Progressive Image Coding using Augmented Zerotrees of Wavelet Coefficients

Nasir Rajpoot, Roland Wilson
Department of Computer Science,
University of Warwick,
Coventry

(September 18, 1998)

Abstract

Most state-of-the-art image compression techniques exploit the dependencies between the subbands in a wavelet transformed image. We propose a progressive image coding technique which is based upon the augmented zerotrees of wavelet coefficients. An augmented zerotree can be regarded as a spatial orientation tree with some extended characteristics. Both the coder and decoder require to maintain only one list. The augmented zerotrees are constructed in an efficient recursive manner and the algorithms are provided. A comparison reveals that the coder performance is close to, and sometimes outperforms, other well known coding methods. Suggestions are made for further improvements regarding the speed and compression performance of the coder.

Contents

List of Figures

1 Introduction

Wavelets are a new family of basis functions that satisfy certain mathematical requirements and are used in representing data and other functions. Even though the idea is not new, wavelets have generated tremendous interest because of an increased level of activity in image and video compression in recent years. The fact that the basis functions are compactly supported over a finite range in both time and frequency domains gives the wavelet transform an edge over the conventional Fourier transform, whose basis offers compact support only in the frequency domain. Moreover, one can choose a suitable basis function from infinitely many basis functions, depending upon nature of the problem [1].

Subband decomposition is a way of dividing the data into different frequency components using the wavelet transform. The presence of many zeros in the high frequency bands and the dependencies, between high frequency coefficients at the coarser scales and the corresponding high frequency coefficients at finer scales, make it possible to efficiently code the image for compression purposes.

In recent years, there has been a lot of work on the compression of still and video images using the wavelet transform [2]. Many state-of-the-art image coding techniques [3, 4, 5] utilise subband coding for the compression purposes. The landmark achievement of very low bit-rate image compression was made by Shapiro in his work [3] on the embedded zerotree coding, namely EZW, of the wavelet coefficients with reasonable image quality. EZW generates an embedded bitstream thus allowing the progressive transmission and precise control of target bit rate or target distortion. In their work [4], Said and Pearlman presented an extended version of EZW, by defining set partitioning rules for dividing the hierarchical trees during coding or decoding. We note, however, that these rules are similar to those employed by EZW, with scanning order being the only difference.

In this report, an image coding algorithm, based on efficient coding of the augmented zerotrees of the wavelet coefficients, is proposed for still image compression. The main objective of this scheme is to minimise the number of codewords needed to efficiently code the positions of wavelet coefficients that are significant (or in other words, the *significance map*). We present efficient algorithms for detection (or construction) of the augmented zerotrees by the coder (or the decoder). The high frequency subbands in an octave subband decomposition represent the high frequency contents of the image in a particular direction. It makes sense, therefore, that the order in which these coefficients are scanned and encoded takes into account these orientations.

The organisation of subsequent sections is as follows. An introduction to the wavelet transforms, filter banks, and subband decomposition is presented in Section 2. Embedded image coding is briefly introduced in Section 3 and some state-of-the-art image coding algorithms are described briefly. Section 4 presents an augmented zerotree image coding (AZIC) algorithm, based on the efficient coding of augmented zerotrees. Some preliminary results are tabulated and plotted against other well known image coding techniques. Issues related to further improvement of this algorithm's performance are also discussed. The final section concludes the report with a summary.

2 Wavelet Transform and Subband Decomposition

Wavelets are a new family of basis functions that are used in representing the data and other functions. The tremendous amount of interest that wavelets has generated is mainly due to two reasons. First, wavelets are the basis functions that are compactly supported in both the time and frequency domains. This gives the wavelet transform an edge over the conventional Fourier transform, which offers compact support only in the frequency domain and thus is not good at approximating the discontinuities. Secondly, the mutliresolution nature of wavelet transform allows one to analyse the signal at different resolutions of time and frequency.

A brief introduction of the wavelet transform follows. Let $f(t)$ be a function defined on $\Re$. The function $f(t)$ is said to be in the *Hilbert space* $L^2(\Re)$, if $|f(t)|^2$, the space of square-integrable functions, is Lebesgue integrable [6]. It is desired to find out a complete orthonormal basis which spans $L^2(\Re)$.

Let $\psi(t)$ be a function which is nonzero for a finite amount of time only, and satisfies $\int \psi(t)dt{=}0$. In order to provide compact support in both time and frequency, all the basis functions are constrained to be in a set of basis functions $\{\psi_{jk}\}$, each with a finite support of different width [7]. Here, all the basis functions in $\{\psi_{jk}\}$ are the scaled and translated versions of the same prototype function ψ, known as the *mother wavelet* or the *analysing wavelet*. Considering a scale factor of 2, these functions ψ_{jk}, called *wavelets*, are defined as,

$$\psi_{jk}(t) = 2^{j/2}\psi(2^j t - k). \tag{1}$$

So the signal $f(t)$, when represented in the wavelet domain, will be written as,

$$f(t) = \sum_j \sum_k c_{jk}\psi_{jk}(t) \tag{2}$$

where

$$c_{jk} = \Big\langle f(t), \psi_{jk}(t) \Big\rangle \tag{3}$$

and $\langle ., . \rangle$ denotes the inner product.

The wavelet bases nicely decompose the function $f(t)$ into its pieces belonging to different subspaces. These subspaces are of two types: the scaling space V_j and the wavelet space W_j. The subspace V_j is spanned by the basis $\phi(2^j t - k)$ and the subspace W_j is spanned by the basis $w(2^j t - k)$, where $\phi(t)$ is the scaling function and $w(t)$ is the *mother* wavelet function

[8]. This decomposition is such that,

$$V_j \oplus W_j = V_{j+1}. \tag{4}$$

It is obvious here that W_j corresponds to the *detail* at level j. Together the space V_0 and the spaces W_j (for all $j \in Z$) satisfy the completeness condition for $L^2(\Re)$ and thus can represent any arbitrary function $f(t)$ from $L^2(\Re)$.

2.1 Filter Banks

In order to achieve the nice wavelet decomposition for multiresolution analysis of the discrete-time signals , the concept of filter banks is utilised. A filter bank is composed of multiple filters which are used either to decompose the signal or to reconstruct the signal. The filter bank which is used to decompose the signal is called the analysis bank and the one which is used to reconstruct the signal is called the synthesis bank. In an *m-channel filter bank*, there are m filters in each of the two banks.

There are two filters in a two-channel filter bank, as shown in Figure 1. The lowpass filter $\mathbf{H}$ and the highpass filter $\mathbf{G}$ correspond to the scaling function $\phi(t)$ and the wavelet function $w(t)$, respectively, in the continuous time. These filters separate the signal into signal contents belonging to particular frequency ranges. To avoid the doubling of storage required by the outputs of $\mathbf{H}$ and $\mathbf{G}$, both the outputs $\mathbf{U_0}$ and $\mathbf{U_1}$ are downsampled by a factor of 2, the operation denoted by $\downarrow 2$. This constitutes the *analysis bank*. The reconstruction of signal $\mathbf{X}$, ie. $\widehat{\mathbf{X}}$, is achieved by upsampling of the filtered outputs $\mathbf{Y_0}$ and $\mathbf{Y_1}$ followed by the addition of filtered outputs $\mathbf{X_0}$ and $\mathbf{X_1}$. The two filters $\widetilde{\mathbf{H}}$ and $\widetilde{\mathbf{G}}$ form the *synthesis bank*.

2.2 Perfect Reconstruction

It is required that approximation of the signal, $\widehat{x}(n)$, should be exactly the same as the original input signal $x(n)$. That is to say that the reconstruction

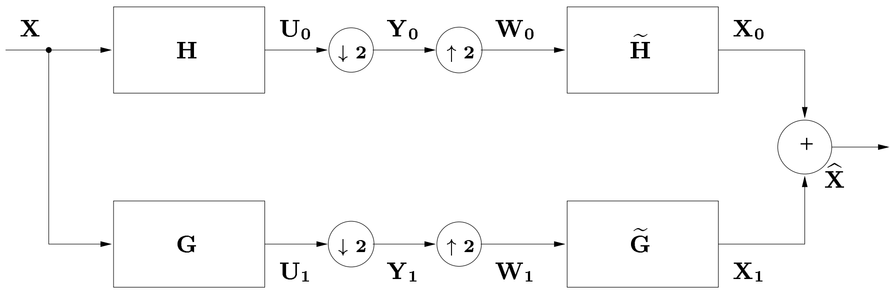

Figure 1: Block diagram of a two-channel filter bank

should be *perfect* and there should be no loss of information while using wavelets as an intermediate representation tool. This is possible if $X(w)$ is band-limited to either the upper half-band or the lower half-band [8].

But there is a hurdle to be crossed for achieving the perfect reconstruction of the original signal $x(n)$. The downsampling operation is a linear but time-variant operation and is responsible for the possible aliasing introduced in the frequency spectrum of the downsampled signal, due to the presence of $X(w+\pi)$ term. This term is undesired and should be countered by the synthesis filter bank. In 1988, Daubechies published her work [9] on orthonormal filter banks of different filter lengths, generated by the spectral factorization of a half-band filter, which satisfy the conditions for alias cancellation and no distortion. The Daubechies' wavelets are optimal with respect to the approximation accuracy and the orthogonality [8]. In an orthogonal filter bank,

$$\widetilde{\mathbf{H}} = \mathbf{H}^{\mathbf{T}} \tag{5}$$

and

$$\widetilde{\mathbf{G}} = \mathbf{G}^{\mathbf{T}}, \tag{6}$$

which means that in an orthogonal filter bank, the synthesis bank is a transpose of the analysis bank. The coefficients of Daubechies' 4-tap lowpass filter

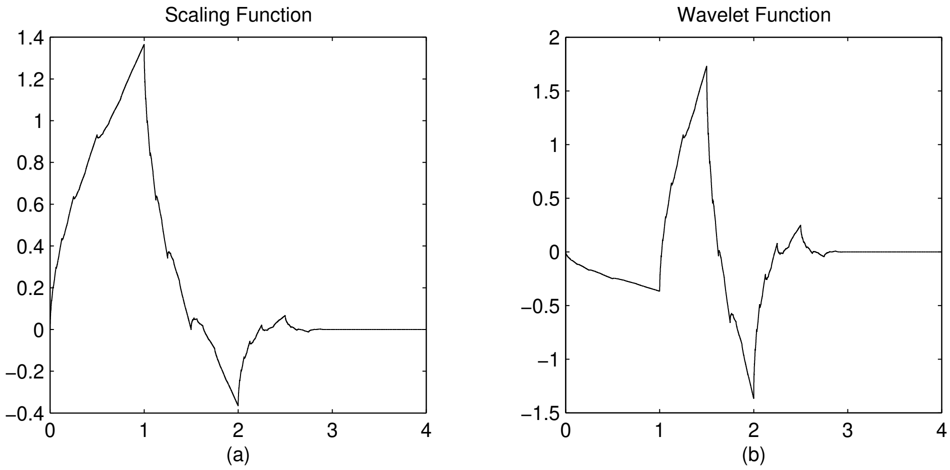

Figure 2: Daubechies-4 orthonormal functions
(a) scaling function and (b) wavelet function

are given by,

$$\{h(m)\} = \frac{1}{\sqrt{32}}\{\sqrt{3}+1, \sqrt{3}+3, 3-\sqrt{3}, 1-\sqrt{3}\}. \tag{7}$$

The scaling function and the wavelet function associated with these filters, denoted by DAUB4 for further reference, are shown in Figure 2.

2.3 Other Wavelets

For compression purposes, it is required that the transform being used highly decorrelates the image data. Adelson et al. [10] developed odd-tap quadrature mirror filter (QMF) kernels which are quite compact and provide very good frequency localization as well. Although these kernels do not offer an orthogonal basis set (unless they have at least $(N+4)/2$ taps for an image of size N), they can approach orthogonality even with a rather small number of taps. The coefficients of a 9-tap lowpass QMF kernel are,

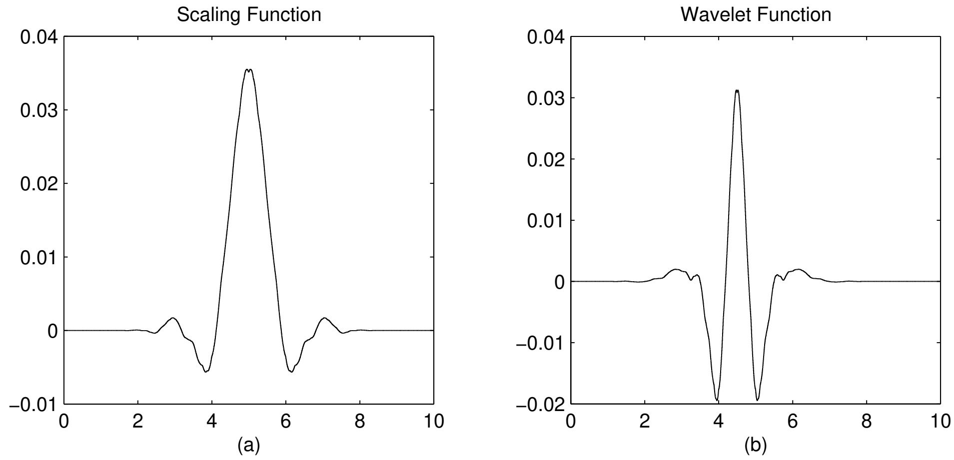

Figure 3: Adelson et al. near-orthogonal functions
(a) scaling function and (b) wavelet function

$$(2^{-\frac{1}{2}})\{h(m)\} = \{0.01995, -0.04271, -0.05224, 0.29271, 0.56458,$$
$$0.29271, -0.05224, -0.04271, 0.01995\}. \tag{8}$$

The scaling function and the wavelet function associated with these filters, denoted by ADELSON9, are shown in the Figure 3.

The orthogonality requirement can be relaxed to preserve the linear phase corresponding to symmetry for the wavelet. Cohen et al. [11] used biorthogonal bases to construct linear phase FIR filters with the exact reconstruction property, regardless of any regularity considerations. One such filter set is derived from a spline variant with less dissimilar lengths. The coefficients of 9-7 biorthogonal filters are given by,

$$(2^{-\frac{1}{2}})\{h(m)\} = \{0.026749, -0.016864, -0.078223, 0.266864, 0.602949,$$
$$0.266864, -0.078223, -0.016864, 0.026749\} \tag{9}$$

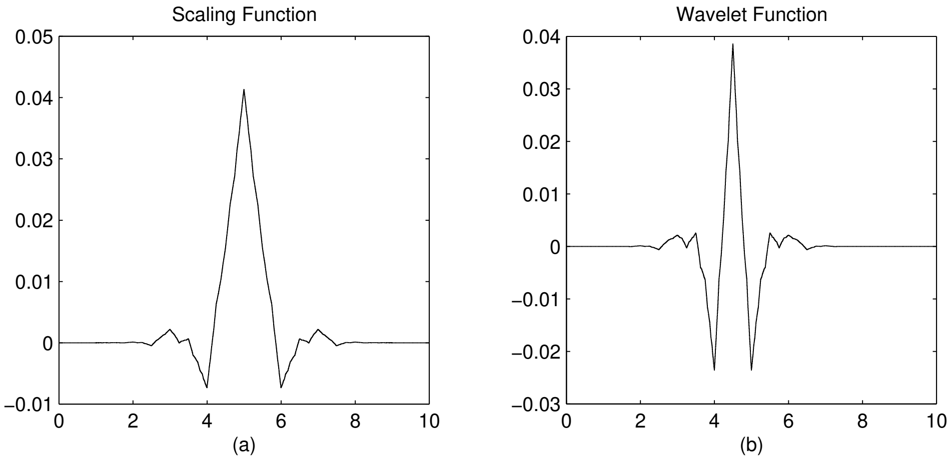

Figure 4: Daubechies et al. 9/7-pair biorthogonal functions
The analysis bank (a) scaling function and (b) wavelet function

and

$$(2^{-\frac{1}{2}})\{\widetilde{h}(m)\} \quad = \quad \{0.0, -0.045636, -0.028772, 0.295636, 0.557543, \\ 0.295636, -0.028772, -0.045636, 0.0\}, \qquad (10)$$

where the highpass filters $\{g(m)\}$ and $\{\widetilde{g}(m)\}$ are derived from the following equations.

$$\widetilde{g}(m) = (-1)^m h(1-m)$$

$$and$$

$$g(m) = (-1)^m \widetilde{h}(1-m) \qquad (11)$$

The scaling function and the wavelet function associated with the analysis filter bank, denoted by BIORTH97, are shown in the Figure 4.

2.4 Decomposition of an Image into Subbands

The multiresolution analysis of a signal can be performed by decomposing the signal with the help of filter banks, such as the ones mentioned above.

Each of these filter sets decomposes the signal into its components belonging to different frequency ranges, with the difference that each of them does so with its own frequency localization properties discussed briefly in the previous section.

In order to decompose an image, which is a two-dimensional discrete-time signal, a simple and fast method is required for the computation of wavelet transform coefficients of the image. Realising that wavelet transform is separable in the two dimensions, Wilson [12] defines the 1D discrete wavelet transform (DWT) in terms of a quadrature mirror filter (QMF) pair $\{h(m)\}$ and $\{g(m)\}$ given as follows,

$$c^i(n) = \sum_k (h(k)c^{i-1}(2n-k)) \tag{12}$$

and

$$d^i(n) = \sum_k (g(k)c^{i-1}(2n-k)), \tag{13}$$

where $c^0(n) = f(n)$ represents the original signal and corresponds to the subspace V_0 in the continuous time, and $\{d^i(n)\}, i = 1, 2, \ldots$ are the wavelet coefficients. Daubechies and Adelson et al. selected $\{g(m)\}$ to be derived from $\{h(m)\}$ given by,

$$g(m) = (-1)^m h(1-m). \tag{14}$$

The separability of 2D DWT intuitively urges one to compute the 2D DWT of an image as follows. The 1D DWT is performed on the rows (treating each of them as a 1D signal), followed by a 1D DWT on the columns as shown in Figure 5. The order does not matter here. This 2D DWT splits an image of size $m \times n$ into four subimages, each of size $\frac{m}{2} \times \frac{n}{2}$. These subimages are called *subbands*, since they represent the frequency information as shown in Figure 6. The subband LL represents the low frequency components of the original image at a coarse scale. The remaining

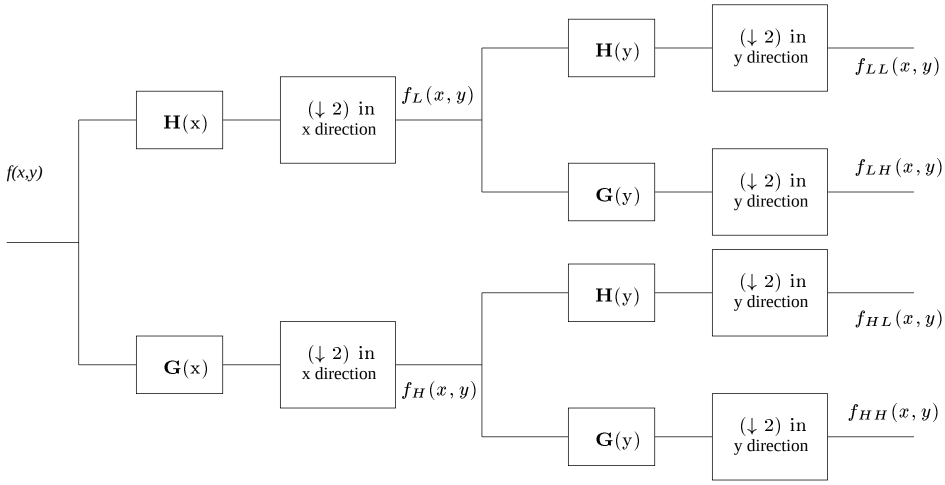

Figure 5: 2D DWT of $f(x, y)$ using the analysis filter bank

three subbands HL, LH, and HH represent the high frequency components, more precisely the *edges*, of the image at a coarse scale. A closer look at these three subbands reveals that the HL, LH, and HH represent the high frequency components of the image in horizontal, vertical, and diagonal directions respectively.

Perfect recovery of the original image from its subband decomposed version is guaranteed if the filter bank used for decomposition is orthogonal, such as the DAUB4 filters mentioned in Section 2.1. In this case, the original image is perfectly recovered by taking the inverse DWT of its decomposed version. Such a recovery by a 1-level inverse DWT is shown in Figure 7.

It should be noted here that the image could still be decomposed into more subbands, depending on its dimensions. A further 1-level decomposition of the image into more subbands can be achieved by operating another 1-level DWT on the LL subband of the image, giving in turn a 2-level DWT consisting of seven subbands, and so on.

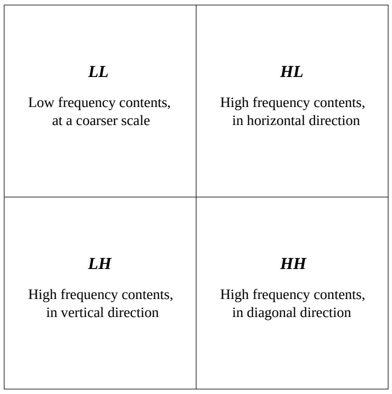

Figure 6: Subband decomposition of an image

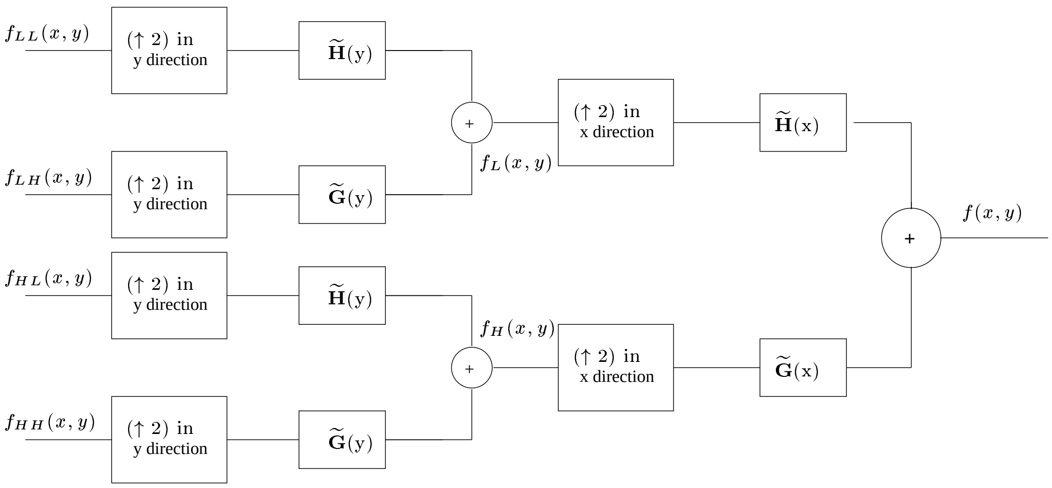

Figure 7: 2D inverse DWT to recover $f(x, y)$ using the synthesis filter bank

3 Image Compression using Wavelets

There are two main factors which contribute a lot when compressing an image represented in the wavelet domain. First, the presence of many near-zero values (or *insignificant* information) encourages one to apply an entropy coding technique to eliminate this redundancy after quantization. Secondly, a closer look at the wavelet transform coefficients reveals that there are certain type of dependencies between the wavelet coefficients in a coarser scale high frequency band and the wavelet coefficients, corresponding to the same spatial location and orientation, in a finer scale high frequency band. Earlier wavelet compression techniques, such as [13, 14], tended to exploit the former fact by using certain quantization and entropy coding techniques. Shapiro devised an efficient algorithm [3] using embedded zerotrees of wavelets, namely EZW, to compress the images by exploiting both of the above mentioned facts.

The next section explains a general schematic diagram of an embedded transform coder. Section 3.2 describes how EZW achieves very low bit-rate image compression. A detailed discussion of the SPIHT coding algorithm from an EZW perspective is given in Section 3.3. Some factors contributing towards the better performance of SPIHT over EZW are described. This is followed by a brief introduction of other popular image compression schemes.

3.1 Embedded Image Coding

The schematic diagram of an embedded image coding scheme, employing the image transform, is shown in Figure 8. The image coder constitutes of three main blocks: transform, quantizer, and entropy coder. The sole purpose of an image *transform* $\mathbf{T}$ operated on an image $\mathbf{I}$ is to decorrelate the image data. The output $\mathbf{C}$ of the transform would, then, be given by,

$$\mathbf{C} = \mathbf{T}\mathbf{I}. \tag{15}$$

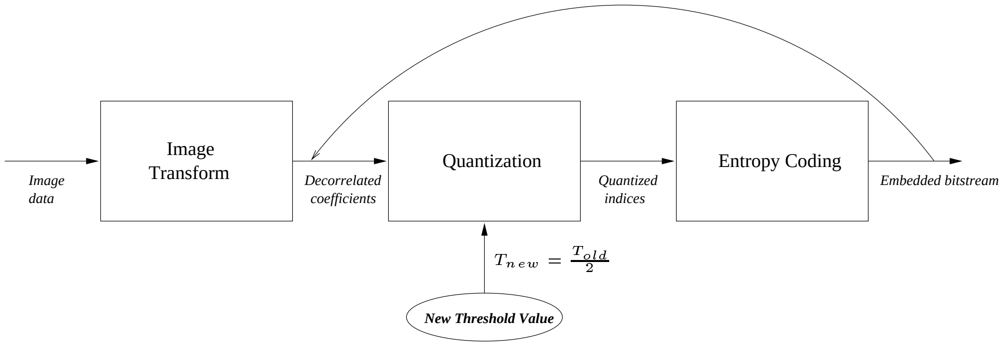

Figure 8: Schematic diagram of an embedded transform image coder

In the case of the DWT, $\mathbf{C}$ is the subband decomposed version of the original image and consists of the wavelet coefficients, at various scales, and the lowest frequency contents of the image at the coarsest scale. The image $\mathbf{I}$ can be perfectly recovered by operating the inverse transform $\mathbf{T^{-1}}$ on $\mathbf{C}$ as given by,

$$\mathbf{I} = \mathbf{T^{-1}C}. \tag{16}$$

The next block in Figure 8 is a *quantizer*, which quantizes the transformed image into a sequence of integers. This part of the image coder is mainly responsible for losses, since it is an irreversible operation. Output of the quantizer, which has quite lot of statistical redundancies, is coded using an *entropy coding* scheme (such as Huffman coding, run-length coding, arithmetic coding etc.). This is a completely reversible operation. In some coders, the output of the quantizer is reorganised as per understanding of the decoder. The quantization and entropy coding cycle continues until a target bit-rate is achieved.

3.2 EZW Compression

Following are the main features of Shapiro's embedded zerotree wavelets (EZW) image coding algorithm [3].

a). EZW exploits the self-similarities across different scales of an image wavelet transform. This is done with the help of zerotrees. In this quadtree kind of data structure, every node (except the leaves) has four children. A *zerotree root* node means that the information in wavelet transform coefficients corresponding to all the nodes in this tree is insignificant, with respect to a particular threshold. It means that we do not need to encode the whole tree and encoding only the root of this tree would do the job, thus providing a fair amount of compression. This is based upon Shapiro's hypothesis that, *if a wavelet coefficient at a coarser scale is insignificant, all wavelet coefficients at the same orientation and at the same spatial locations at the finer scales are more likely to be insignificant.*

In order to achieve very low bit-rate compression, the probability of the most likely symbol after quantization, ie. the zero-symbol, must be extremely high. Figure 9 shows these parent-offspring dependencies (or self-similarities, in a sense). The order in which the coefficients are scanned, and then coded, is shown in Figure 10. The scanning starts from the lowest-frequency subband LL_3 (for a 3-level DWT) and ends at HH_1. All positions in a subband are scanned before the scan moves to the next subband, and all the parents are scanned before their children. As obvious, this scanning order is employed by both the coder and the decoder.

b). The *significance map* (the positions of significant wavelet coefficients) is encoded during the *dominant pass*, using the following codewords in the scan order mentioned above:

i) POS (positive significant),

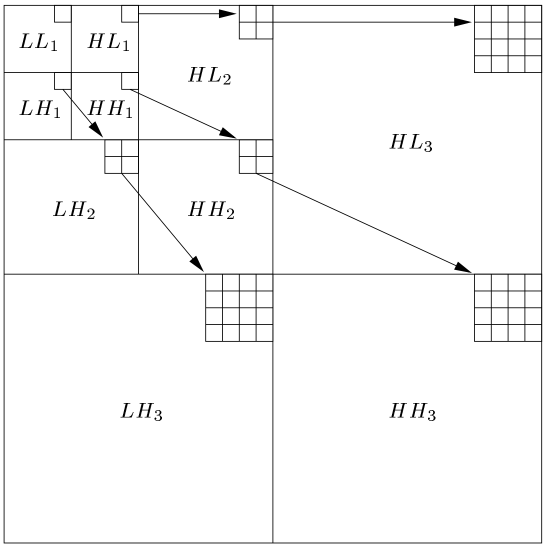

Figure 9: Parent-offspring dependencies in a 3-level DWT

ii) NEG (negative significant),

iii) IZ (isolated zero/insignificant), and

iv) ZTR (root of a zerotree).

After every dominant pass, the *subordinate pass* encodes single bit information for the decoder to decide whether a wavelet coefficient which has already been detected to be significant is in the upper half or in the lower half of the uncertainty interval.

c). The zerotree approach had previously been utilised by other researchers in their work [15, 16] on image and video compression. The main thrust of EZW was *embedded coding* of the image data which makes possible the progressive coding (reconstruction) of an image by the coder (decoder) and the precise control of target bit-rate.

The embedded coding was achieved by the *successive approximation quantization* (SAQ) in EZW. The value of absolute deterministic thresh-

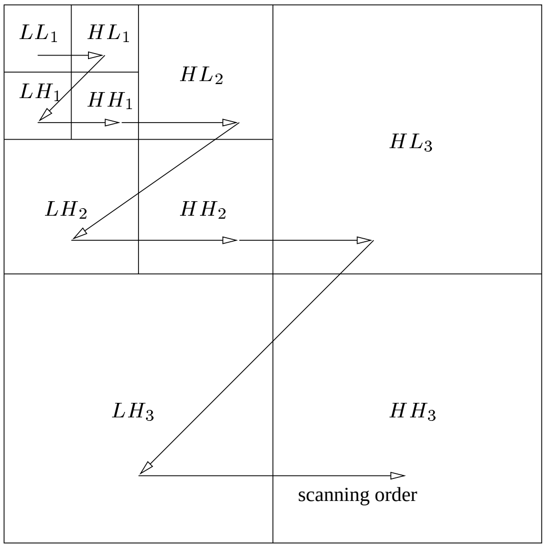

Figure 10: Scanning order of a 3-level DWT's coefficients

old is halved after each execution of dominant and subordinate passes. This corresponds, in a sense, to the bit-plane coding of the wavelet coefficients and provides a coarse-to-fine multiprecision logarithmic representation of coefficient amplitude information.

3.3 Set Partitioning in Hierarchical Trees (SPIHT)

Shapiro's EZW coding scheme generates an embedded bitstream. It achieves low bit-rate image compression and beats the standard image coding technique JPEG [17], in terms of the peak signal to noise ratio (PSNR) at a desired bit- rate. Although the codeword for zerotree root (ZTR) is quite useful in avoiding the encoding of insignificant information, it was observed that the bitstream generated by EZW contains some redundant information that can be avoided. As described earlier, if a coefficient at a particular coordinate is insignificant, with respect to an absolute threshold, but is not a zerotree root, it is encoded as IZ. It was found that there are quite lot of IZ codewords in the EZW- generated bitstream. This means that EZW spends quite some portion of the bit-budget in encoding the insignificant information.

The set partitioning in hierarchical trees (SPIHT) image compression scheme of Said and Pearlman [4] could be regarded as an extension of EZW. It eliminates the redundancies in EZW-generated bitstream to some extent (explained by factor 1 in the end of this section). The SPIHT coding algorithm employs two concepts from EZW: the parent-offspring dependencies (or self-similarities) across the scales, and the bit-plane coding of significant wavelet coefficients during the refinement stages (like the subordinate pass in the EZW).

The spatial orientation trees of SPIHT are similar to the zerotrees of EZW. Both the coder and decoder maintain three lists: list of insignificant sets (LIS), list of insignificant pixels (LIP), and list of significant pixels (LSP). Following are the four kinds of sets used in SPIHT. The set $\mathcal{H}$ denotes the set of all coordinates in the highest pyramid level. $\mathcal{O}(i,j)$ denotes the set of coordinates of all offspring of the node (i,j), whereas the set of all descendants of node (i,j) is denoted by $\mathcal{D}(i,j)$. Another type of set is $\mathcal{L}(i,j)$ which is defined as $\mathcal{L}(i,j) = \mathcal{D}(i,j) - \mathcal{O}(i,j)$. all the three sets $\mathcal{O}(i,j)$, $\mathcal{D}(i,j)$ and $\mathcal{L}(i,j)$ make use of the parent-child dependencies among subbands across the scales.

The sorting and refinement stages employed in the SPIHT are a replica of the dominant and subordinate passes of the EZW. The sorting stage of SPIHT coder, like the dominant pass of an EZW coder, sends information regarding the significance of coefficients to its decoder. The only difference is that EZW employs the scanning order described in Section 3.2, whereas the SPIHT utilises the set partitioning rules to dictate the scanning order. The refinement stage of the SPIHT, like the subordinate pass of the EZW, refines the magnitude available to the decoder.

We note that the set partitioning rules, used in SPIHT, are essentially the same as those implicit in EZW coding scheme. The initial partition adds all coordinates (i, j) to the list *LIP* and $\mathcal{D}(i, j)$ to the list *LIS*, where (i, j) are the coordinates belonging to $\mathcal{H}$. This is exactly how the EZW coder starts. The SPIHT coder then proceeds as follows. If found significant, the set $\mathcal{D}(i, j)$ is partitioned into the four offspring of (i, j) and the set $\mathcal{L}(i, j)$. The set $\mathcal{L}(i, j)$, if significant, is partitioned into four sets $\mathcal{D}(k, l)$, where (k, l) belongs to the set $\mathcal{O}(i, j)$. Both of these partitioning rules are quite like the ones assumed during the dominant pass of EZW, although in a different scanning order.

Our findings are that following factors contribute towards the improved compression performance of SPIHT over EZW:

1. One bit is required, in SPIHT, to encode whether or not a set of coordinates is significant, with respect to an absolute threshold. This helps SPIHT in eliminating the redundancies in an EZW-generated bitstream, discussed earlier in this section.

2. As reported in [4], the SPIHT uses the 9/7-pair biorthogonal wavelet filters of [11]. An improvement in the compression performance is inevitable, due to the better frequency localization properties of these filters as compared to the 9-tap near-orthogonal wavelet filters of [10], which are used in EZW.

3. Like EZW, the adaptive arithmetic coding utilised in SPIHT also exploits the statistical dependence between significance of a pixel and the significance of the set of all of its descendants. The SPIHT image compression scheme also exploits the dependencies between the magnitudes of adjacent pixels by keeping together a group of 2×2 adjacent pixels and then using an *order-m* arithmetic coder, where m is the number of insignificant pixels in that group.

3.4 Fast Adaptive Wavelet Packet Image Compression

The wavelet compression methods, such as the ones described above, avoid the blocking artifacts which become visible at low bit rates if a block transform coding method (such as the DCT) is used. Long oscillatory patterns require many of the fine scale coefficients which describe the rapid variations of intensity. In order to describe such patterns, much larger libraries of wavelets, called wavelet packets [18], have been developed. This library is composed of functions (or atoms) having different time and frequency localization properties. The wavelet packets, in an image, are patterns that can vary in scale, frequency, and location.

A general wavelet packet basis can be adaptively tailored to the image frequency content at the cost of losing the parent-offspring dependencies, as described in Section 3.2. In a recent work [5], adaptive wavelet packets were used for the compression of the images. In order to select an orthonormal basis inside the wavelet packet library, each wavelet packet is matched to the image, and the best matches are selected based upon a particular criterion. This selected collection of patterns is called the "best basis". Meyer et al. used the first-order entropy as the selection criterion (or the cost function), as suggested in [19]. This differs from the approach of selecting the best basis described by Ramachandran and Vetterli who use the rate-distortion function as a selection criterion [20]. But the pruning algorithm of [20] is expensive from a computational viewpoint. Meyer et al. propose the best basis selection which requires single pruning of the wavelet packet tree. The coefficients in the tree are thresholded and each node is either accepted or rejected depending upon the cost of the node, the first-order entropy. This helps in speeding up the selection of best basis.

Moreover, they propose a 2-D factorization of 9-7 biorthogonal filters of [11] into four filters, each of length 3, reducing the number of multiplications and additions. This is quite similar to the factorization used by Wim

Sweldens in [21]. Finally, the compression is achieved by quantizing the wavelet packet coefficients after organizing them in an increasing frequency (or decaying magnitude) order. The quantized values are bit-plane coded using an adaptive arithmetic coder. The experimental results show that this technique, despite being fast, outperforms the SPIHT in most cases. Due to the fact that the best basis was chosen in order to minimize the entropy, and not the distortion, ringing artifacts are experienced, sometimes, on the borders of smooth regions.

3.5 The Generalized Lapped Biorthogonal Transform (GLBT)

Recently, a new lattice structure has been proposed [22] which spans a large class of the *M-channel* linear phase perfect reconstruction filter banks (aka LPPRFB). The new lattice represents a family of generalized lapped biorthogonal tranform (GLBT) with arbitrary integer overlapping factor, by relaxing the orthogonality constraint. Experimental results for a 16-channel GLBT (which was optimised for coding gain, DC and mirror frequency attenuation) using this lattice show that this technique outperforms SPIHT by a lift in the PSNR of as much as 2.65dB.

4 The Compression Algorithm

Ever since Shapiro introduced the embedded coding of images using the zerotrees of the wavelet coefficients, there have been many suggestions made for the improvement of its compression performance. The set partitioning in hierarchical trees (SPIHT) algorithm of Amir Said and William Pearlman is also an extension of EZW, as described earlier in Section 3.3. In 1996, it was suggested by Barreto et al. [23] that the efficiency of EZW can be improved by introducing unique scanning directions for each of three types of high frequency subbands and using a higher-order arithmetic coder. (We note, however, that the coding gains reported were mainly due to the 9-7 biorthogonal filter of [11] and using an order-3 arithmetic coder.) Liang [24] redefined, in 1997, the symbol set as $\{ZTRZ, ZTRS, IZ, IS\}$, where ZTRZ denotes a zerotree root with zero value (same as ZTR in EZW), ZTRS denotes a zerotree root with significant value, IZ is similar to the IZ in EZW, and IS represents an isolated significant coefficient whose descendants are not all zero.

We propose a progressive image coding algorithm which is based upon the *augmented zerotrees* of wavelet coefficients. The augmented zerotrees are different from the spatial orientation trees of SPIHT in two ways. An augmented zerotree helps us to exploit the dependencies, whatsoever, between the coefficients in the lowest frequency subband and the corresponding coefficients at the same spatial location in three coarsest scale high frequency subbands. Secondly, it takes into account the zerotree root nodes which are not insignificant themselves. The scheme employs a recursive detection (or construction) of the augmented zerotrees by the coder (or decoder) in an efficient manner. Our method is memory efficient since it does not require us to maintain more than one list of coefficients. There is no sorting required for this list by the coder or the decoder. The same list helps us deal with the coefficients already detected so as to reduce the distortion at a target bit rate. We introduce a new scanning order to exploit the inter-band dependencies between the wavelet coefficients.

4.1 The Augmented Zerotrees

Our work is based on a second hypothesis, in addition to Shapiro's hypothesis described in Section 3.2, that *if a coefficient at the lowest frequency subband is insignificant, it is more likely that the corresponding coefficients at the same spatial location in three coarsest scale high frequency subbands would be insignificant as well.* This hypothesis was proved empirically. It was observed that the improvement is more for smaller number of stages, which correspond to the higher value of threshold. As the threshold decreases, after a certain point, the improvement in the bit-rate starts declining. This implies that utilising these dependencies would help in improving the compression performance at low bit rates.

We augment the symbol set of the entropy coder by adding two more symbols to it, to take into account the zerotree root nodes which are not insignificant themselves. This helps in reducing the generation of redundant information for the zerotree root node which is not insignificant itself. Therefore, the augmented symbol set consists of the symbols POS, NEG, IZ, ISZTR (similar to the ZTR in EZW, which represents a zerotree root node which is insignificant itself as well), and two new symbols PSZTR, and NSZTR which represent zerotree root nodes, that are either positive significant or negative significant respectively.

It should be noted here that we do not use any explicit zerotree data structure. The construction of zerotrees, as described later in Section 4.4, is done in an efficient recursive manner without using any zerotree data structure. A graphical illustration of the augmented zerotrees is shown in Figure 11.

4.2 The Coding Stages

The compression algorithm executes two stages iteratively, for every new value of the threshold. These are: the detection stage and the fine-tuning stage. The *detection stage* determines the significance of coefficients, detects the augmented zerotrees, and encodes the significance map using the new symbol set, described in Section 4.1. The *fine-tuning stage* refines the values of coefficients already detected, by sending single bit in a bitplane coding fashion. These stages are described in greater detail, in Section 4.7.

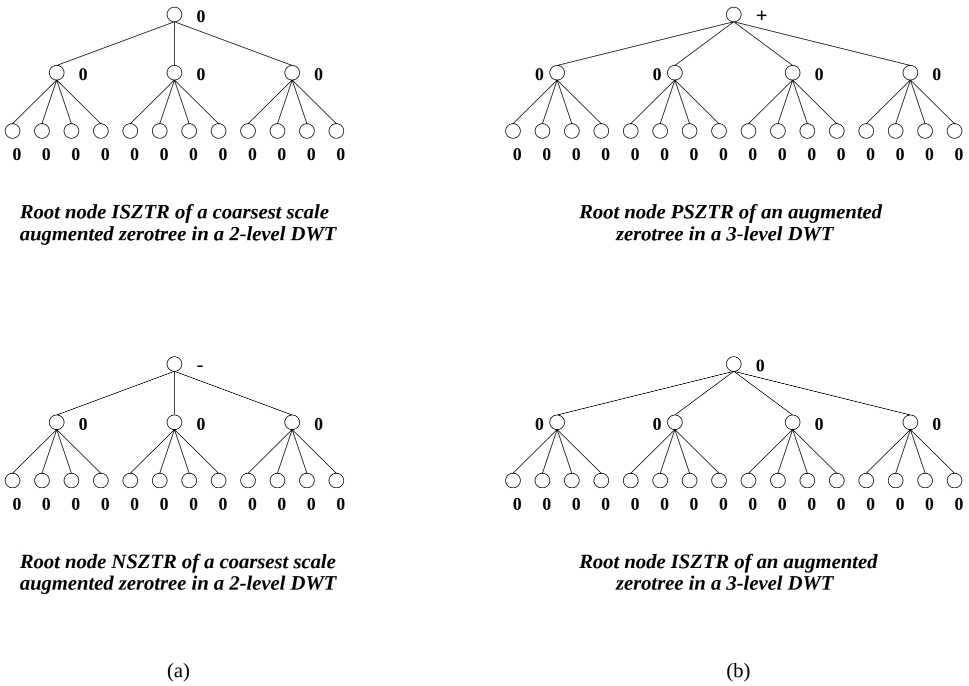

Figure 11: A graphical illustration of the augmented zerotrees

4.3 The List of Detected Coefficients (LDC)

There is an issue of what to do with a coefficient that has already been detected, as being significant, during one of the previous detection stages. One clear solution is to assign it the value of zero, immediately after it has been encoded as being significant, to keep a copy of its coordinates and actual value for further fine tuning stages, and to ignore it during further detection stages. This approach has the disadvantage that if the coefficient is ignored, there is a risk of losing the opportunity of finding a zerotree at the coordinates of this coefficient. Our approach is not to ignore this coefficient during further detection stages, in order to exploit the chances of occurrence of a zerotree with this coefficient as root of the zerotree. It was observed that the improvement, after utilising this approach, is more for low bit rates, mainly due to the fact that the latter approach succeeds in finding more zerotrees as compared to the former one. This improvement seems to cease at high bit rates, or at low values of the threshold, when the finest scale coefficients start getting detected.

List of Detected Coefficients. In order to implement the above idea, a list of detected coefficients $\mathcal{D}$ is maintained. It contains the coordinates and original value of the coefficient detected as significant. During every detection stage, entries are added to this list in the scanning order described later in Section 4.5.

It is to be noted that the order in which the entries are added to $\mathcal{D}$ in the decoder should be the same as the one employed by the coder. Our coding algorithm requires to maintain no other list but $\mathcal{D}$, as compared to two lists (dominant and subordinate lists) in the EZW and three lists (LIS, LIP, and LSP) in the SPIHT. We do not perform any sorting operation on the list $\mathcal{D}$.

4.4 Detection/Construction of the Augmented Zerotrees

We present an algorithm which constructs the augmented zerotrees during the detection part of the detection stage in an efficient recursive manner so that there is no need for the zerotree construction at the finer levels. This part of the algorithm looks for augmented zerotrees (if any) and marks the significance of all of its elements, except the root node, as DNC (Do Not Code). All such coefficients whose significance has been marked as DNC are ignored during the coding part of the detection stage. The augmented zerotree detection algorithm, employed by the coder, is as follows.

ALGORITHM – I

```
Boolean IsZerotree(x,y,Level)
  if Level is FinestScale
    if Significance(x,y) is IZ
      returnTRUE
    Otherwise
      returnFALSE
  if IsZerotree(2x,2y,Level+1) AND IsZerotree(2x+1,2y,Level+1) AND
    IsZerotree(2x,2y+1,Level+1) AND IsZerotree(2x+1,2y+1,Level+1)
      if Significance(x,y) is IZ
        Significance(x,y) ← ISZTR
        Change the significance of (2x,2y), (2x+1,2y),
        (2x,2y+1), and (2x+1,2y+1) to DNC
        returnTRUE
      if Significance(x,y) is POS
        Significance(x,y) ← PSZTR
        Change the significance of (2x,2y), (2x+1,2y),
        (2x,2y+1), and (2x+1,2y+1) to DNC
        returnFALSE
```

if Significance(x,y) is NEG

Significance(x,y) $\leftarrow$ *NSZTR*

Change the significance of (2x,2y), (2x+1,2y),

(2x,2y+1), and (2x+1,2y+1) to DNC

returnFALSE

Otherwise

returnFALSE

$\odot$

A short note about the decoder here. The decoder operates in the same sequence. When the decoder reads one of the augmented zerotree root nodes, it constructs the zerotree in the same recursive manner. Following is the algorithm for constructing the augmented zerotrees when the decoder reads one of these symbols: ISZTR, PSZTR, or NSZTR.

<u>ALGORITHM – II</u>

ConstructAugmentedZerotree(x,y,Level)

if Level is FinestScale

Significance(x,y) $\leftarrow$ *DNC*

return

else

Change the significance of (2x,2y), (2x+1,2y),

(2x,2y+1), and (2x+1,2y+1) to DNC

ConstructAugmentedZerotree(2x,2y,Level+1)

ConstructAugmentedZerotree(2x+1,2y,Level+1)

ConstructAugmentedZerotree(2x,2y+1,Level+1)

ConstructAugmentedZerotree(2x+1,2y+1,Level+1)

$\odot$

4.5 Subband Dependent Scanning Order

The high frequency subbands of an image represent the edges at a particular orientation and scale. As described earlier in Section 2.4, the subbands $\mathbf{HL_k}$, $\mathbf{LH_k}$, and $\mathbf{HH_k}$ represent the edges in horizontal, vertical, and diagonal directions respectively. It sounds, therefore, reasonable to encode these subbands in an order compatible with their respective orientations.

We propose the scanning order, illustrated in Figure 12. According to this approach, the $\mathbf{HL_k}$ subbands are scanned in the horizontal direction, the $\mathbf{LH_k}$ subbands are scanned in the vertical direction, and the $\mathbf{HH_k}$ subbands are scanned in the diagonal direction. The diagonal scan is similar to the one employed by JPEG.

4.6 Embedded Coding of the Augmented Zerotrees

A simple rule for generating the embedded bitstream is that the more significant information should be encoded before the less significant. It is to be noted that if $\mathbf{T}$ is a unitary transform, then encoding the larger magnitude coefficients (before the smaller magnitude ones) decreases the mean square error (MSE) between the original and the reconstructed image. This corresponds to the bit-plane method for progressive transmission, described in [25], and has been utilised in our research as well.

In this work, a uniform quantization is applied to the wavelet coefficients using an absolute deterministic threshold across all the subbands. A coefficient is said to be *insignificant* if its absolute value is below certain threshold, and it is called *significant* otherwise. The benefit of applying an absolute deterministic threshold across all the subbands is twofold: first, it ensures that a coefficient in a lower energy subband is not ignored due to its energy statistics. Secondly, it produces many zerotrees, thus providing more compression. The *detection* and the *fine tuning* stages follow the quantization. The output is then entropy coded using a context-based adaptive

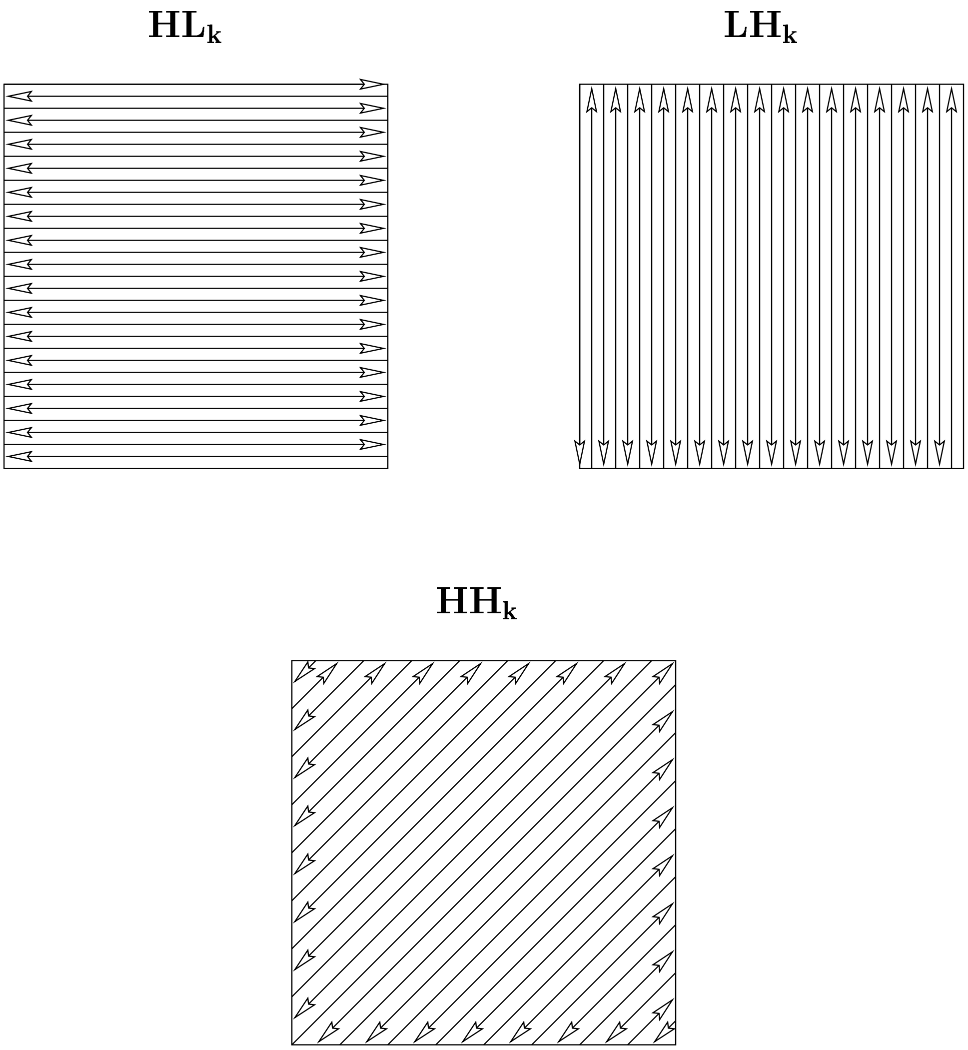

Figure 12: New scanning order for high frequency subbands

arithmetic coder, as described in [26]. The deterministic threshold is halved, after every stage, and this cycle (quantization, detection, fine tuning, and entropy coding) is iterated until a target rate is achieved.

4.7 The Coder Algorithm

Following is a pseudocode for the compression algorithm, excluding the wavelet transform and the entropy coding parts. The list of detected coefficients $\mathcal{D}$ is empty in the beginning, requiring no memory at all. It grows as the coding/decoding proceeds when the wavelet coefficients get detected as being significant, with respect to one of the threshold values T_i. Let c_{ij} denotes the values of wavelet transform coefficient at the coordinate (i, j).

1. **Initial Quantization**

   Set $T = \frac{max_{(i,j)}\{|c_{ij}|\}}{2}$.

2. **Detection Stage**

   a) For each coefficient c_{ij} at the coordinate (i, j), where $(i, j) \in$ set of all coordinates of the image, do the following in the scanning order described in Section 4.5:

      If $|c_{ij}| \ \geq \ T$ (ie. c_{ij} is *significant*),

      Add $\Big((i, j), c_{ij}\Big)$ to the list $\mathcal{D}$.

   b) For each coefficient c_{ij} at the coordinate (i, j), where $(i, j) \in H_c$, the set of all coordinates in the three coarsest scale high frequency subbands, do:

      If $|c_{ij}| \ < \ T$ (ie. c_{ij} is *insignificant*),

      Detect the augmented zerotrees (if any) with root at (i, j) or at any of its descendants.

   c) For each coefficient c_{ij} at the coordinate (i, j), where $(i, j) \in L_c$, the set of all coordinates in the lowest frequency subband, do:

If c_{ij} is *insignificant*,

Detect the coarsest scale augmented zerotrees

(if possible) with root at (i, j).

d) For each coefficient c_{ij} at the coordinate (i, j), where $(i, j) \in$ set of all coordinates of the image, do the following in the scanning order described in Section 4.5:

- If c_{ij}'s significance is *DNC*,

  Do nothing.

- If c_{ij} is *significant*,

  * If all four of the offsprings of (i, j) have been detected as ISZTR,

    Output PSZTR or NSZTR, depending upon the sign of c_{ij}.

    Otherwise,

    Output POS or NEG, depending upon the sign of c_{ij}.

- If c_{ij} is *insignificant*,

  * If all four of the offsprings of (i, j) have been detected as ISZTR,

    Output ISZTR.

    Otherwise,

    Output IZ.

3. **Fine-Tuning Stage**

- For each coordinate $(i, j) \in \mathcal{D}$, do the following:

  - If c_{ij} is in lower half of the corresponding dequantized interval,

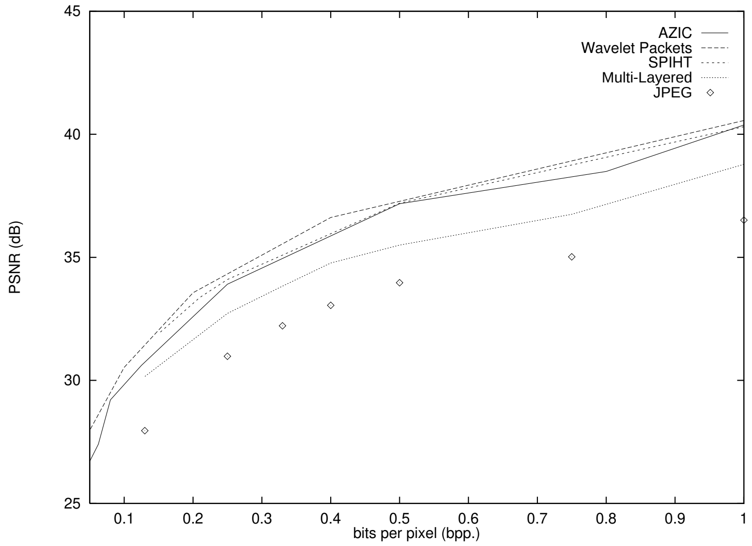

Figure 13: Rate-distortion curves for 512 $\times$ 512 *Lena* image

Output 0.

Otherwise,

Output 1.

4.8 Experimental Results

The coder takes the input image file and generates an output data file for a specified target bit rate. The decoder takes this data file as input and tries to reconstruct the original image at a given bit rate. Due to the lossy compression, there would, of course, be some distortion. The distortion measure used, here, is the peak-to-signal-ratio (PSNR) given by,

$$PSNR = 10log_{10}\Big(\frac{255^2}{MSE}\Big). \tag{17}$$

The experimental results for 512$\times$512 *Lena* image are shown in Table 1. The wavelet filters used were the 9-7 biorthogonal filters of [11], due to

Target bitrate ($bpp.$)	Compression ($: 1$)	PSNR (dB)	Execution Time (s)	
			Coder	*Decoder*
2	4	44.74	18	6
1	8	40.38	8	5
0.8	10	38.49	7	5
0.5	16	37.18	6	5
0.25	32	33.91	5	4
0.125	64	30.61	5	3
0.080	100	28.12	4	3
0.0625	128	27.41	4	3

Table 1: Compression performance for 512×512 *Lena* image
Target bits per pixel (bpp.), compression ratio, PSNR in dB, compression time in seconds, and decompression time in seconds, for the *Lena* image.

their excellent frequency localization properties. A rate-distortion curve for AZIC, alongwith some other coders is shown in Figure 13. These results show that the coder outperforms the multi-layered image compression [27] and the DCT-based JPEG[1] [17], at all bit-rates. The coder performance is very close to, and sometimes bit better than, that of the SPIHT [4]. The adaptive wavelet packets scheme [5] was unbeaten by a small margin. The timing results, obtained for a 143MHz Ultra Sparc machine, presented in Table 1 are also encouraging.

4.9 Further Improvements

We propose that the quantization noise in the DC subband can be reduced by filtering the lowest frequency band of the decoded wavelet transform of the image, resulting in a better reconstructed image. Moreover, a higher-order entropy coder may be used in order to efficiently exploit the inter-band and intra-band similarities, rather than a simple order-1 arithmetic coder employed in our work.

[1]JPEG images with different bit rates were created using the `ixv` utility.

The issue of best basis selection [19] for a particular image or for a particular rate-distortion measure [20] still remains to be investigated. A nonseparable 2D wavelet transform, like the one proposed in [28], would be desirable for efficiently transforming the image with better frequency separation. In order to further speed up the overall algorithm, the factorized nonseparable fast filters, such as the ones proposed in [5] can be used. Trellis coded quantization (TCQ) has shown to be performing quite effectively for image compression [29, 30]. A move from the uniform scalar quantization towards the TCQ might be worthwhile. In order to achieve the objective of lossless compression, the lifting scheme [21] has been used [31], since the main hurdle in lossless compression otherwise is the floating point coefficients generated by the ordinary wavelet transform. The factorization in [32] converts every finite filter wavelet transform into lifting steps, thus allowing for an integer to integer version of the wavelet transform.

Recent work [33] on the experimental evaluation of flexible parsing [34], for dictionary based data compression schemes, suggests that this scheme is quite useful for the entropy coding of data with small alphabet, which is the case with quantized indices, as described in Section 3.1. It has been shown that this scheme achieves the entropy much faster than the standard UNIX™ utilities `gzip` and `compress`, which are based upon the LZ77 [35] and the LZW implementation [36] of the LZ78 [37] data compression schemes, for data files emanating from sources with small alphabet. Moreover, it needs to be investigated if the dictionary based properties of this scheme can be utilised for exploiting the similarities across the subbands.

5 Summary

We have addressed the issue of how to improve the performance of zerotree-style image coders, from both the computation and compression standpoints. A brief discussion of the performance of zerotree image coders EZW and its extension SPIHT was presented, after a brief review of the wavelet transforms. We argued that the spatial orientation trees and the set partitioning rules of SPIHT are similar to those in EZW, with scanning order being the only difference. We have proposed a progressive image coding scheme based on the augmented zerotrees of the wavelet coefficients. We also suggested efficient data structures to speed up the coder execution time. The basic coder algorithm and the algorithms for the detection and construction of augmented zerotrees were also presented.

References

[1] I. Daubechies. *Ten Lectures on Wavelets.* SIAM, Philadelphia, PA, 1992. Notes from the 1990 CBMS-NSF Conference on Wavelets and Applications at Lowell, MA.

[2] A. Graps. An Introduction to Wavelets. *IEEE Computational Sciences and Engineering*, 2:50–61, 1995.

[3] J. Shapiro. Embedded Image Coding Using Zerotrees of Wavelet Coefficients. *IEEE Transactions on Signal Processing*, 41:3445–3462, 1993.

[4] A. Said and W. A. Pearlman. A New Fast and Efficient Image Codec Based on Set Partitioning in Hierarchical Trees. *IEEE Transactions on Circuits and Systems for Video Technology*, 6:243–250, 1996.

[5] F. G. Meyer, A. Averbuch, J-O. Stromberg, and R. R. Coifman. Fast Adaptive Wavelet Packet Image Compression. In *Proceedings Data Compression Conference, Snowbird, Utah.* IEEE Computer Society Press, 1998.

[6] M. Vetterli and J. Kovacevic. *Wavelets and Subband Coding.* Prentice Hall, Englewood Cliffs NJ, USA, 1995.

[7] M. L. Hilton, Bjorn D. Jawerth, and A. Sengupta. Compressing Still and Moving Images with Wavelets. *Multimedia Systems*, 43:218–227, 1995.

[8] G. Strang and T. Nguyen. *Wavelets and Filter Banks.* Wellesley-Cambridge Press, revised edition edition, 1997.

[9] I. Daubechies. Orthonormal Bases of Compactly Supported Wavelets. *Communications on Pure and Applied Mathematics*, XLI:910–996, 1988.

[10] E. H. Adelson, E. Simoncelli, and R. Hingorani. Orthogonal Pyramid Transforms for Image Coding. In *Proceedings SPIE, Visual Communications and Image Processing II*, volume 845, pages 50–58, 1987.

[11] A. Cohen, I. Daubechies, and J. Feauvea. Bi-orthogonal bases of compactly supported wavelets. *Comm. Pure and Applied Mathematics*, 45:485–560, 1992.

[12] R. Wilson. BMVC93 Tutorial Notes on Wavelet Transforms. In *British Machine Vision Conference*, 1993.

[13] J. W. Woods. *Subband Image Coding*. Kluwer, Boston MA, USA, 1991.

[14] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies. Image coding using Wavelet Transform. *IEEE Transactions on Image Processing*, 1:205–219, 1992.

[15] A. S. Lewis and G. Knowles. Image Compression Using the 2-D Wavelet Transform. *IEEE Transactions on Image Processing*, 1:244–250, 1992.

[16] A. S. Lewis and G. Knowles. A 64 kB/s Video Codec Using the 2-D Wavelet Transform. In *Proceedings Data Compression Conference, Snowbird, Utah*. IEEE Computer Society Press, 1991.

[17] W. Pennebaker and J. Mitchell. *JPEG Still Image Data Compression Standard*. Van Nostrand Reinhold, 1992.

[18] R. R. Coifman and Y. Meyer. *Wavelets and their Applications*, pages 125–150. Jones and Barlett, 1992.

[19] R. R. Coifman and M. V. Wickerhauser. Entropy-Based Algorithms for Best Basis Selection. *IEEE Transactions on Information Theory*, 38(2):713–718, 1992.

[20] K. Ramachandran and M. Vetterli. Best Wavelet Packet Bases in a Rate-Distortion Sense. *IEEE Transactions on Image Processing*, 2:160–175, 1993.

[21] Wim Sweldens. The Lifting Scheme: A Construction of Second Generation Wavelets. *SIAM Journal of Mathematical Analysis*, 29(2):511–546, 1997.

[22] Trac D. Tran, Ricardo de Queiroz, and Troung Q. Nguyen. The generalized lapped biorthogonal transform. In *Proceedings IEEE ICASSP, Seattle*, May 1998.

[23] C. S. Barreto and G. V. Mendonca. Enhanced Zerotree Wavelet Transform Image Coding Exploiting Similarities Inside Subbands. In *Proceedings IEEE Conf. on Image Proc.*, volume II, pages 549–551. IEEE Press, 1996.

[24] Jie Liang. Highly Scalable Image Coding for Multimedia Applications. In *Proceedings of the Fifth ACM Intl. Multimedia Conference*, pages 11–19, 1997.

[25] M. Rabbani and P. W. Jones. *Digital Image Compression Techniques.* SPIE Optical Engineering Press, 1991.

[26] I. Witten, R. Neal, and J. Cleary. Arithmetic Coding for Data Compression. *Communications of the ACM*, 30:520–540, 1987.

[27] F. G. Meyer, A. Averbuch, J-O. Stromberg, and R. R. Coifman. Multilayered image compression. In *Wavelet Applications in Signal and Imaging Processing VI*, 1998.

[28] J. Kovacevic and M. Vetterli. Nonseparable Two- and Three-Dimensional Wavelets. *IEEE Trans. on Signal Processing*, 43:1269–1273, 1994.

[29] A. Bilgin, P. J. Sementilli, and M. W. Marcellin. Progressive Image Coding Using Trellis Coded Quantization. *IEEE Transactions on Image Processing*, 1998. submitted.

[30] M. W. Marcellin and T. R. Fischer. Trellis Coded Quantization of Memoryless and Gauss-Markov Sources. *IEEE Transactions on Communications*, 38:82–93, 1990.

[31] R. Calderbank, I. Daubechies, W. Sweldens, and B.-L. Yeo. Losless image compression using integer to integer wavelet transforms. In *Proceedings IEEE Conf. on Image Proc.* IEEE Press, 1997.

[32] I. Daubechies and W. Sweldens. Factoring wavelet transforms into lifting steps. *Journal of Fourier Analysis Applications*, 4(3):245–267, 1998.

[33] Y. Matias, N. Rajpoot, and S. C. Sahinalp. Implementation and experimental evaluation of flexible parsing for dynamic dictionary based data compression. In *Proceedings Workshop on Algorithmic Engineering (WAE98)*, August 1998.

[34] Y. Matias and S. C. Sahinalp. Optimal parsing for dictionary based data compression. Technical report, Bell Laboratories, 1997.

[35] A. Lempel and J. Ziv. A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory*, 23(3):337–343, 1977.

[36] T. A. Welch. Techniques for high-performance data compression. *IEEE Computer*, pages 8–19, 1984.

[37] A. Lempel and J. Ziv. Compression of individual sequences via variable-rate coding. *IEEE Transactions on Information Theory*, 24(5):530–536, 1978.