are primarily high-pass in the vertical direction, which substantiates our claim.

To exploit this redundancy in the sign information, we use five model contexts for coding $\chi_i[\mathbf{k}]$, according to the available information concerning the signs of the immediate horizontal and vertical neighbors. Since there are four such neighbors, each of which may be insignificant, positive or negative, there would appear to be $3^4 = 81$ unique neighborhood configurations. However, two inherent symmetry properties dramatically reduce this number: there is no reason not to expect horizontal and vertical symmetry amongst the neighborhoods; and the conditional distribution of $\chi_i[\mathbf{k}]$ given any particular neighborhood should be identical to the conditional distribution of $-\chi_i[\mathbf{k}]$, given the dual neighborhood with the signs of all neighbors reversed. Taking these symmetries into account, it is not difficult to show that the number of unique conditional distributions is at most 13.

TABLE II
ASSIGNMENT OF THE FIVE SC CONTEXTS BASED ON THE SIGNS OF SIGNIFICANT HORIZONTAL AND VERTICAL NEIGHBORS

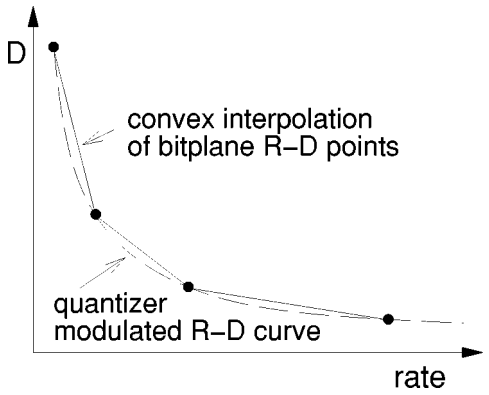
$\bar{h}_i[\mathbf{k}]$	$\bar{v}_i[\mathbf{k}]$	$\hat{\chi}[\mathbf{k}]$	Label
1	1	1	4
1	0	1	3
1	−1	1	2
0	1	−1	1
0	0	1	0
0	−1	1	1
−1	1	−1	2
−1	0	−1	3
−1	−1	−1	4

We further reduce this to five contexts by not distinguishing the case in which opposite neighbors are both significant with the same sign. Let $\overline{h}_i[\mathbf{k}]$ equal 0 if both horizontal neighbors are insignificant or both are significant with different signs, 1 if at least one of the horizontal neighbors is positive and -1 if at least one of the horizontal neighbors is negative. Define $\overline{v}_i[\mathbf{k}]$ in similar fashion for the vertical neighbors. Table II identifies the five unique probability contexts formed by $\overline{h}_i[\mathbf{k}]$ and $\overline{v}_i[\mathbf{k}]$, along with the sign prediction, $\hat{\chi}_i[\mathbf{k}]$, which is used to exploit the second type of symmetry mentioned above. The binary valued symbol which is coded with respect to the relevant context is $\chi_i[\mathbf{k}] \cdot \hat{\chi}_i[\mathbf{k}]$. In view of the transposition of code-blocks from the HL subbands, the same strategy may be applied to the LH and HL subbands; for convenience, it is extended without modification to the less important LL and HH subbands as well.

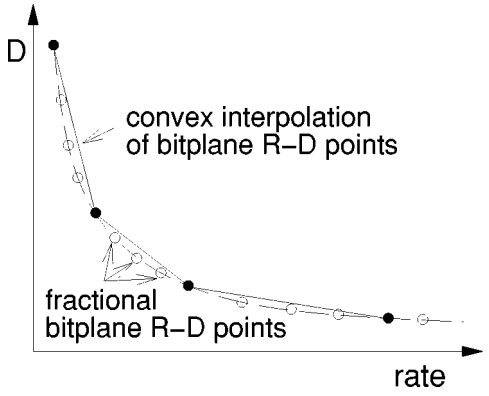
4) Magnitude Refinement (MR): The objective here is to code the value of $\nu_i^p[\mathbf{k}]$, given that $\nu_i[\mathbf{k}] \geq 2^{p+1}$. Experience shows that the conditional distribution of $\nu_i^p[\mathbf{k}]$ is only weakly dependent on the magnitude, $\lfloor 2^{-p-1}\nu_i[\mathbf{k}] \rfloor$, represented by the previously encoded bit-planes and also only weakly dependent on the magnitude of neigbouring samples. For this reason, we use only three model contexts for magnitude refinement. It is convenient to introduce a second state variable, $\tilde{\sigma}_i[\mathbf{k}]$, which transitions from 0 to 1 after the the MR primitive is first applied to $s_i[\mathbf{k}]$. The magnitude refinement context depends upon the value of $\tilde{\sigma}_i[\mathbf{k}]$ and also on whether or not any immediate horizontal or vertical neighbors are significant. Specifically, $\nu_i^p[\mathbf{k}]$ is coded with context 0 if $\tilde{\sigma}_i[\mathbf{k}] = h_i[\mathbf{k}] = v_i[\mathbf{k}] = 0$, with context 1 if $\tilde{\sigma}_i[\mathbf{k}] = 0$ and $h_i[\mathbf{k}] + v_i[\mathbf{k}] \neq 0$, and with context 2 if $\tilde{\sigma}_i[\mathbf{k}] = 1$.

D. Fractional Bit-Planes and Scanning Order

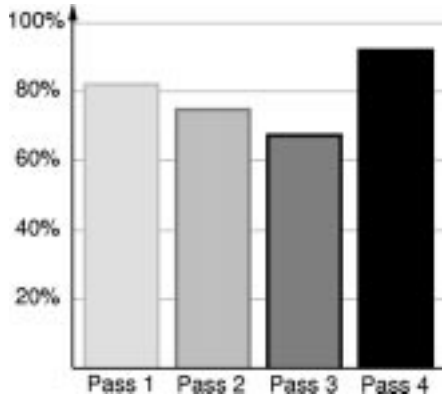
For each bit-plane, p, the coding proceeds in a number of distinct passes, which we identify as "fractional bit-planes." In this work we consider a total of four such passes, $\mathcal{P}_1^p$, $\mathcal{P}_2^p$, $\mathcal{P}_3^p$ and $\mathcal{P}_4^p$ and we identify the truncation available points with these



(a)



(b)



(c)

Fig. 6. Rate-distortion properties of (a) regular bit-plane coding and (b) fractional bit-plane coding. (c) shows the percentage of code-block bit-planes in which each of the four EBCOT coding passes yields a point on the convex hull of the rate-distortion curve. Data are obtained by averaging results at 1 bpp from the three most popular images in the JPEG2000 test set: "bike," "cafe," and "woman," each of which measures 2560 × 2048.

coding passes, so that R_i^n is the number of leading bytes from the arithmetic code word, which are required to uniquely decode the symbols in the first n fractional bit-plane coding passes. The reason for introducing multiple coding passes is to ensure that each code-block has a finely embedded bit-stream.

Fig. 6 is helpful in understanding the goals and benefits of fractional bit-plane coding. The dashed line in Fig. 6(a) identifies the rate-distortion curve described by modulating the quantization step size and decoding all bit-planes. It is important to note that this curve is almost invariably convex. The solid dots in Fig. 6(a) identify the rate and distortion associated with the end of each coding pass of a conventional bit-plane coder. These points lie on the dashed rate-distortion curve because discarding the last p bit-planes from the bit-stream is equivalent to multiplying the quantization step size by 2^p; neither the distortion nor the information content and hence entropy are affected by whether we discard bit-planes or scale the step size. The solid lines in Fig. 6(a) illustrate the rate-distortion curve which one could expect to obtain by truncating the bit-stream produced by a conventional bit-plane coder to an arbitrary bit-rate. Suppose (R_1, D_1) and (R_2, D_2) are the rate-distortion pairs corresponding to two adjacent bit-planes, p_1 and p_2. If we truncate to some arbitrary bit-rate, $R_1 < R < R_2$, so that a fraction, t, of the samples is refined to bit-plane p_2 and the remainder are available only at bit-plane p_1, then we expect to find $R = R_1 + t(R_2 - R_1)$ and $D = D_1 + t(D_2 - D_1)$, because there is no reason to suppose that the initial samples coded in each bit-plane pass exhibit different statistics to later samples. Consequently, the solid lines in Fig. 6(a) are simply the convex interpolation of the R-D points corresponding to whole bit-planes. This is necessarily sub-optimal with respect to the convex dashed rate-distortion curve.

On the other hand, if we separate the code-block samples into smaller subsets with different statistics, then it is possible to improve upon this behavior by coding the next bit-plane one subset at a time, starting with the subsets which are expected to offer the largest reduction in distortion for each extra bit in the code length. This de-interleaving of the samples into subsets with distinct statistics is the goal of fractional bit-plane coding. In the present work, there are four subsets corresponding to the four coding passes and the end of the fourth coding pass, $\mathcal{P}_4^p$, marks the point at which all samples have been updated to bit-plane p. Thus, the solid dots in Fig. 6(a) and (b) may be associated with this coding pass. Since the initial coding passes generally have steeper rate-distortion slopes, the end points of each coding pass lie below the convex interpolation of the bit-plane termination points, as indicated in the figure.

The rate-distortion points corresponding to the various fractional bit-plane coding passes can even lie below the dashed line in Fig. 6(a). Fig. 6(c) provides empirical evidence for this. Recall from Section II that the candidate truncation points for a given embedded bit-stream are those which lie on the convex hull of the rate-distortion curve described by all available truncation points. Fig. 6(c) clearly shows that each of the four coding passes frequently generates a point on this convex hull; moreover, the rate-distortion points corresponding to fully coded bit-planes at the end of coding pass $\mathcal{P}_4^p$, do not always lie on the convex hull, so that other passes occasionally yield superior rate-distortion performance to that which is achievable by coding all samples with a fixed quantization step size.

Fig. 7 provides a helpful illustration of the appearance of information within the embedded bit-stream generated for each code-block. Here, $\mathcal{S}^p$ denotes the quad-tree code which identifies which sub-blocks are significant in bit-plane p. Notice that $\mathcal{S}^p$ appears immediately before the final coding pass, $\mathcal{P}_4^p$, not the initial coding pass, $\mathcal{P}_1^p$, for the bit-plane. This means that sub-blocks which become significant for the first time in bit-plane p, are ignored until pass $\mathcal{P}_4^p$. We now define the roles played by each coding pass.

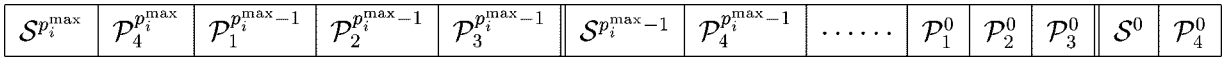


Fig. 7. Appearance of coding passes and quad-tree codes in each block's embedded bit-stream.

1) "Forward Significance Propagation Pass," $\mathcal{P}_1^p$: In this pass, we visit the sub-block samples in scan-line order, skipping over all samples which are either insignificant or do not have a so-called "preferred neighborhood." For the LH and HL subbands, sample $s_i[\mathbf{k}]$ is said to have a preferred neighborhood if at least one of its horizontal neighbors is significant, i.e., $\sigma[k_1 \pm 1, k_2] = 1$. Recall that HL subband code-blocks are transposed so that both LH and HL subbands tend to contain horizontal line segments. The LL subband is treated in the same way for convenience, while the HH subbands' samples are said to have a preferred neighborhood if one or more of the four diagonal neighbors is significant, i.e., $\sigma[k_1 \pm 1, k_2 \pm 1] = 1$. To each such sample, we apply the ZC and RLC primitives, as appropriate, to identify whether or not the sample first becomes significant in bit-plane p; if so, we invoke the SC primitive to code its sign. We call this the "forward significance propagation pass" because samples which have been found to be significant typically serve as seeds for waves of new significance determination steps which propagate in the direction of the scan.

2) "Reverse Significance Propagation Pass," $\mathcal{P}_2^p$: This coding pass is identical to $\mathcal{P}_1^p$, except that the samples are visited in the reverse order and the notion of a preferred neighborhood is expanded to encompass samples for which at least one of the eight immediate neighbors has already been found to be significant. Of course, we skip over samples for which information was coded in the previous pass.

3) "Magnitude Refinement Pass," $\mathcal{P}_3^p$: During this pass we skip over all samples except those which are already significant, i.e. $\sigma_i[\mathbf{k}] = 1$, and for which no information has been coded in the previous two passes. These samples are processed with the MR primitive.

4) "Normalization Pass," $\mathcal{P}_4^p$: Here we code the least significant bit, $\nu_i^p[\mathbf{k}]$, of all samples not considered in the previous three coding passes, using the SC and RLC primitives as appropriate; if a sample is found to be significant in this process, its sign is coded immediately using the SC primitive.

IV. Layer Formation and Representation

In this section we consider the second tier coding engine of Fig. 3, which is responsible for efficiently identifying the contribution of each code-block to each bit-stream layer, along with other summary information for the code-blocks. Recall that the final bit-stream is composed of a collection of quality layers, $\mathcal{Q}_q$. Together, layers $\mathcal{Q}_1$ through $\mathcal{Q}_q$ contain the initial $R_i^{n_i^q}$ bytes of each code-block, B_i. Here, we write n_i^q for the truncation point selected for the qth quality layer, with some abuse of the notation established in Section II where n_i^λ denotes the R-D optimal truncation point corresponding to a threshold of λ on the distortion-rate slope. Thus, n_i^q is short-hand for $n_i^{\lambda_q}$ with λ_q denoting the distortion-rate threshold selected for layer $\mathcal{Q}_q$. Layer $\mathcal{Q}_q$ contains the incremental contribution, $L_i^q = R_i^{n_i^q} - R_i^{n_i^{q-1}} \geq 0$, from code-block B_i. As illustrated in Fig. 2, some code-blocks might contribute no bytes at all to some layers. Along with these incremental contributions, the length of the segment, L_i^q, and the number of new coding passes, $N_i^q = n_i^q - n_i^{q-1}$, in the segment must be explicitly identified. Finally, if B_i makes its first nonempty contribution to quality layer $\mathcal{Q}_q$ then the most significant bit-plane, $p_i^{\max}$, must also be identified, as mentioned in Section III-A.

We focus our description on the two quantities which exhibit substantial inter-block redundancy and show how this redundancy is exploited within the second tier coding engine; full details may be found in [19], [1]. These two quantities are $p_i^{\max}$ and the index, q_i, of the quality layer to which B_i first makes a nonempty contribution. The latter quantity, q_i, is encoded using a separate embedded quad-tree code within each subband as follows. Let B_i^t denote the sequence of quads at level t in the quad-tree, with $B_i^0 = B_i$ denoting the leaves and B_0^T the root of the tree, representing the entire subband. Let q_i^t be the index of the first layer in which any code-block in quad B_i^t makes a nonempty contribution, i.e. $q_i^t = \min\{q_j | B_j \subset B_i^t\}$. In each quality layer, $\mathcal{Q}_q$, a binary quad-tree code is used to identify whether or not $q_i > q$. That is, a single bit is used to identify whether or not $q_i^t > q$ for each quad at each level, t, in the tree, skipping over quads for which the value of this bit may be inferred by the decoder, for one of the following two reasons: 1) $q_i^t \leq q-1$, in which case the value of q_i^t has already been identified in a previous quality layer; or 2) $q_j^{t+1} > q$ where B_i^t belongs to the parent quad, B_j^{t+1}, in which case we must have $q_i^t > q$. To see how this code exploits inter-block redundancy, consider an initial set of lowest quality layers which correspond to very low bit-rates; in this case, it is reasonable to suppose that none of the code-blocks from the highest frequency subbands have sufficiently steep distortion-rate slopes to make any contribution to these layers. The quad-tree code for each such subband consists of a single 0 bit for each of these empty layers. More generally, the distortion-rate slopes for individual code-blocks depend upon local image statistics; if these statistics vary slowly over the image then neighboring code-blocks should have similar or identical q_i values.

The other quantity which exhibits substantial inter-block redundancy is $p_i^{\max}$. One might consider using a similar embedded quad-tree code to represent this quantity. However, the value of $p_i^{\max}$ is irrelevant until the quality layer $\mathcal{Q}_{q_i}$ in which the code-block first makes a contribution to the bit-stream and the code-blocks in any given subband do not generally all make their first contribution in the same quality layer. The embedding principle suggests that we should avoid sending any unnecessary information concerning $p_i^{\max}$ until layer $\mathcal{Q}_{q_i}$. EBCOT achieves this, while still exploiting inter-block redundancy in coding the $p_i^{\max}$ values, by means of a modified embedded quad-tree, which is driven from the leaves rather than the root of the tree. Specifically, let B_i^t denote the elements of the quad-tree structure built on top of the

code-blocks, B_i, from any given subband, exactly as described above; define $p_i^{\max,\,t} = \max\{p_j^{\max} | B_j \subset B_i^t\}$. Also, let $B_{i_t}^t$ denote the ancestry of quads from which B_i is descended, so that $\{B_i\} = B_{i_0}^0 \subset B_{i_1}^1 \subset \cdots \subset B_{i_T}^T = B_0^T$. Let P be a value which is guaranteed to be larger than $p_i^{\max}$ for any code-block, B_i, in the relevant subband. When code-block B_i first contributes to the bit-stream in quality layer $\mathcal{Q}_{q_i}$, we code the value of $p_i^{\max} = p_{i_0}^{\max,0}$ using the following algorithm.

- For $p = P-1, P-2, \cdots, 0$
  Send binary digits to identify whether or not $p_{i_t}^{\max,\,t} < p$ for $t = T, \cdots, 1, 0$, skipping all redundant bits. If $p_i^{\max} = p$ then stop.

The redundant bits mentioned above are those corresponding to conditions $p_{i_t}^{\max,\,t} < p$ which can be inferred either from previously coded conditions in the same partial quad-tree scan, i.e. if $p_{i_{t+1}}^{\max,\,t+1} < p$, or from the partial quad-tree code which was used to identify $p_j^{\max}$ for a different code-block, B_j.

In this way, we delay sending information for any condition, $p_i^{\max,\,t} < p$, which is not relevant to the code-blocks which are contributing for the first time to the quality layer at hand. As one might expect, efficient implementation strategies exist for this leaf-driven embedded quad-tree coding algorithm. At first glance, our policy of exploiting inter-block redundancy through a second tier coding engine would appear to interfere with the random access property mentioned in Section I, since code-blocks are no longer strictly independent. However, the second tier coding engine operates only on summary information for whole code-blocks, rather than individual samples, so that the second tier decoding process is best viewed as a somewhat elaborated parser for recovering pointers to code-block segments in the bit-stream.

V. Numerical Results

Table III provides numerical results to illustrate the performance of the proposed EBCOT algorithm under a variety of conditions. Results are presented for the well-known USC images, "Lenna" and "Barbara," as well as the three most popular test images from the JPEG2000 test suite, "bike," "cafe," and "woman," which are substantially more complex and less blurred than the USC images. The first column of PSNR results corresponds to the well known SPIHT [13] algorithm with arithmetic coding. The remaining columns are obtained with the EBCOT algorithm, running within the framework of JPEG2000 Verification Model VM3A [20]. In all cases, we use the popular Daubechies 9/7 bi-orthogonal wavelet filters with a five level transform. For EBCOT we use code-blocks of size 64×64 with sub-blocks of size 16×16.

Recall that the EBCOT bit-stream is composed of a collection of quality layers and that SNR scalability is obtained by discarding unwanted layers. The second column in the table corresponds to a bit-stream with only one layer, so that the overall bit-stream is not SNR scalable. Results in this case are obtained by generating a separate compressed bit-stream for each of the relevant bit-rates. Each of the remaining columns are obtained by truncating a single bit-stream to the relevant bit-rates. The third column corresponds to a limited form of SNR scalability in which there are only five quality layers, optimized for each of the target bit-rates in the table; this may be sufficient for some applications. The fourth column corresponds to the extreme case in which 50 separate layers are included in the bit-stream spanning bit-rates ranging from approximately 0.05 bpp to 2.0 bpp; in this case, the layer bit-rates are spaced approximately logarithmically through this range by selecting an appropriate set of distortion-rate slope parameters, λ_q, but no rate-control iteration is performed to adjust the λ_q values for specific target bit-rates.

As might be expected, performance decreases as more layers are added to the bit-stream, because the overhead associated with identifying the contributions of each code-block to each layer grows. Nevertheless, performance continues to be competitive with respect to state-of-the-art compression algorithms, significantly outperforming the common reference, SPIHT. All results for the EBCOT algorithm are obtained using a single quantization step size, regardless of the image or bit-rate, with rate control implemented exclusively through truncation of the embedded block bit-streams. For the smaller images, at very low bit-rates the EBCOT results are slightly penalized by the 59 byte header included by the JPEG2000 VM3A software [20]. Some of the results appear to be counter-intuitive. Specifically, at 0.25 bpp, the performance for "Lenna" with five layers appears to be marginally higher than that with only one layer, even though the overhead associated with signaling five layers is undoubtedly

TABLE III
PSNR Results, Measured in dB, for Various Images and Bit-Rates

Lenna (512 × 512)					
Bit Rate	Spiht	1 layer	5 layer	Generic	Spacl
0.0625	28.38	28.30	28.30	28.10	28.27
0.125	31.10	31.22	31.20	31.05	31.22
0.25	34.11	34.28	34.29	34.16	34.40
0.5	37.21	37.43	37.41	37.29	37.49
1.0	40.41	40.61	40.57	40.48	40.49
Barbara (512 × 512)					
Bit Rate	Spiht	1 layer	5 layer	Generic	Spacl
0.0625	23.35	23.45	23.49	23.34	23.79
0.125	24.86	25.55	25.52	25.37	25.95
0.25	27.58	28.55	28.51	28.40	29.03
0.5	31.39	32.48	32.43	32.29	33.06
1.0	36.41	37.37	37.32	37.11	37.87
Bike (2560 × 2048)					
Bit Rate	Spiht	1 layer	5 layer	Generic	Spacl
0.0625	23.44	23.89	23.88	23.78	24.05
0.125	25.89	26.49	26.47	26.37	26.71
0.25	29.12	29.76	29.73	29.60	29.91
0.5	33.01	33.68	33.64	33.46	33.65
1.0	37.70	38.29	38.25	38.09	38.02
Cafe (2560 × 2048)					
Bit Rate	Spiht	1 layer	5 layer	Generic	Spacl
0.0625	18.95	19.10	19.10	19.06	19.13
0.125	20.67	20.88	20.87	20.82	20.99
0.25	23.03	23.29	23.26	23.20	23.47
0.5	26.49	27.00	26.97	26.87	27.10
1.0	31.74	32.27	32.24	32.03	32.06
Woman (2560 × 2048)					
Bit Rate	Spiht	1 layer	5 layer	Generic	Spacl
0.0625	25.43	25.67	25.67	25.63	25.70
0.125	27.33	27.46	27.45	27.39	27.53
0.25	29.95	30.15	30.13	30.04	30.23
0.5	33.59	33.81	33.78	33.70	33.86
1.0	38.28	38.67	38.63	38.49	38.54

higher. Similar behavior is observed with the "Barbara" image. The explanation for this lies with the observation in Section II that it is not generally possible to find a distortion-rate threshold, λ, which satisfies the rate constraint exactly. As a result, the portion of the bit-stream which is actually decoded in SNR progressive tests sometimes includes part of the next quality layer after that which was optimized for the target bit-rate, as in Fig. 2.

Whereas the first four columns of PSNR results in Table III are obtained using a five-level Mallat decomposition of the form shown in Fig. 1(a), the fifth column corresponds to a five-level decomposition of the form shown in Fig. 1(b), using five targeted bit-stream layers as for the second column. This has been coined the "Spacl" decomposition within the JPEG2000 community. Evidently, this decomposition structure typically leads to lower MSE distortion (higher PSNR) at all but the highest bit-rates. We point out that tree-based coders such as SPIHT cannot readily be adapted to non-Mallat decomposition structures.

VI. Visual Distortion Metrics and Performance

The numerical results in Table III are obtained with MSE as the distortion metric for the PCRD optimization algorithm of Section II. It is well known that MSE is a poor model for the visual significance of distortion. Various authors (e.g., [7], [8]) have considered the relatively straightforward extension to frequency weighted MSE, in which the overall image distortion is taken to be a weighted sum of the MSE contributions from each subband, with the weights derived from studies of the contrast sensitivity function (CSF). These approaches have two notable drawbacks: the weights depend strongly upon the angle subtended by each reconstructed pixel at an assumed viewing distance; and the model fails to account for the substantial impact of visual masking effects. In fact, the CSF accounts primarily for the modulation transfer function (MTF) of the physical properties of the optics and aperture of the cones in the human eye; the MTF of the relevant display device is often also incorporated.

More generally, we may consider spatially varying distortion metrics which attempt to exploit the masking phenomenon. Watson's work [22] on visual optimization of JPEG compressed images is noteworthy in this regard, as is the work of Höntsch and Karam [3]. In these and other previous works, visual masking effects must be taken into account by explicitly modifying the quantization parameters; scalable compression is not considered; and rate-control must be performed iteratively.

The EBCOT algorithm provides an excellent context within which masking phenomena can be exploited without substantially increasing computational complexity or sacrificing other properties such as random access or scalability. In our studies, the following distortion metric has been found to yield significantly superior visual image quality than the MSE metric

$$D_i^n = w_{b_i}^2 \sum_{\mathbf{k}} \frac{(\hat{s}_i^n[\mathbf{k}] - s_i[\mathbf{k}])^2}{\sigma_{b_i}^2 + (V_i[j,k])^2}. \tag{4}$$

Here w_{b_i}, $s_i[\mathbf{k}]$ and $\hat{s}_i^n[\mathbf{k}]$ are all as in Section II, $V_i[\mathbf{k}]$ is the "visual masking strength" at sample $s_i[\mathbf{k}]$ and σ_{b_i} is a "visibility floor" term which establishes the visual significance of distortion in the absence of masking.

The proposed visual masking strength operator, $V_i[\mathbf{k}]$, has the form

$$V_i[\mathbf{k}] = \frac{\sum_{\mathbf{k}' \in \Phi_i[\mathbf{k}]} |s_i[\mathbf{k}']|^\rho}{||\Phi_i[\mathbf{k}]||}. \tag{5}$$

Here $\Phi_i[\mathbf{k}]$ denotes a neighborhood of samples about $s_i[\mathbf{k}]$ and $||\Phi_i[\mathbf{k}]||$ denotes the size of this neighborhood. The nonlinear operation is to be understood within the context of normalized image samples with a range of 0 to 1 and a normalized wavelet transform whose low-pass and high-pass analysis filters have unit gain at DC and the Nyquist frequency, respectively, so that $0 \le V_i[\mathbf{k}] \le 1$. For our experiments, the exponent ρ is set to $\frac{1}{2}$, and the neighborhoods, $\Phi_i[\mathbf{k}]$, are obtained by partitioning the code-block into 8×8 cells, using the same masking strength value for all samples in any given cell. This reduces the complexity of computing the visual distortion metric to a small fraction of that for the entire encoder. Our formulation is closely related to the models used in [22] and [3] in which $\rho = 0.7$ and $\rho = 0.6$, respectively; in our experiments, $\rho = \frac{1}{2}$ appears to give superior visual performance. We use a single small visibility floor, σ_b, of 10^{-2} for all subbands, so that the distortion metric is rendered independent of any assumptions on the viewing distance, which is highly desirable for uncontrolled viewing conditions.

Not surprisingly, this visual masking metric has a greater effect on image quality when the code-blocks are small; we find the best performance over a wide range of images when using 32×32 code-blocks. Fig. 8 provides a comparison of SPIHT [13] with arithmetic coding against EBCOT, operating in the context of JPEG2000 VM3A with this visual distortion metric. When optimizing for MSE alone, the visual quality of EBCOT compressed images is very similar to that of SPIHT. Although only small segments from the 2560×2048 image "woman" can be reproduced here, we note that quality is uniformly improved over the entire image. The EBCOT images exhibit substantially less ringing around edges and superior rendition of texture; some details preserved in the EBCOT images are completely lost by the SPIHT algorithm. In fact, for this image we find that the image quality obtained using EBCOT at 0.2 bpp is comparable to that obtained using SPIHT at 0.4 bpp. Similar results are observed with other large images having comparable content, although the method is less effective with some image types. Although we make no assumptions here concerning viewing distance, other studies [8] have shown that the visual masking metric outlined here can be successfully combined with an appropriate global compensation for known CSF characteristics, yielding complementary improvements in quality.

VII. Utility of the Random Access Attribute

Since EBCOT partitions each subband into relatively small code-blocks and codes each of them independently, it would appear to be suited to applications requiring some degree of "random access" into the image. At one extreme we may consider applications which intend to decompress the entire image, but in a different order to that in which it was compressed,

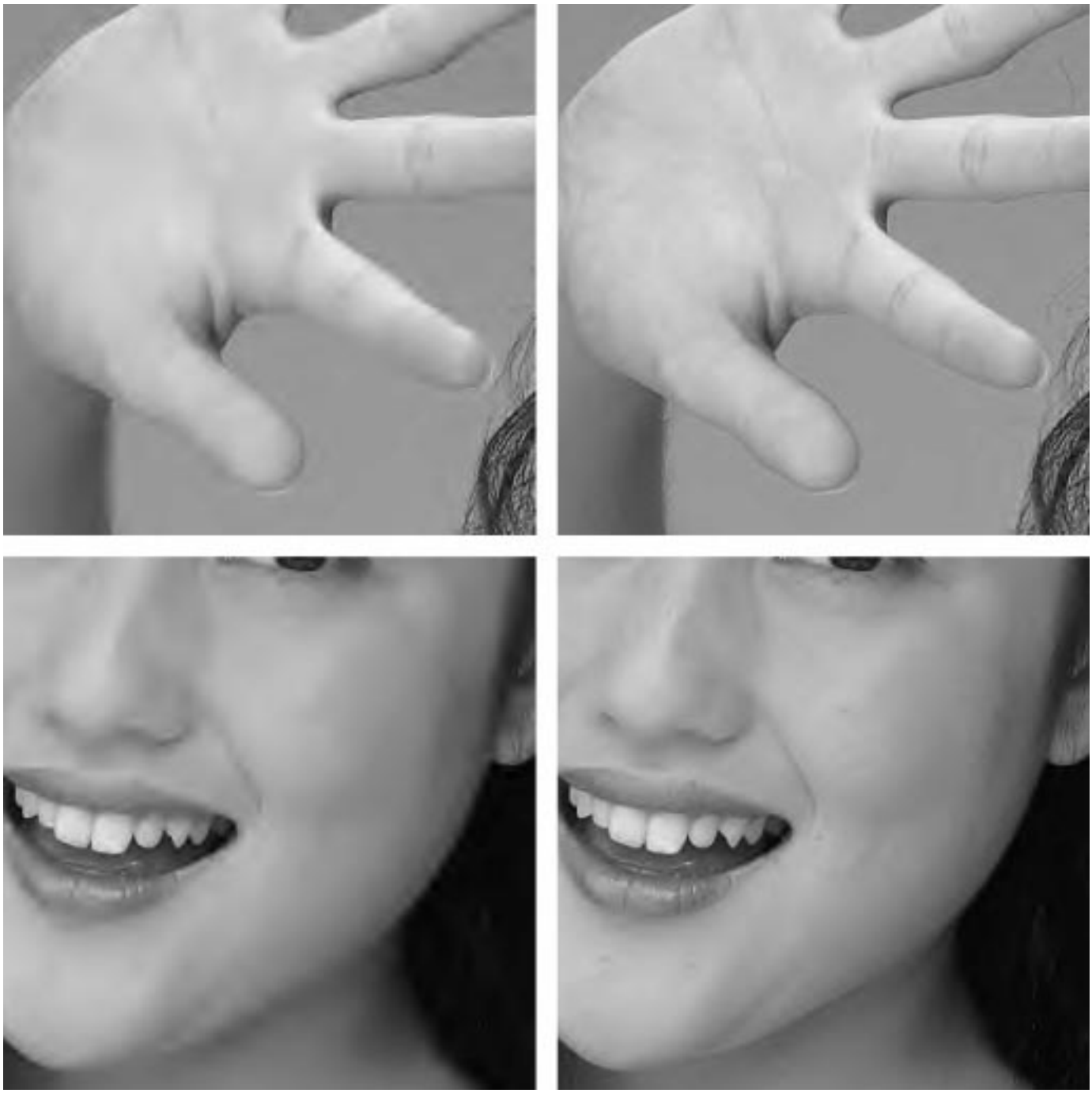


Fig. 8. Comparison between SPIHT (left) and EBCOT using the visual masking metric (right). Shows 500 × 500 regions from the 2560 × 2048 test image "woman," at 0.15 bpp.

e.g., from bottom to top or left to right. Since the code-blocks can be decoded in any order and the wavelet transform may be incrementally implemented in any direction, it is clear that EBCOT is well suited to such applications. At the opposite extreme we consider applications which require only a small, arbitrarily located region in the original image. In general, the number of subband samples represented by the code-blocks required to synthesize the requested image region will be larger than the region itself. In this section we attempt to quantify this inherent inefficiency. Remote browsing applications lie between these two extremes: a client interactively requests regions from a server, which may or may not ultimately constitute the entire image. The efficient realization of such applications depends upon careful caching of code-blocks which will often belong to multiple regions.

For the purposes of this investigation, we assume that a square image region with $R \times R$ pixels must be decompressed from an EBCOT bit-stream representing an original image with very much larger dimensions so that we can ignore boundary conditions. Fig. 9 contains a log-log plot of R vs. $\overline{R}$, where $\overline{R} \times \overline{R} \times \mu$ is the number of bits required to reconstruct the requested region and μ is the compressed bit-rate of the original image. Thus, the

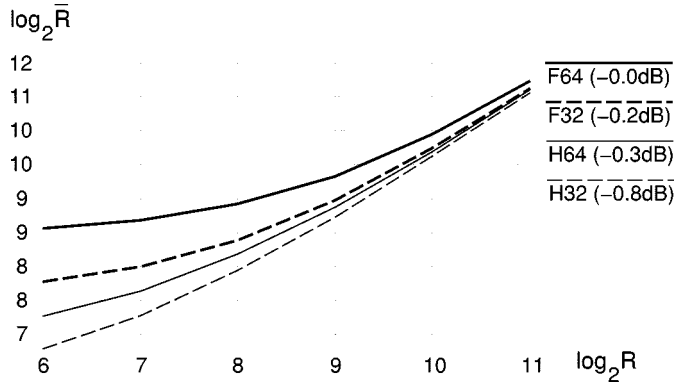


Fig. 9. Random access efficiency for various regions of size 64 × 64 through to 2048 × 2048.

cost of recovering the requested $R \times R$ region is equivalent to the cost of recovering an $\overline{R} \times \overline{R}$ image, separately compressed to the same bit-rate. In our case $\mu = 1$ bpp. The curves in Fig. 9 are averages, assuming a uniform distribution for the location of the requested region. The relevant subband samples and hence code-blocks are computed as in [9], assuming Daubechies 9/7 wavelet kernels with the usual Mallat decomposition. The transmission cost is determined by taking into account the average bit-rate in each of the subbands, as determined from the statistics of the three large images in Table III.

The four curves in Fig. 9 correspond to four different block size configurations: the curves labeled "F64" and "F32" correspond to code-blocks with a fixed size of 64 × 64 and 32 × 32 in every subband; The curves labeled "H64" and "H32" utilize the "frames" mode in the JPEG2000 VM3A software [20] to obtain code-blocks whose size depends upon the resolution level. In the "H64" case, code-blocks of size 64 × 64 are used in the highest resolution subbands, while blocks of size 32 × 32 are used in the next lower resolution level and so forth. We start with 32 × 32 code-blocks in the "H32" case. Each of the curves in Fig. 9 is also labeled with the average loss in compression performance relative to that reported in Table III. It would appear that the two most attractive configurations for applications requiring this type of random access are those corresponding to "F32" and "H64." Both exhibit only relatively small losses in compression efficiency for a significant gain in random access efficiency. While "H64" offers superior random access efficiency, we note that it degenerates into the less desirable "H32" case if the image is browsed at half the original resolution.

Evidently, random access granularity is relatively coarse: in the "H64" case, a region of size 256 × 256 requires the same number of bits as an image of size 400 × 400 compressed to the same bit-rate. Nevertheless, the capability is attractive for interactive browsing of very large images, where the requested regions might represent a significant portion of the client's display.

VIII. Relationship to JPEG2000

Since EBCOT was first adopted as the basis for the JPEG2000 image compression standard, some modifications have been introduced to the entropy coding part of the algorithm described in Section III. Since many readers are likely to have an interest in JPEG2000 and PART-I of the standard is now stable [1], we include here a brief summary of these changes. Most of the changes are described in [6].

A. Changes to Enhance Compression Efficiency

In this paper, the 18 probability models used by the conditional arithmetic coder are initialized to the usual equi-probable state. By contrast, in JPEG2000 some of the contexts are started in an assumed highly skewed state to reduce the model adaptation cost in typical images.

B. Changes to Reduce Complexity

A low complexity arithmetic coder, known as the MQ coder [21], has replaced the more classical arithmetic coder used in this paper. The MQ coder avoids multiplications and divisions in a similar manner to the more widely known QM coder. The JPEG2000 entropy coder does not transpose the HL subband's code-blocks, as described in Section III-A; instead, the corresponding entries in the ZC context assignment map are transposed. The JPEG2000 standard uses only three coding passes per bit-plane instead of four; the forward and reverse significance propagation passes have been merged into a single forward significance propagation pass, whose preferred neighborhood is identical to that of the reverse pass. This ensures that all coding passes have a consistent scan direction, at a small expense in compression efficiency.

Each sub-block is scanned column-by-column, rather than row-by-row, and sub-blocks have been reduced to size 4 × 4 from the optimal size of 16 × 16 considered in this paper. With these very small sub-blocks and highly skewed initialization of the probability models, we found that explicit coding of sub-block significance, as in Section III-B, is no longer justified. The coder behaves as though all sub-blocks are significant from the outset so that the corresponding bit-stream entries, $\mathcal{S}^p$, in Fig. 7 are all empty. With this modification, the coder is now more easily understood as operating on stripes of four rows each. Nevertheless, it appears that the most efficient software implementations, such as that in JPEG2000 VM5, are those which exploit properties of the MQ coder to realize the sub-block paradigm at an implementation level.

The cumulative effect of these modifications is an increase of about 40% in software execution speed for the entropy coding part of the system, with an average loss of about 0.15 dB, relative to the results reported in Section V. Since the software implementation of the entropy decoder in VM5.1 [1] has been heavily optimized (short of actually resorting to assembly code), the timing results reported in Table IV help to establish the complexity of the EBCOT coder. With small images (e.g., 512 × 512), the JPEG2000 VM5.1 entropy coder has comparable execution speed to the official public domain implementation of SPIHT [13] without arithmetic coding; this coder suffers about 0.5 dB loss relative to that reported in Table III for SPIHT with arithmetic coding. For the larger images of Table IV, SPIHT suffers from inherently nonlocal memory accesses, and runs 6 to 30 times more slowly than JPEG2000 VM5.1, depending the bit-rate.

C. Options in JPEG2000

JPEG2000 has an optional mode to enable parallel implementation of the coding passes within any code-block. Although the

TABLE IV
CPU DECODING TIMES OBTAINED USING JPEG2000 VM5.1 WITH A 400 MHz PENTIUM II PROCESSOR. CPU TIMES EXPRESSED IN SECONDS PER MILLION IMAGE PIXELS

Bit Rate	"Bike"	"Cafe"	"Woman"
0.0625	0.022	0.019	0.018
0.125	0.043	0.040	0.039
0.25	0.086	0.084	0.074
0.5	0.165	0.150	0.157
1.0	0.307	0.301	0.294

independent coding principle in EBCOT ensures that multiple code-blocks may be processed in parallel, microscopic parallelism at the level of individual coding passes can be exploited for more efficient hardware implementations. To enable this behavior, the arithmetic codeword generation process can be reset at the commencement of each coding pass and the various coding contexts reset to their skewed initial states. The context quantization process for the various coding primitives may also be constrained to be *vertically stripe causal*, meaning that samples from future stripes may be considered insignificant when determining the coding contexts. These options typically result in an additional loss of about 0.15 dB with 64×64 code-blocks and about 0.35 dB with 32×32 code-blocks at modest bit-rates. Most of the ideas behind these parallel processing options are explained in [11], [12].

A so-called *lazy coding* option has also been introduced, in which the arithmetic coding procedure is completely bypassed for most of the significance propagation and magnitude refinement coding passes. This mode substantially reduces complexity at high bit-rates, with the loss of less than 0.1 dB in compression performance.

IX. CONCLUSIONS

The EBCOT image compression algorithm offers state-of-the-art compression performance together with an unprecedented set of bit-stream features, including resolution scalability, SNR scalability and a "random access" capability. All features can coexist-exist within a single bit-stream without substantial sacrifices in compression efficiency.

The ability to produce independent, finely embedded bit-streams, for relatively small blocks of subband samples enables the effective use of post-compression rate-distortion optimization. In turn, this enables the successful exploitation of visual masking, which has been hampered by causality constraints in more conventional compression frameworks. The EBCOT algorithm also introduces the concept of abstract quality layers which are not directly related to the structural properties of the underlying entropy coder. This endows the bit-stream with tremendous flexibility, since the encoder may decide to construct any number of layers from any combination of code-block contributions whatsoever.

REFERENCES

[1] *JPEG2000 Verfication Model 5.0 (Technical Description)*, ISO/IEC JTC1/SC29/WG1 N1420, 1999.

[2] H. Everett, "Generalized Lagrange multiplier method for solving problems of optimum allocation of resources," *Oper. Res.*, vol. 11, pp. 399–417, 1963.

[3] I. Höntsch and L. Karam, "APIC: Adaptive perceptual image coding based on subband decomposition with locally adaptive perceptual weighting," in *Proc. IEEE Int. Conf. Image Processing*, vol. 1, 1997, pp. 37–40.

[4] J. Li and S. Lei, "Rate-distortion optimized embedding," in *Proc. Picture Coding Symp.*, Berlin, Germany, Sept. 10–12, 1997, pp. 201–206.

[5] S. Mallat, "A theory for multiresolution signal decomposition: The wavelet representation," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 11, pp. 674–693, July 1989.

[6] *Reduced Complexity Entropy Coding*, ISO/IEC JTC1/SC29/WG1 N1312, June 1999.

[7] A. Mazzarri and R. Leonardi, "Perceptual embedded image coding using wavelet transforms," in *Proc. IEEE Int. Conf. Image Processing*, vol. 1, 1995, pp. 586–589.

[8] *Quality Improvement Using Contrast Sensitivity Filtering*, ISO/IEC JTC1/SC29/WG1 N1306, June 1999.

[9] D. Nister and C. Christopoulos, "Lossless region of interest with a naturally progressive still image coding algorithm," in *Proc. IEEE Int. Conf. Image Processing*, Oct. 1998, pp. 856–860.

[10] E. Ordentlich, M. Weinberger, and G. Seroussi, "A low-complexity modeling approach for embeddded coding of wavelet coefficients," in *Proc. IEEE Data Compression Conf.*, Snowbird, UT, Mar. 1998, pp. 408–417.

[11] E. Ordentlich, D. Taubman, M. Weinberger, G. Seroussi, and M. Marcellin, "Memory efficient scalable line-based image coding," in *Proc. IEEE Data Compression Conf.*, Snowbird, UT, Mar. 1999, pp. 218–227.

[12] *Line-Based Coding—On Merging VMA and VMB*, ISO/IEC JTC1/SC29/WG1 N1201, Mar. 1999.

[13] A. Said and W. Pearlman, "A new, fast and efficient image codec based on set partitioning in hierarchical trees," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, pp. 243–250, June 1996.

[14] J. M. Shapiro, "An embedded hierarchical image coder using zerotrees of wavelet coefficients," in *IEEE Data Compression Conf.*, Snowbird, UT, 1993, pp. 214–223.

[15] F. Sheng, A. Bilgin, P. Sementilli, and M. Marcellin, "Lossy and lossless image compression using reversible integer wavelet transforms," in *Proc. IEEE Int. Conf. Image Processing*, Oct. 1998.

[16] D. Taubman and A. Zakhor, "Multi-rate 3-D subband coding of video," *IEEE Trans. Image Processing*, vol. 3, pp. 572–588, Sept. 1994.

[17] D. Taubman, "Directionality and scalability in image and video compression," Ph.D. dissertation, Univ. Calif., Berkeley, 1994.

[18] *Embedded, Independent Block-Based Coding of Subband Data*, ISO/IEC JTC1/SC29/WG1 N871R, July 1998.

[19] *EBCOT: Embedded Block Coding with Optimized Truncation*, ISO/IEC JTC1/SC29/WG1 N1020R, Oct. 1998.

[20] *JPEG2000 Verification Model VM3A*, ISO/IEC JTC1/SC29/WG1 N1143, Feb. 1999.

[21] *Proposal of the Arithmetic Coder for JPEG2000*, ISO/IEC JTC1/SC29/WG1 N762, Mar. 1998.

[22] A. B. Watson, "DCT quantization matrices visually optimized for individual images," *Proc. SPIE*, vol. 1913, pp. 202–216, 1993.

David Taubman (S'93–M'95) received the B.S. degree in computer science and mathematics in 1986 and the B.E. degree in electrical engineering in 1988, both from the University of Sydney, Australia. He received the M.S. degree in 1992 and the Ph.D. degree in 1994, both in electrical engineering, from the University of California, Berkeley.

From 1994 to 1998, he was with Hewlett-Packard Research Laboratories, Palo Alto, CA, joining the University of New South Wales, Sydney, in 1998 as a Senior Lecturer in the School of Electrical Engineering. Since 1998, he has been heavily involved in the working group ISO/IEC JTC1/SC29/WG1, developing the JPEG2000 image compression standard; he is the author of the Verification Model (VM3A) software and accompanying documentation upon which the standard is based. His research interests include highly scalable image and video compression technology, inverse problems in imaging, perceptual modeling and optimization, and network distribution of compressed multimedia content.

Dr. Taubman received the University Medal from the University of Sydney in 1988 for electrical engineering. In the same year, he also received the Institute of Engineers, Australia, Prize and the Texas Instruments Prize for Digital Signal Processing, Australasia. His 1996 paper with A. Zakhor, entitled "A common framework for rate and distortion based scaling of highly scalable compressed video" was awarded Best Paper by the IEEE Circuits and Systems Society in 1997.