A New Embedded Image Codec Based on the Wavelet Transform and Binary Position Coding

Hongsheng Cai and Gagan Mirchandani
Department of Computer Science and Electrical Engineering
The University of Vermont
Burlington, Vermont 05405, USA
October 31, 1996

Abstract

1 Using multiresolution representations, impressive coding gains are currently obtained with methods that separate position and amplitude information for independent coding. Thresholding is employed to obtain binary position patterns across subbands which are then efficiently coded. Amplitude information is embedded within the position information. A new, simple and efficient approach for an embedded image codec, that exploits intraband rather than interband redundancy is presented. The new codec applies autoadaptive block coding to the binary position information of the wavelet transform coefficients (BPW). A nonstationary source model is used to describe the binary position information. It is shown that the coding gain obtained stems from the exploitation of the nonstationarity. And that intraband redundancy is more significant than interband redundancy. Experimental results with benchmark gray-level images, show its performance close to one of the state-of-the-art algorithms, even without further entropy coding. The codec can be made either progressive-resolution or progressive-fidelity with no corresponding implementation loss of coding efficiency.

EDICS number: IP 1.1
Corresponding author: Hongsheng Cai, Quantum Corporation, SHR1-3/E29, 333 South Street, Shrewsbury, MA 01545.
Tel.: (508) 770-2197 Fax: (508) 770-2604 Email: cai@uvm-gen.emba.uvm.edu

1Permission to publish this abstract separately is granted.
1. Introduction

Embedded image codecs are used in progressive image transmission having applications in areas where a successive refinement of information is desired or necessary. Current techniques in progressive transmission, achieve progressive-fidelity through an embedded coding of position and amplitude information. For example, in Shapiro’s [1] embedded zerotree wavelet (EZW) algorithm, position information, referred to as the significance map, is coded using the concept of zerotrees. Amplitude information is coded using a successive-approximation quantization and is attached within the position information. A further coding gain is obtained through a lossless entropy-based coding scheme. The performance of EZW is among the best in reported coding results[1]. In earlier work, similar in philosophy to that of the zerotree method, partition priority coding proposed by Huang et al. [2] relies on arithmetic coding to encode the position information. Recently, work by Said and Pearlman [3] reports a further gain over EZW. This has been obtained with an improved zerotree structure that reduces redundancy yet further, by removing already coded tree nodes from further consideration and updating the significance tree for each threshold. Both zerotree based techniques are seen to exploit both intraband and interband redundancy by examining correlation between the wavelet transform coefficients at the same spatial locations. Intraband, in that spatial regions of insignificance within subbands are examined and interband, in that these regions are connected by a tree structure across subbands. This correlation, occurring after the decorrelating effect of the wavelet transform, has been categorized by Shapiro [1] as mostly nonlinear in nature.

In [1], an image model based on a decreasing spectrum assumption is proposed to justify use of the tree structure for position encoding. In [3], spatial self-similarity between subbands is the basis for the so-called set-partitioning algorithm. The concept of zerotree coding is similar in philosophy to that of the so-called region quadtree [4] or what is also known as autoadaptive block coding (ABC) [5], [6] for binary image coding. ABC is based on the recursive decomposition of a two dimensional binary image into equal-sized quadrants until homogeneous blocks are reached. The zerotree approach, on the other hand, operates on a binary position tree (obtained by scanning the wavelet transform coefficient tree). Each position tree is recursively divided into four sub-
trees until empty sub-trees, i.e., zerotrees are reached. They both use a similar “divide and conquer” approach. In that sense, the zerotree method is the quadtree concept applied to a wavelet position tree. The success of both these coding schemes depends on the sparsity and/or the degree of spatial nonstationarity of the image being coded.

In this work we propose an alternative to the position encoding schemes of [1] and [3]. To allow progressive-resolution transmission and also less complex coding, we examine subbands independently, i.e., only intraband redundancy is exploited. The simpler autoadaptive block coding method is utilized for the binary images generated. A simple nonstationary model for characterizing the binary position patterns is employed to analyze the performance of ABC. It is seen that the code length using ABC, applied to a set of benchmark gray-level images is closer to the source entropy than that obtained with other block coding schemes. It is also seen that the rate-distortion gains are very close to those obtained with the EZW algorithm, even without additional entropy coding. This suggests that the coding gain with the zerotree method is generated mainly from the reduction of intraband redundancy. By decoupling subbands, a progressive-resolution transmission is achieved as well as reduced codec complexity.

Images available in multiple resolutions are suitable in meeting different user requirements and processing capabilities. In image database browsing, for example, a user may initially only need a low resolution, low precision image to recognize major features. Higher resolution details would be requested, as necessary. While the multiresolution feature is difficult to achieve for block transform based coding such as JPEG, it comes naturally with wavelet-based coders, where subbands with increasing resolution can be transmitted progressively. With zerotree based coding, however, subband coefficients are organized in a tree based data structure. While the precision of the coefficients can be progressively improved, the resolution remains the same throughout the coding process. Since coefficients from different subbands are not separable, progressive-resolution transmission is not possible without losing the coding efficiency of the original full-resolution codes. While it is true that due to the decreasing spectrum assumption [7], low resolution coefficients are more likely to be transmitted first, especially for the first few large thresholds, the decoder still decodes the bit stream as a full resolution (predetermined) code with most of the high resolution band coefficients likely being zeros. As the thresholds get lower, higher resolution
coefficients emerge. It is possible to have multiresolution reconstructions with these codecs using only coefficients from specific low frequency subbands. But transmitting and decoding unnecessary subbands is wasteful of channel bandwidth, computing power and storage space resources. On the other hand, providing multiple codes with different resolutions using the same zerotree approach (with different tree depths) would put the burden on the encoder (i.e., more storage space and encoding time). Recently, in an extension of [3], a progressive-resolution option where the subbands are coded separately has been proposed [8]. But complexity is increased through employment of more complex arithmetic coding schemes to achieve compression.

In this work in embedded coding, which has its roots in [9] and [10], position information for each of the subbands is coded independently. By carefully indexing the code for each subband, the coder fully exploits the multiresolution feature of the wavelet transform, permitting either progressive progressive-fidelity or progressive-resolution coding. The latter is done by simply transmitting code up to a certain threshold for a specific number of subbands. Eliminating unnecessary subband transmission translates into shorter transmission and decoding time. The coder also fits well in today’s variable rate ATM network environment and facilitates prioritized transmission of subbands in which more important subbands are transmitted through error resilient high priority channels and less important high frequency subbands transmitted through low priority channels. Furthermore, the coding efficiency is not affected with these added capabilities. The encoder only needs to maintain that same code for each of the subbands. By varying the number of subbands and the thresholds, the encoder can generate a code stream that provides either progressively increasing resolution or progressively increasing fidelity. This scheme also has a potential application in scalable video [11] where video signals are coded with multiresolutions.

The paper is organized as follows: We start by describing the proposed image coder in Section 2. Autoadaptive block coding is applied to code the subband position patterns resulting from scanning the wavelet transform coefficients. Efficient implementation of ABC is obtained through an array representation of the quadtrees. Performance of ABC for coding the position patterns is analyzed in Section 3 and compared with that of other block coders. ABC is shown to be suitable for our application. Sections 4 and 5 provide experimental results and a conclusion, respectively.
2. Description of the New Codec

Given an image of size $N \times N$, after an $R$-level separable 2-D discrete wavelet transform, we get a transform coefficient set $W$:

$$W = \{ LL, HL_l, LH_l, HH_l : l = 1, \ldots, R \}$$  

(1)

where $LL$ represents the lowest frequency band, $l$ is the level of a subband and $HL$, $LH$, $HH$ are horizontal, vertical and diagonal orientations respectively. For an embedded coder, subband coefficients are scanned with a set of logarithmically decreasing thresholds $T_i = T_{i-1}/2, i = 1, 2, \ldots$, where the largest threshold ($T_0$) is determined by the magnitude of the largest coefficient among all the subbands, with the $LL$ subband mean subtracted. The scanning scheme is similar to that used in [1] and [3]. For subband $X$ and threshold $T_i$, a binary subband image (significance map) $B_i^X$, representing the positions of coefficients in the subband falling within the corresponding magnitude range, is obtained.

$$B_i^X(m, n) = \begin{cases} 
1 & \text{if } T_i \leq |X(m, n)| < T_{i-1} \\
0 & \text{otherwise}
\end{cases}$$  

(2)

where $m, n$ are the coefficient indices. Subbands of the binary position images thus obtained are coded separately with the autoadaptive block coding method described in the following section. The amplitude bits are then attached to the position code stream.

2.1. Autoadaptive Block Coding of the Significance Map

Block encoding techniques have found use in bi-level facsimile as well as in gray-scale image coding [12]. Kunt [13] first proposed a simple block coding method suitable for compression of sparse binary images. It was later generalized to autoadaptive block coding (ABC) [5]. Block coding has also been used to encode bit-planes directly associated with gray-scale images, giving a lossless encoding scheme; but its performance suffers when coding the less significant bit-planes where the bit patterns are more random and hence more difficult to compress. In our application
with wavelet transform coding, subbands are sparse in large-magnitude coefficients and so are the associated binary position images. Hence they appear to be ideal candidates for block coding. We use here, two-dimensional autoadaptive block coding with two dimensional partitioning [14]. ABC is very similar to one of the so-called pointerless representations of the region quadtrees [4]. It is a preorder (depth-first) traversal of the quadtree with the alphabet consisting only of symbols "0" and "1". Compression is achieved due to the existence of large sized "0" blocks in the position images. Figure 1 illustrates the coding scheme. A binary image is first divided into large blocks called initial blocks, that are coded sequentially. If all the pixels in the block being coded are zero, the block is coded with a single bit "0"; otherwise a prefix "1" is assigned to the block which is then divided into four sub-blocks. Each sub-block is treated in the same way as was the initial block until a predefined smallest block is reached, which is coded by simply listing the bit pattern of the block. In practice, a hierarchy of blocks of sizes $2^k \times 2^k, 2^{k-1} \times 2^{k-1}, \ldots 2 \times 2$ are used, where the initial block is $2^k \times 2^k$, $k$ integer and the final block is $2 \times 2$.

2.2. Efficient Implementation of ABC for the Image Codec

For simplicity of discussion, images are assumed to be square. Generalization to non-square images is straightforward. To implement the algorithm we need first to find the position maps of subbands. One way is to implement equation (2), for each of the thresholds. Alternately, the position maps for different thresholds can be generated simultaneously, facilitating parallel construction of quadtrees for all the position maps. This is done by dividing the absolute values of the wavelet transform coefficients with a predetermined smallest threshold and converting the results to 16-bit fixed-point words. Sign bits are then prefixed to the fixed-point words. This is known as a sign-magnitude code. The fixed-point words are then separated to get position and magnitude information. In the implementation, the total number of thresholds is limited to fourteen. This is sufficient since in practice, the number of thresholds seldom exceeds a dozen or so before the position patterns become so dense that block coding is not efficient any more. After the conversion, each coefficient is expressed as an integer multiple of the smallest threshold, which is the largest threshold ($T_0$) divided by $2^{14}$. The most significant bit (MSB) of the word is the sign bit and the second MSB is used for another purpose, as indicated later. The remaining
fourteen bits are the fixed-point representations of the absolute value of the coefficient. It is clear that the most significant “1” in the fourteen bits provides the position information needed for block coding while the following bits give the magnitude information. These fourteen bits, representing position and magnitude information, need to be separated so as to implement block coding (Figure 2). The two words after separation are referred to as the position word and the amplitude (sign plus magnitude information) word, respectively. Each “bit-plane” of the position words in a subband will be coded by the ABC scheme. The bit-plane represents the bottom level of the quadtree for a particular threshold. Hence there are a total of fourteen quadtrees for each subband. After the separation, each group of four adjacent position words in a subband are “OR”ed to generate their parent’s node value for all the fourteen quadtrees (Figure 3). It should be noted that the bitwise “OR” operations are carried out for all the fourteen thresholds at the same time. This process continues with the next level (parent nodes) of the quadtrees and so on. Finally, a single word for the roots of all the fourteen quadtrees is obtained for the subband. To avoid the use of pointers in the quadtree coding, a two-dimensional array is used to represent the quadtrees (similar to the implicit 1-D array representation of a binary tree) for each level (three orientations) of the subbands. Figure 4 shows the array representation of the quadtrees for one level of subbands. Note that this is not the original hierarchical wavelet transform coefficient array. The shaded area of the array stores the position words for one level (three orientations) of the transform coefficients. The upper left quadrant of the array stores all the internal nodes of the quadtrees for all the fourteen bit positions. The children of an internal node with an array index \((i, j)\) of a quadtree are nodes with array indices \((2i, 2j)\), \((2i+1, j)\), \((2i, 2j+1)\), \((2i+1, 2j+1)\). This is done for each level of the subbands. The total overhead for the full quadtree representations for all the subbands is about one third of the total number of the coefficients. After the construction of the full quadtrees, quadtree coding (ABC) for a particular threshold is simply a depth-first traversal of the corresponding quadtree. This is computationally fast since the building of the quadtrees is done in parallel for all the thresholds.

A potential problem with this technique is the memory overhead of storing position and amplitude words separately, since two arrays appear to be needed. The problem can be resolved by merging the two words after the parent node of the position word is generated (i.e., as before
they were separated). And for each composite word, the second MSB is used as a flag. Initially
this bit is set to “1”. When a quadtree is traversed later during the position coding, all the bits
are masked by their corresponding flags. Once a “1” is read out of a bottom-level word (the first
“1” must be the position bit) the flag is set to “0”. That means that the bits following the “1”
are magnitude bits. They will be masked by the “0” flag during later passes of position coding
so that they will not be confused as position bits.

2.3. Automatic Initial Block Size Selection

When a binary image is coded using ABC, an initial block size has to be determined first. The
more sparse the binary pattern, the larger the initial block size. By grouping larger blocks of
zeros and representing them with one bit (“0”), high compression can be achieved. On the other
hand, if the density of 1’s is high, large blocks of zeros are less prevalent; using a larger initial
block size would result in greater overhead. Since the performance of the method depends on the
probability of 1’s, initial block sizes should be determined based on that statistic. Accordingly,
the parameter (block size) setting of the resulting coder is image-dependent. An advantage of
our implementation is that the optimal initial block size for a position map can be determined
easily with little computational overhead.

In order to determine the optimal initial block size, we determine analytically the code length
for the autoadaptive block code. Assume the size of the binary position image as $N \times N$, where
$N = 2^M$, $M$ an integer. The depth of the full quadtree is $M$. We label the root node as depth
0, the bottom level nodes as depth $M$. Let $P_i$ be the probability of empty blocks at level $i$ (with
size $2^{(M-i)} \times 2^{(M-i)}$), $L(i)$ the estimate of the code length using ABC when the starting block size
is $2^{(M-L)} \times 2^{(M-L)}$, $l = 0, \ldots, (M-1)$. At level $i$, the number of blocks is $2^{2i}$ and the probability
of nonempty blocks is $1 - P_i$. The average number of nonempty blocks is thus $2^{2i}(1 - P_i)$. Each
nonempty block splits into four blocks at the next level ($i + 1$), while empty blocks result in
terminal nodes at the current level. The code length using ABC, starting at level $l$, is merely the
total number of nodes at all levels from $l$ to $M$, i.e.,

$$L(l) = 2^l + \sum_{i=1}^{M-1} 2^{(i-1)}(1 - p_i)$$  \hspace{1cm} (3)$$

Let $I$ be the optimal starting level (corresponding to a initial block size of $2^{(M-I)} \times 2^{(M-I)}$), then

$$L(I) < L(I + 1), \quad 0 \leq I < M - 1$$  \hspace{1cm} (4)$$

and

$$L(I) \leq L(I - 1), \quad 1 \leq I \leq M - 1$$  \hspace{1cm} (5)$$

where we assume that if code length at two adjacent levels are equal, the one with a smaller initial block size is used. Substituting equation (3) for $L(I)$ in (4) and (5), we get

$$p_{I-1} \leq \frac{1}{4} \quad \text{and} \quad p_I > \frac{1}{4}$$  \hspace{1cm} (6)$$

We now show that $I$ is easily determined in our array implementation of the quadtree. We start from the topmost level of the quadtree. A counter is used to count the number of zero nodes at that level (see Figure 5). As soon as the count exceeds $1/4$ of the number of blocks at that level, the counting stops since the optimal level has been reached. Otherwise go to the next level and start a new counter for zero nodes therein. Continue the process until condition (6) is met. Because of the sparsity of the position patterns for most of the initial thresholds, it is quite likely that large initial block sizes will be chosen. The number of the associated upper-level nodes is small so the computational cost is minimal. Figure 5 shows a typical sample position pattern and its upper-level nodes in the array representation. (Only one HH subband quadtree is shown.) After the optimal initial block size is determined, the choice of the block size is coded and sent to the decoder as side information and encoding starts at that level.
3. Performance Analysis of ABC for Position Map Coding

It has been shown by Huang et al. [2] that the entropy of an unordered data source with independent symbols equals the entropy of magnitude ordered data source plus the entropy of the position information required to reconstruct the original data source. For a source with memory, this is usually not true since memory among the original symbols may be lost in the position patterns. For a memoryless source, separating position and amplitude information does not increase the total entropy. Huang et al. proposed a partition priority coding approach that coded such sources without overhead. The advantage of this approach is that source symbols are coded according to their degree of importance (magnitude) and progressive-fidelity transmission is realized. That same approach is used in EZW coding where the position information is coded using zerotrees with overhead. Said and Pearlman's improved zerotree method has less overhead since coded symbols are removed from later consideration, which is similar to Huang's approach. In our coder the position maps are coded separately for each subband. Since coded symbols are set to zeros, instead of being taken out of consideration, the separation of position and amplitude introduces overhead. But this overhead is small, since zeros cost little to code using ABC.

As shown in [1], the coding cost of the position patterns accounts for a significant portion of the overall cost. To measure the performance of our coder, we need to determine the information content (entropy) of the position map. To determine the entropy it is necessary to have a source model that closely approximates the underlying source. A good model might also suggest suitable coding algorithms for the source. Kunt [14] studied models for binary facsimile images. Several different types of models were investigated and it was shown that the block source model gave the lowest or close to the lowest entropy among all the proposed models for six test images (see table II in [14]). This justifies the use of the simple block coding algorithm for binary facsimile images proposed also by Kunt [6]. Most image models assume stationary processes because of their ease of analysis. While Kunt's simple 2-D block model, based on a first-order Markov process, fits the experimental results well for facsimile images, it gives large errors for the binary position maps obtained from scanning the wavelet coefficients. Accordingly, we investigate a more appropriate model for the position maps.
3.1. A Simple Nonstationary Model

An example of a subband binary position image is shown in Figure 6. The image is divided into small blocks. The distribution of 1’s is not uniform and hence the source is spatially nonstationary. A model based on a stationary Markov assumption is clearly not suitable here. Accordingly, we turn to a nonstationary model. We make the following assumptions about the model: The source is memoryless; it consists of two uniform distributions with different probabilities of 1’s; regions in a binary image may belong to either of the two distributions, which are called type I and type II regions, respectively. These assumptions are reasonable since, first, a natural image can be roughly divided into smooth areas and edges. Smooth areas generate little output in high-frequency subbands (corresponding to type I regions in the position maps), while edges are highlighted by the wavelet transform in high-frequency subbands (corresponding to type II regions in the position maps). Secondly, due to the decorrelating effect of the wavelet transform, the subband coefficients are roughly uncorrelated and hence ideally, justify the memoryless assumption. The 1’s in the position map represent coefficients within the same magnitude range and based on the memoryless assumption we assume them to be independent. A still more general model with multiple uniform distributions may be used for a more accurate description of the source. From our simulations, a two-distribution model is seen to be adequate. Let $p_1$ be the probability of 1’s in type I region, $p_2$ the probability of 1’s in type II region and $p_1 < p_2$. A parameter $\alpha$ is used to denote the relative area of type I region, where $0 < \alpha < 1$. Assume an image size as $N \times N$. The average probability of 1’s is $p$

$$\alpha p_1 + (1 - \alpha)p_2 = p$$

The source entropy for the single distribution memoryless source is

$$H = N^2 h(p)$$
where

\[ h(p) = -p \log_2(p) - (1 - p) \log_2(1 - p) \]

The source entropy for the two-distribution model is

\[ H_2 = N^2(\alpha H(p_1) + (1 - \alpha) H(p_2)) \]

For the general multiple distribution model, the entropy is given by

\[ H_m = N^2 \sum_{i=1}^{m} (\alpha_i H(p_i)) \]

where \( \alpha_i \) is the relative area of the region with probability of 1's being \( p_i \) and

\[ \sum_{i=1}^{m} \alpha_i = 1 \quad \sum_{i=1}^{m} \alpha_i p_i = p \]

The source entropy is less than that with a single distribution model if the \( p_i \)'s are not equal. An efficient coder that exploits the nonstationarity of the source can achieve a compression close to the new entropy bound. Autoadaptive block coding is suitable for coding this type of source. ABC, in its simplest form (no further entropy coding), splits the image into “quiet” and “busy” quadrants. For quite (all zero) quadrants, it labels each of them with a “0” and stops further splitting. For busy (nonempty) quadrants, the splitting process continues until empty blocks or the bottom of the quadtree (pixel level) is reached. Note that it does not code significant pixels in groups, i.e., it does not exploit memory among those pixels. The more active a block, the more bits are needed to code it. Essentially, ABC adapts itself to local statistics of the image. The efficiency of the ABC method, like that of the zerotree method, depends on the existence of large number of empty blocks. This requires that the binary image be sparse and/or highly nonstationary.
3.2. Parameter Estimation for Given Images

To find out how accurately the model fits the actual binary images and to use the model to analyze the performance of ABC, we need to estimate the model parameters ($\alpha_i$ and $p_i$) for test images. From now on, we assume a two-distribution model where there are three parameters to be estimated: $\alpha_i$, $p_1$, and $p_2$. A binary position image is first divided into blocks of size $4 \times 4$ and each block is considered to belong to either one of the two types. Since each pixel in a block is independent and uniformly distributed, the probability of blocks containing $k$ 1’s for type I or type II blocks obeys the binomial distribution. To simplify the problem, we use the Poisson approximation. This is reasonable since the position patterns are sparse and the number of 1’s is small compared with the block size. The parameters for the two Poisson distributions are $\lambda_1$ and $\lambda_2$, and

$$\lambda_1 = n^2 p_1, \quad \lambda_2 = n^2 p_2$$

(13)

where $n \times n$ is the size of the blocks used for estimation (here $n = 4$). The Poisson distribution of blocks containing $k$ 1’s for type I region is

$$f_1(k; \lambda_1) = e^{-\lambda_1} \frac{\lambda_1^k}{k!}$$

(14)

and for type II region

$$f_2(k; \lambda_2) = e^{-\lambda_2} \frac{\lambda_2^k}{k!}$$

(15)

So the overall probability of blocks containing $k$ 1’s is

$$f(\alpha, \lambda_1, \lambda_2; k) = \alpha e^{-\lambda_1} \frac{\lambda_1^k}{k!} + (1 - \alpha) e^{-\lambda_2} \frac{\lambda_2^k}{k!}$$

(16)

A maximum likelihood estimation is carried out for sample images obtained from scanning the subbands. The choice of block size in the estimation process is based on the following consideration: the size should be large enough so that adjacent blocks are approximately independent.
(required by ML estimation) and the Poisson approximation should hold (the blocks are still sparse). On the other hand, it should also be small enough so that there are sufficient number of blocks to make the ML estimation meaningful and easy to compute. The likelihood function is defined as

$$ l(\alpha, \lambda_1, \lambda_2) = \prod_{i=1}^{M} f(\alpha, \lambda_1, \lambda_2, k_i) $$

(17)

where $M$ is the number of blocks used in the estimation and $k_i$ the number of 1’s in the $i$-th block. The maximum likelihood estimates for $\alpha, \lambda_1, \lambda_2$ are the ones that maximize $l(\alpha, \lambda_1, \lambda_2)$. Table 1 gives the results of the estimation for five position patterns from scanning the HL2 subband of image “Lena” with thresholds $T_6$ through $T_{10}$. The resulting binary position images are labeled as $BP_6$ through $BP_{10}$. Figure 7 shows the measured distribution of the blocks and the one calculated from equation (16) with estimated parameters for the position pattern $BP_7$ in table 1. We observe that a good estimate is obtained. The source entropy for the images with the estimated model parameters (given by equation (10)) are also listed in table 1 along with the zeroth-order entropy for single-distribution model (given by equation (8)). We see that entropies based on two distribution model is significantly lower than the zeroth-order entropies based on single distribution.

### 3.3. Code Length Estimation of the ABC Based on the Model

The estimated parameters are used to predict the code length using ABC. As seen from the last section, the code length using ABC depends on the choice of the initial block size. Large initial block sizes for patterns with high probability of 1’s would mean high coding overhead and small initial block sizes with sparse patterns would unnecessarily divide the large empty regions, thus lose coding gain. Here we assume the optimal initial block sizes are always chosen (with the scheme proposed in section 2.3) and the final block size is always $2 \times 2$ (down to pixel level). Then the average estimated code length is given by equation (3) where $P_i$ is given by

$$ P_i = \alpha(1 - p_1)(\frac{N}{2^2})^2 + (1 - \alpha)(1 - p_2)(\frac{N}{2^2})^2 $$

(18)
Note that in estimating the code length with ABC, we implicitly assume that type I and type II regions are not mixed; they cluster to their own kind instead. Such a model would generate even lower source entropy than that given by equation (10).

The estimated and actual code lengths (labeled as $L^{(est)}$ and $L$, respectively) are listed in Table 1 and the error is less than 3.5% for five test images with average probability of 1’s ranging from 0.0163 to 0.2299. Similar results are obtained for other subband position patterns. The model then appears satisfactory for predicting the code length.

3.4. Comparison of ABC with Other Block-based Coders

We have observed that ABC is suitable for coding nonstationary sources. ABC can be compared with other block-based coders, e.g., Kunt’s simple nonadaptive code [14], Zeng and Ahmed’s block code (ZA) [15] and Kavalarchik’s generalized block code (GBC) [16].

**Kunt’s Nonadaptive Block Coding:**

In Kunt’s nonadaptive block coding method [13] [6], the image is first divided into initial blocks of size $m \times n$. The codeword for the most likely configuration (all-white block) is simply “0”. Any other nonempty block is represented by an $(mn + 1)$ bit codeword, the first bit being a prefix “1” and the remaining $mn$ bits being the listing of all the pixels in the block. Hence, the code is variable-length with only two possible lengths, namely 1 and $mn + 1$.

**Zeng-Ahmed (ZA) Block Coding:**

The ZA code proposed by Zeng and Ahmed [15] is a modification of Kunt’s code. In [15] one-dimensional blocks of length $n = 2^b$ are considered. Between any two adjacent blocks a “0” is introduced. If there is no black pixel in a block, then no coding is needed for the block. If there are black pixels in a block, then each one is assigned a prefix “1” followed by $b$ bits to indicate its location in the block. This location is with respect to the left end of the $2^b$-sized block and numbered from 0 through $2^b - 1$. In the ZA coding, each black pixel is coded independently, introducing a fixed amount of overhead. When the pattern is sparse, the code length is much closer to the zeroth-order entropy of the source than Kunt’s block code (see next section).

**Generalized Block Coding**

Kavalarchik [16] gives a generalized approach for block coding (GBC). Kunt’s code and the ZA
code can be considered as its special cases. Better coding results are obtained with a modified ZA code based on the GBC approach and the coding scheme works even better under unknown source models. But the actual implementation of GBC could be complex. So we omit the GBC in our performance comparison. More details of GBC can be found in [16].

3.4.1. Stationary Memoryless Source

Performance analysis and comparisons of the above-mentioned block-based coders under a stationary memoryless source assumption are given in [15] and [16]. In those references, 1-D blocks were used. Since the source is stationary and memoryless, there is no difference in performance between 1-D and 2-D blocks for those coders as long as the blocks contain the same number of pixels. We use 2-D blocks for ABC and assume that optimal block sizes are always chosen. We also assume that the image has \( K = 2^{a+b} \) pixels and is divided into \( 2^a \) blocks, each block containing \( 2^b \) pixels. For ABC, \( b \) is assumed to be an even number so that square blocks with power-of-two side lengths are possible. In the experiment \( K \) is chosen to be \( 128 \times 128 \).

The code length for Kunt’s block code is given in [16]:

\[
I^{(\text{Kunt})} = \min_{a,b} 2^a (1 + 2^b (1 - \text{Probability(all zero block)})) \\
= \min_b \left( \frac{K}{2^b} + K (1 - (1 - q)^{2^b}) \right), \quad b = 1, 2, \ldots, 14
\]

(19)

where \( q \) is the probability of 1’s. As indicated in [6], it is difficult to find the optimal block size. Here the minimum is obtained by trial and error.

The code length for the ZA code is given in [16]:

\[
I^{(\text{ZA})} = \min_a (2^a + (\log_2 K - a + 1)Kq), \quad a = 1, 2, \ldots, 14
\]

(20)

where the optimal block size \( a \) for the ZA code is given in [15] as

\[
a = \left \lfloor \log_2 \left( \frac{Kq}{\ln 2} \right) \right \rfloor
\]

(21)

where \( \left \lfloor \right \rfloor \) represents the rounding operation.
The code length for ABC is given by equation (3). For a stationary source, the lower bound for the average length of any code is the zeroth-order source entropy \( H \). The code lengths versus the probability of 1’s are plotted in Figure 8. At low probabilities, we see that the ZA coder performs better than Kunt’s coder. Also, ABC outperforms Kunt’s block code at low probabilities and as the probability reaches about 0.083, ABC turns into simple nonadaptive block code. That is because the optimal initial block size changes from \( 4 \times 4 \) to \( 2 \times 2 \) at this point. This happens when the probability of empty \( 4 \times 4 \) blocks is less than or equal to \( 1/4 \) (from (6)). \( i.e., \frac{(1 - p)^{4 \times 4}}{4} \) \( (22) \)

and hence,

\[
p \geq 1 - 2^{-1/8} \approx 0.083
\]

(23)

The block size of Kunt’s coder is restricted to powers of two, missing some other block sizes. A similar restriction was applied to Kunt’s coder in [15] and [16] in order to compare it with the ZA coder and GBC. Kunt’s coder would have better performance had the restriction not been imposed. But the effect of this restriction is small since for most of the probabilities, Kunt’s coder has small optimal block sizes and only a few other choices would be missed. For instance, Kunt’s optimal block size is less than 8 for \( p > 0.038 \). In this range, other eligible choices of Kunt’s block sizes are limited to 5, 6, 7 \(^2\) only. Also, each side of the 2-D blocks used in ABC is restricted to powers of two. The probability given by (23) could be higher were the restriction removed. But other choices would make implementation of the algorithm more complex.

The results show that for stationary memoryless sources, the performance of ABC is better than that of Kunt’s block coder for sparse binary patterns (probability range: \( p < 0.083 \)). They are both inferior to the ZA coder when the probability of 1’s is less than 0.191. This is because of the high average coding overhead for Kunt’s code and ABC when the pattern is very sparse while the overhead for ZA code is relatively small in this range [15].

\(^2\)Block sizes less than 4 are not appropriate due to a high prefix overhead.
3.4.2. Nonstationary Memoryless Source

Unlike the stationary memoryless case where there is only one parameter ($p$) for the model, there are now three parameters ($p_1, p_2$ and $a$) for the two-distribution model. These parameters change with the zeroth-order probability $p$. Usually the disparity between $p_1$ and $p_2$ is large at small $p$ and small at large $p$, but the relationship is image dependent and difficult to characterize analytically. It is therefore difficult to test all the cases. Accordingly, coder performance is obtained empirically on sample test images. Table 1 lists the code lengths for ZA and Kunt's block code. The optimal block size of Kunt's code is obtained by trial and error and the block size for ZA code is obtained using equation (21).

It should be mentioned that the performance of the ZA code is not affected by the stationarity of the source. It only depends on the number of 1's of the whole image. This makes ZA code insensitive to changes in source statistics as long as the total number of ones remains the same. On the other hand, this also means that unlike ABC and Kunt's code, ZA code can not exploit the nonstationarity of the source.

From Table 1 we observe that ABC achieves lower than zeroth-order entropy compression for three out of five test images and outperforms both the ZA and Kunt's code. Note that the ZA code can not achieve compression lower than the zeroth-order entropy. Also ABC is fully adaptive with no training and needs little knowledge of the source statistics. Because of the hierarchical structure of the algorithm, it is possible to determine the optimal initial block sizes during the process of coding, while that is difficult to accomplish with Kunt's block code and the ZA code.

The source entropy using the nonstationary model provides the lower bound for the compression achievable with that model. However, for actual binary images, blocks of the same type tend to cluster. This implies that they are not strictly independent, and hence the actual entropy should be lower than that obtained with the proposed model. But even for this simplified model, there is still room for improvement, since code lengths with ABC for the test images are still significantly higher than the entropies based on the two-distribution model.

Similar analysis could be applied to the zerotree type algorithms since their underlying philosophy is the same: recursive decomposition of a nonhomogeneous object into homogeneous
ones. In essence, block coding plays a role similar to simple entropy coding. It exploits the nonstationarity of the source and can achieve compression lower than the zeroth-order entropy. The combination of thresholding with ABC is thus similar to that of scaler quantization with an entropy coder. This is the key to the success of the proposed coder, though ABC is in a sense only a rudimentary form of an entropy coder. The hierarchical structure makes it dynamic and adaptive but also introduces significant overhead in the code. One way to reduce the overhead is to combine it with further entropy coding. Other more sophisticated binary coding schemes could also be employed to code the position information and may result in better performance. But the simplicity and adaptivity of ABC make it attractive in this application.

Huang et al. used a similar multiple distribution assumption to design their arithmetic coder. But the image model used a fixed division of images and the associated arithmetic coder needed training. While in our case there is no need to determine the model parameters, ABC automatically adapts to each individual image and adjust its behavior to match the local statistics.

4. Experimental Results

The Daubechies biorthogonal wavelet with the corresponding filter orders of 9 and 7 is used. It is known to perform well for image coding applications [17]. The test images “Lena” and “Goldhill” are of size 512 by 512 at 8 bits/pixel. A five-level wavelet transform is applied. Figure 9(a) is the original “Lena” image and Figure 9(b)-9(f) are reconstructed “Lena” images at different bit rates using BPW. The reconstructed image is recognizable at a very low bit rate. Figure 10 shows the comparisons of BPW with Shapiro’s EZW and also with the codec proposed by Said and Pearlman. We see that the new codec achieves similar PSNR (within 0.3 dB) to that of EZW even without further entropy coding. In [3] it is reported that entropy coding can increase PSNR by another 0.3 to 0.6 dB. Since redundancy still exists in our bit stream, a similar gain in PSNR could be expected were entropy coding to be employed. Hence it can be concluded that the performance of the proposed coder is at about the same level as that of EZW. This is achieved without exploiting any interband redundancy. This suggests that the interband redundancy is not significant when compared to intraband redundancy and the coding gain of the EZW comes
mainly from exploiting the latter.

5. Conclusion

A new application of autoadaptive block coding is proposed. The BPW coder, used for coding position information, accomplishes intraband redundancy reduction among coefficients with similar magnitude. The performance of ABC is analyzed with a simple nonstationary model for the binary position images. It is shown that ABC fits the nonstationary nature of the images. The new image coder, which treats subbands independently, achieves a performance close to that of the EZW coder. This points to a new direction in wavelet-transform based coding. By decoupling the subbands in the coding, the coder can be made either progressive-resolution or progressive-fidelity with no corresponding implementation loss of coding efficiency.

Currently, zerotree based coders provide the best rate-distortion performance for still image compression. They utilize the concept that when subband position maps are coded together in a tree structure, the joint entropy is smaller than the sum of entropies of the subbands, if the position maps are correlated; hence higher compression gain can be expected. The gain obtained is derived from grouping zeros and not from grouping ones. However, as we have observed, the cost of coding zeros is small, in both EZW and BPW, as compared with that of coding significant coefficients. And EZW has no advantage over BPW in coding significant coefficients. Accordingly, both algorithms, EZW and BPW have a similar performance.

In the coder proposed here, interband effects are not exploited and hence taking advantage of interband correlation is precluded. But this also removes restrictions generated by the “parent-
children” structure. It is now possible to exploit intraband redundancy to greater extent. This is effected by utilizing larger block sizes for block coding. Resulting coding gains compensate for the losses incurred by not using an interband grouping. So the overall gain obtained from the independent subband coding is close to that obtained with the EZW method. The proposed coder is extremely simple, the simplicity stemming from abandoning the tree structure.

While the PSNR of the new coder is less than 1 dB lower than that of Said and Pearlman’s
algorithm, our coder is truly scalable. Users can request variable resolutions of the image best
suited to their needs and processing capabilities, i.e., it can be either progressive in resolution or progressive in fidelity. The added feature neither increases the complexity nor decreases the coding efficiency of the original code.

The coding gains of the wavelet transform based coders are obtained in two ways. The first is due to the nonstationarity, in the strict sense, of the coefficients in the subbands. The nonstationarity implies that coefficients in the subbands do not have the same pdf. For the associated binary position maps, this results in a lower total entropy than that for the stationary case, assuming that the average probability of ones is the same. The nonstationarity across subbands leads to a similar reduction in entropy. The zerotree method attempts to exploit the nonstationarity across both intra and inter subbands, while BPW only addresses the intra-subband nonstationarity. However, since the performance of BPW is very close to that of the EZW coder, it can be concluded that the intraband redundancy (due to nonstationarity) is more significant than the interband redundancy. Both the zerotree and the ABC methods assume no correlation among significant coefficients. They only reduce the redundancy among insignificant coefficients by grouping zeros into zerotrees or empty blocks. This is also why the memoryless model gives a good code length estimation for ABC.

The second way to obtain a coding gain is to exploit the correlation among significant coefficients (both interband and intraband). Empirical statistical studies show that the linear correlation among coefficients is small [1] [18]. Because of the scanning scheme, the correlation carried over to the binary position patterns is even smaller (this also justifies the use of memoryless model in our work). It is more costly to reduce this type of redundancy. Efforts in both [1] and [3] on reducing this type of redundancy achieve limited results. They include arithmetic coding and other entropy codings that exploit the high-order entropy of the source. For our intraband coding scheme, the second type of redundancy can be reduced by increasing the efficiency of binary position encoding. One possible choice is the generalized block code (GBC) proposed by Kavalerehik[16]. Other coding approaches that can be applied include context based entropy coder (JBIG, for example) and combination of the block coding with high-order entropy coding. But like the entropy coding with the zerotree methods, the additional coding gain may not be great enough to justify the use of such more complex coders.
List of Tables

1 Block coding results for sample position images from “Lena” ........................................... 28

List of Figures

1 An autoadaptive block coding example ................................................................. 23
2 Separation of position and amplitude words ......................................................... 23
3 “OR” operation on the position words ........................................................................ 24
4 Array representation of a quadtree ........................................................................... 24
5 A sample position pattern and its upper-level nodes in the array representation of the corresponding quadtree ................................................................. 25
6 A sample position pattern ....................................................................................... 25
7 Comparison of the estimated and actual pmf ................................................................ 26
8 Comparison of ABC with other block codes for stationary memoryless sources ................ 26
9 Compression results for “Lena” image ....................................................................... 27
9 Compression results for “Lena” image (con’t) ............................................................ 29
10 Comparison between BPW and zerotree based methods ........................................... 30

References


Figure 1: An autoadaptive block coding example

Figure 2: Separation of position and amplitude words
Figure 3: “OR” operation on the position words

Figure 4: Array representation of a quadtree
Figure 5: A sample position pattern and its upper-level nodes in the array representation of the corresponding quadtree

Figure 6: A sample position pattern
Figure 7: Comparison of the estimated and actual pmf

Figure 8: Comparison of ABC with other block codes for stationary memoryless sources
Figure 9: Compression results for “Lena” image
Table 1: Block coding results for sample position images from “Lena”

<table>
<thead>
<tr>
<th></th>
<th>BP₆</th>
<th>BP₇</th>
<th>BP₈</th>
<th>BP₉</th>
<th>BP₁₀</th>
</tr>
</thead>
<tbody>
<tr>
<td>p</td>
<td>0.0163</td>
<td>0.0404</td>
<td>0.0714</td>
<td>0.1346</td>
<td>0.2299</td>
</tr>
<tr>
<td>H</td>
<td>1967.8</td>
<td>4000.1</td>
<td>6077.5</td>
<td>9336.8</td>
<td>12743</td>
</tr>
<tr>
<td>α</td>
<td>0.881</td>
<td>0.766</td>
<td>0.657</td>
<td>0.395</td>
<td>0.0611</td>
</tr>
<tr>
<td>λ₁</td>
<td>0.025</td>
<td>0.043</td>
<td>0.125</td>
<td>0.51</td>
<td>0.919</td>
</tr>
<tr>
<td>λ₂</td>
<td>2.0</td>
<td>2.61</td>
<td>3.086</td>
<td>3.23</td>
<td>3.857</td>
</tr>
<tr>
<td>p₁</td>
<td>0.0016</td>
<td>0.0027</td>
<td>0.0078</td>
<td>0.0319</td>
<td>0.0574</td>
</tr>
<tr>
<td>p₂</td>
<td>0.125</td>
<td>0.1631</td>
<td>0.1929</td>
<td>0.2019</td>
<td>0.2411</td>
</tr>
<tr>
<td>H₂</td>
<td>1302.5</td>
<td>2796.9</td>
<td>4685.2</td>
<td>8511.3</td>
<td>12575</td>
</tr>
<tr>
<td>L(Kurt)</td>
<td>3072</td>
<td>5104</td>
<td>7516</td>
<td>10488</td>
<td>14240</td>
</tr>
<tr>
<td>L(ZA)</td>
<td>2114</td>
<td>4334</td>
<td>6724</td>
<td>10711</td>
<td>15394</td>
</tr>
<tr>
<td>L(est)</td>
<td>1792.5</td>
<td>3708.2</td>
<td>6116.7</td>
<td>10766</td>
<td>14586</td>
</tr>
<tr>
<td>L</td>
<td>1840</td>
<td>3592</td>
<td>5988</td>
<td>10436</td>
<td>14240</td>
</tr>
<tr>
<td>error</td>
<td>-2.58%</td>
<td>3.23%</td>
<td>2.15%</td>
<td>3.16%</td>
<td>2.43%</td>
</tr>
</tbody>
</table>


Figure 9: Compression results for “Lena” image (con’t)


Figure 10: Comparison between BPW and zerotree based methods


