

James S. Walker "Wavelet-Based Image Compression"
The Transform and Data Compression Handbook
Ed. K. R. Rao et al.
Boca Raton, CRC Press LLC, 2001

Chapter 6

Wavelet-Based Image Compression

James S. Walker

University of Wisconsin-Eau Claire

Truong Q. Nguyen

Boston University

6.1 Introduction

One of the most successful applications of wavelet methods is transform-based image compression (also called coding). Such a coder [depicted in Fig. 6.1(a)] operates by transforming the data to remove redundancy, then quantizing the transform coefficients (a lossy step), and finally entropy coding the quantizer output. Because of their superior energy compaction properties and correspondence with the human visual system, wavelet compression methods have produced superior objective and subjective results [5]. Since a wavelet basis consists of functions with both short support (for high frequencies) and long support (for low frequencies), large smooth areas of an image may be represented with very few bits, and detail added where it is needed.

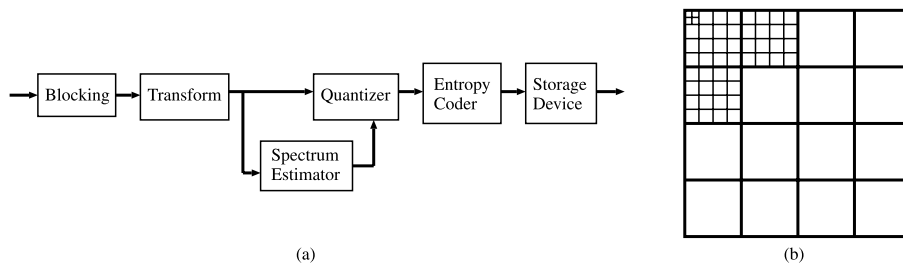


FIGURE 6.1

(a) Transform-based coder. (b) Subband decomposition used in the FBI fingerprint compression standard.

Both orthogonal [73] and bi-orthogonal [1, 70] wavelets have been used for image compression. The recent FBI fingerprint compression standard [70] uses symmetric dyadic wavelets and significantly outperforms the JPEG (Joint Picture Expert Group) standard [38] at compression ratios above 10:1. Fig. 6.1(b) shows the subband decomposition used in the FBI fingerprint compression standard. Interestingly, the wavelet tree used in the FBI specification is a predominantly 4-channel decomposition achieved by cascading 2-channel filter banks.

Most high-quality algorithms today use some form of transform coder. One widely used standard is the JPEG compression algorithm, based on the discrete cosine transform (DCT) [38]. The image is partitioned into 8×8 blocks, each of which is then transformed via a tensor product of two 8-point DCTs. The transform coefficients are then arranged into 64 subbands, scalar-quantized, and adaptively Huffman coded. The JPEG algorithm yields good results for compression ratios of 10:1 and below (on 8-bit gray-scale images), but at higher compression ratios the underlying block nature of the transform begins to show through the compressed image. By the time compression ratios have reached 24:1, only the DC (lowest frequency) coefficient is getting any bits allocated to it, and the input image has been approximated by a set of 8×8 blocks. Consequently, the decompressed image has substantial blocking artifacts for medium and high compression ratios.

Researchers have applied subband coding to images for over a decade [69, 60]; their results reached a new level with the advent of the wavelet transform. Wavelet methods involve *overlapping* transforms with varying-length basis functions. The overlapping nature of the transform (each pixel contributes to several output points) alleviates blocking artifacts, while the multiresolution character of the wavelet decomposition leads to superior energy compaction and perceptual quality of the decompressed image. Furthermore, the multiresolution transform domain means that wavelet compression methods degrade much more gracefully than block-DCT methods as the compression ratio increases. One wavelet algorithm, the embedded zerotree wavelet (EZW) coder, yields acceptable compression at a ratio of 100:1 [48]. The EZW coder is described in detail in Section 6.3.2.

Section 6.2 briefly reviews the concepts of dyadic wavelet transform and multiresolution representation and their design and implementation using two-channel filter banks. Further details can be found in Mallat [26] and Strang and Nguyen [53]. Readers familiar with wavelet theory could skip this section and proceed to Section 6.3 where several coding schemes based on zerotree wavelet coding are described and compared to JPEG.

6.2 Dyadic Wavelet Transform

The dyadic wavelet transform is an octave-band representation for signals; the discrete wavelet transform may be obtained by iterating a two-channel filter bank on its lowpass output. This multiresolution decomposition of a signal into its coarse and

detail components is useful for data compression, feature extraction, and denoising. In the case of images, the wavelet representation is well-matched to psychovisual models, and compression systems based on the wavelet transform yield perceptual quality superior to other methods at medium and high compression ratios. Furthermore, the multiresolution nature of the wavelet transform enables fast browsing of image databases (the user may decompress only the coarsest scale representation of an image to decide whether he or she wants to examine it at a finer resolution).

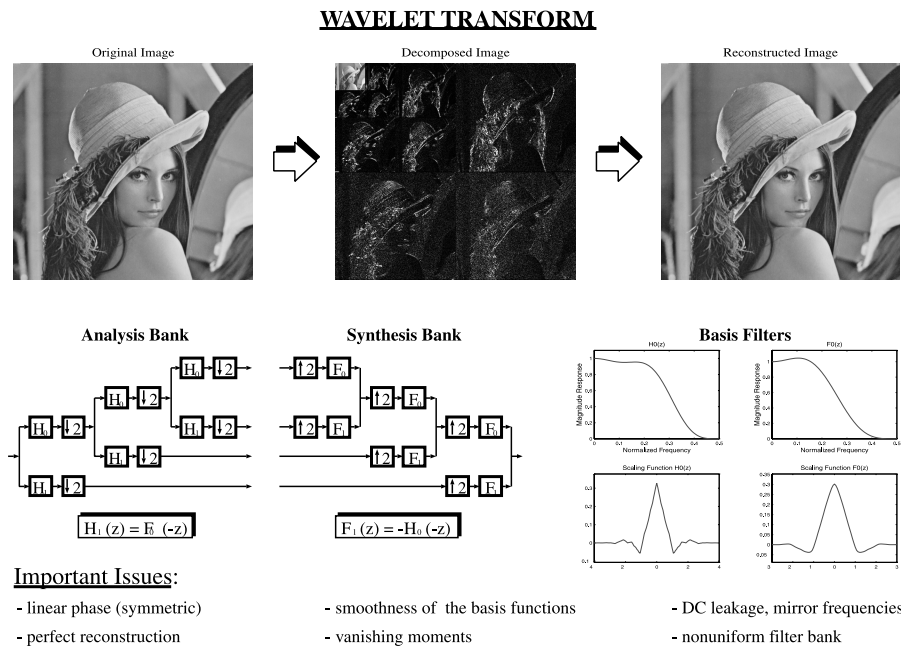


FIGURE 6.2

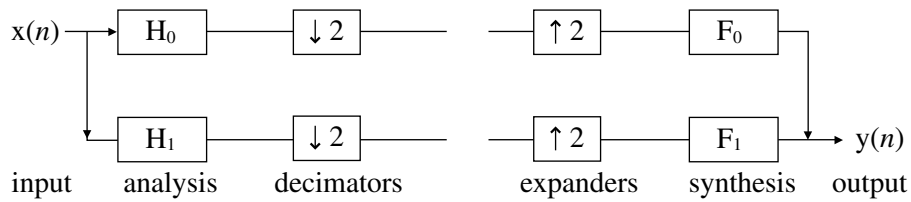
Dyadic wavelet transform, multiresolution representation, implementation using two-channel filter bank and filter characteristics. Reproduced by Special Permission of *Playboy* magazine. Copyright ©1972, 2000 by Playboy.

Fig. 6.2 shows an original image, its wavelet representation, and the reconstructed image without coefficient quantization. Since the wavelet transform is invertible, the reconstructed image is exactly the same as the original image. The decomposed image (wavelet representation) shows a coarse approximation image in the upper left corner and several detail images at various scales. As the scale changes, the sub-image size changes. This is multiresolution and is enabled by the downsampling operation in the structure shown in the bottom left portion of Fig. 6.2. The coarse approximation and detail images are computed by first filtering the original image by lowpass and highpass filters $H_0(z)$ and $H_1(z)$, respectively. The filtered images are then downsampled by a factor of 2 to preserve the total image size. This is reflected in the structure as a two-channel filter bank.

This filter bank is repeated on the coarse approximation image since it still has large energy content (*coarse approximation image* is also referred to as the *all-lowpass subband* in Section 6.3). The structure shows a three-level filter bank and the corresponding decomposed image shows a three-level wavelet decomposition. The above process is repeated column-wise and row-wise. Throughout this chapter, we use the following terminology: *horizontal subband* to denote lowpass filtering on rows and highpass filtering on columns, *vertical subband* to denote highpass filtering on rows and lowpass filtering on columns, and *diagonal subband* to denote highpass filtering on both rows and columns. As shown, most of the energy is concentrated in the coarse approximation image, which is $\frac{1}{64}$ the original image size. The detailed images have small coefficients, as observed from the dark regions in the detailed images. The resulting multiresolution representation enables the user to treat each subband independently; for example, he or she may selectively allocate bits depending on the energy content (variance) of each subband, and the subsequent perceptual or algorithmic processing. Using the above multiresolution representation for image compression, one needs to develop an efficient coding algorithm for the locations of these small coefficients in the detailed images. This topic is discussed in detail in Section 6.3.

The bottom right portion of Fig. 6.2 shows the magnitude frequency responses of the lowpass filters $H_0(z)$ and $F_0(z)$ and their corresponding scaling functions. These are the Daub 9/7 filters used in the FBI fingerprint standard [70] as well as in the JPEG2000 standard. These filters are designed appropriately such that the whole filter bank is invertible and the corresponding basis functions are smooth. The perceptual quality of the reconstructed image depends on both the basis functions and the coding algorithm. Note that both lowpass filters have zeros at frequency π (in fact, they both have four zeros at π in this example). In general, filters with more zeros at π yields smoother basis functions [53].

6.2.1 Two-Channel Perfect-Reconstruction Filter Bank



The figure above shows a two-channel filter bank where H_0 and H_1 are analysis filters used in the decomposition process, and F_0 and F_1 are synthesis filters, used in the reconstruction process. The boxes with down and up arrows denote downsampling and upsampling operations, respectively [54]. The objective is to design these filters such that the overall filter bank has perfect reconstruction; the output is a delayed version of the input. It is clear that a two-channel perfect-reconstruction filter bank yields an invertible discrete dyadic wavelet transform.

Since these filters are used in image compression, only filters with symmetric and finite impulse responses (FIR) are considered. When applying wavelet and filter bank transforms to finite-length signals, symmetry of the filters becomes an important consideration. This is because one must provide special treatment at signal boundaries (e.g., edges of an image). A simple periodic wrap of the signal (circular convolution) will work but can lead to unpleasant artifacts if the signal intensities at opposite boundaries differ significantly. When the filters of the filter bank/wavelet transform are linear-phase (symmetric or antisymmetric), it has been shown [51, 2, 21, 3] that one may *symmetrically extend* the input signal (by reflecting it), and that the subband outputs will also be symmetric, leading to a critically sampled, perfectly invertible representation that behaves smoothly at signal boundaries.

Another advantage of symmetric filters is that they maintain the correct spatial and time positioning of events. In a wavelet representation, if the filters are linear-phase and odd-length (whole-sample-symmetric), then signal details remain centered on signal samples under iteration of the filtering and decimation operation. This is important both for frame-to-frame correlation schemes used in video processing, and for event localization in geophysical signal processing.

There are several design methods for two-channel filter banks and dyadic wavelets. They are based on spectral factorization [32, 50], lattice structure [55, 34], time-domain optimization [33], and quadratic-constrained least-squares (QCLS) [35]. The design method based on spectral factorization is outlined below.

Using z -transform analysis, one obtains the following conditions on the filters such that the overall two-channel filter bank is perfectly reconstructed [54, 53]:

$$\begin{cases} H_0(z)F_0(z) - H_0(-z)F_0(-z) = 2z^{-(2L+1)} \\ H_1(z) = F_0(-z), F_1(z) = -H_0(-z) \end{cases} \quad (6.1)$$

Defining $P_0(z) = H_0(z)F_0(z)$, the first condition above is equivalent to finding a polynomial $P_0(z)$ such that

$$P_0(z) - P_0(-z) = 2z^{-(2L+1)}. \quad (6.2)$$

A $P_0(z)$ that satisfies the above condition is a halfband filter with length $(4L + 3)$ [54, 53]. The design procedure is as follows:

1. Design a symmetric halfband filter $P_0(z)$ with length $(4L + 3)$.
2. Factorize $P_0(z)$ into $H_0(z)$ and $F_0(z)$ such that they are symmetric filters.
3. The highpass filters can be obtained from $H_1(z) = F_0(-z)$,
 $F_1(z) = -H_0(-z)$.

This design procedure yields a two-channel perfect reconstruction filter bank. Recall that one also needs the lowpass filters to have zeros at frequency π so that the resulting basis functions are smooth. This condition implies that the halfband filter $P_0(z)$ also has zeros at frequency π . Daubechies [12] discusses a design method to obtain a halfband filter $P_0(z)$ with the maximum number of zeros at π . For $P_0(z)$

with length $(4L + 3)$, the maximum number of zeros at π is $(2L + 2)$. The Daub 9/7 filter comes from a halfband filter with length 15 and with 8 zeros at π . Each lowpass filter in this case has 4 zeros at π .

6.2.2 Dyadic Wavelet Transform, Multiresolution Representation

A *wavelet* decomposition arises from iteration of the lowpass filtering and decimation steps of a multirate filter bank. For a dyadic wavelet decomposition, one iterates on the lowpass output only, whereas for a wavelet-packet decomposition one may iterate on any output [26, 53]. A finite number of iterations will lead to a discrete-time multiresolution analysis with lowpass frequency response $\prod_{k=1}^n H_0\left(\frac{\omega}{2^k}\right)$. If the lowpass filter h_0 satisfies the orthonormality constraint, $\sum_k h_0[k] = \frac{1}{\sqrt{2}}$, and has one vanishing moment ($\sum_k k h_0[k] = 0$), then the infinite product $\lim_{n \rightarrow \infty} \prod_{k=1}^n H_0\left(\frac{\omega}{2^k}\right)$ converges to a function $\phi(\omega)$, whose inverse Fourier transform is the continuous time function $\phi(t)$ called the scaling function [12, 26, 53]. The scaling function $\phi(t)$ is the solution to the dilation equation

$$\phi(t) = 2 \sum_k h_0[k] \phi(2t - k) , \quad (6.3)$$

and it is orthogonal to its integer translates. If the filter $h_0[n]$ is FIR, then $\phi(t)$ has compact support. The scaling function determines the wavelet $w(t)$ by means of the highpass filter h_1 :

$$w(t) = 2 \sum_k h_1[k] \phi(2t - k) . \quad (6.4)$$

The set of dilates and translates $\{w(2^k t - l)\}_{k,l \in \mathbf{Z}}$ forms a tight frame (and in most cases an orthonormal basis) for $L^2(\mathbf{R})$ [8, 23]. The functional relations Eqs. (6.3) and (6.4) introduce an entirely new set of relationships between discrete and continuous-time signal processing, unique to wavelet representations.

The span of integer translates of the scaling function $\phi(t)$ is the “lowpass” space V_0 , the set of *scale-limited signals* [17]. Any continuous-time function $f(t)$ in V_0 can be expanded as a linear combination $f(t) = \sum_n v_n^0 \phi(t - n)$. The superscript 0 denotes an expansion “at scale level 0.” $f(t)$ is completely described by the sequence $\{v_n^0\}$. Given such a sequence, its coarse approximation [component in V_1 , where V_1 is the signal space with basis function $\phi(2t - n)$] is computed with the lowpass filter of the wavelet filter bank:

$$v_n^1 = \left((v^0 * h_0) \downarrow 2 \right) [n] .$$

This is essentially implemented as lowpass filtering followed by downsampling in the two-channel filter bank structure. Analogously, the details [component in W_1 , where W_1 is the signal space with basis function $w(2t - n)$] are computed with the highpass

filter $h_1[n]$. Hence, if we take a discrete sequence v_n to be the coefficients of a signal $f(t)$ at some fixed scale, the discrete wavelet transform of v_n will decompose the underlying signal f into a coarse-scale component and detail at several intermediate scales, as follows:

$$V_0 = V_1 \oplus W_1 = [V_2 \oplus W_2] \oplus W_1 = [[V_3 \oplus W_3] \oplus W_2] \oplus W_1 = \dots = V_J \oplus \sum_{j=1}^J W_j.$$

In summary, the signal is represented in terms of its coarse approximation at scale J [with basis function $\phi(2^J t - n)$], and the J details [with basis functions $w(2^j t - n)$, $1 \leq j \leq J$]. This transform matches multiresolution models of human and computer vision [27] and has proven very effective for high-quality image compression. It also allows multiscale access to information, for applications such as image browsing and selective decoding of individual channels in a multicarrier system.

6.2.3 Wavelet Smoothness

As with any signal processing structure, one must consider the performance of the filters involved. In the case of the wavelet transform, one is concerned with the smoothness of the iterated lowpass filter. When using wavelets for lossy transform-based image coding, any quantization noise will appear in the decompressed image as linear combinations of the wavelet transform basis functions $\phi(t)$ and $w(t)$. If these basis functions (which are derived from the iterated discrete filter) are not smooth, then perceptually unacceptable artifacts will result. In fact, for all commonly used wavelets, the cascade converges fast enough so that the smoothness of the infinite limit is visually comparable to that of a six-level iterate [41]. Five- and six-level iterates are common in commercial implementations of wavelet-transform-based image compression [73, 10].

The smoothness of continuous-time wavelet systems has been the object of intensive study [12, 13, 15, 18, 64, 65]. Because the wavelet $w(t)$ is determined from the scaling function by means of the highpass filter taps Eq. (6.4), the smoothness of the scaling function (infinitely iterated lowpass filter) determines the smoothness of the overall wavelet system. In the two-band case, Daubechies' construction [12] imposed N vanishing wavelet moments $\int t^k w(t) dt = 0$, $0 \leq k \leq N - 1$ as a means of ensuring smoothness; this condition is equivalent to an N -th order zero at π for the lowpass filter: $\sum_n (-1)^n n^k h_0[n] = 0$, $0 \leq k \leq N - 1$. This condition was motivated by a theorem [11] stating that if an orthonormal system of dilates and translates $\{2^{j/2} w(2^j t - k)\}$ is made up of N times continuously differentiable functions, then the generating wavelet $w(t)$ must have N vanishing moments.

Vanishing moments are also associated with polynomial interpolation properties of the lowpass filter [52]. If a wavelet system has N vanishing moments, then polynomials of degree less than N may be represented as a linear combination of translates of the scaling function. In the setting of digital filter banks, this means that any locally polynomial component of a signal (of degree less than N) is preserved by the lowpass filter and zeroed out by the highpass filter — so long as the wavelet system has N

vanishing moments. These smoothness-under-iteration and polynomial approximation properties help explain why wavelet filters with vanishing moments perform so well in image compression.

6.3 Wavelet-Based Image Compression

There are two types of image compression: *lossless* and *lossy*. With lossless compression, the original image is recovered exactly after decompression. Unfortunately, with images of natural scenes it is rarely possible to obtain error-free compression at a rate beyond 2:1. Much higher compression ratios can be obtained if some error, which is usually difficult to perceive, is allowed between the decompressed image and the original image. This is lossy compression. In many cases, it is not necessary or even desirable that there be error-free reproduction of the original image. For example, if some noise is present, then the error due to that noise will usually be significantly reduced via some denoising method. In such a case, the small amount of error introduced by lossy compression may be acceptable. Lossy compression is also acceptable in fast transmission of still images over the Internet.

We concentrate on wavelet-based lossy compression of gray-level still images. When there are 256 levels of possible intensity for each pixel, then we shall call these images 8 bpp (bits per pixel) images. Images with 4096 gray-levels are referred to as 12 bpp. Some brief comments on color images are also given, and we also briefly describe some wavelet-based lossless compression methods.

6.3.1 Lossy Compression

We concentrate on the following methods of lossy compression: EZW (embedded zerotree wavelet) algorithm, SPIHT (set partitioning in hierarchical trees) algorithm, WDR (wavelet difference reduction) algorithm, and ASWDR (adaptively scanned wavelet difference reduction) algorithm. These are relatively recent algorithms which achieve some of the lowest errors per compression rate and highest perceptual quality yet reported. After describing these algorithms in detail, we shall list some of the other algorithms that are available.

Before we examine the algorithms listed above, we shall outline the basic steps that are common to all wavelet-based image compression algorithms. The five stages of compression and decompression are shown in Figs. 6.3 and 6.4. All of the steps shown in the compression diagram are invertible, hence lossless, except for the *quantize* step. Quantizing refers to a reduction of the precision of the floating point values of the wavelet transform, which are typically either 32- or 64-bit floating point numbers. To use less bits in the compressed transform — which is necessary if compression of 8 bpp or 12 bpp images is to be achieved — these transform values must be expressed with less bits for each value. This leads to rounding error. These approximate,

quantized, wavelet transforms will produce approximations to the images when an inverse transform is performed. Thus creating the error inherent in lossy compression.

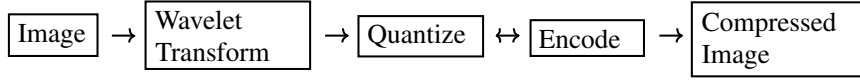


FIGURE 6.3
Compression of an image.

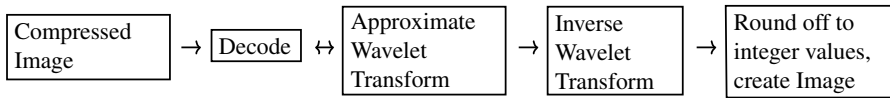


FIGURE 6.4
Decompression of an image.

The relationship between the quantize and encode steps, shown in Fig. 6.3, is the crucial aspect of wavelet transform compression. Each of the algorithms described below takes a different approach to this relationship.

The purpose served by the wavelet transform is that it produces a large number of values having zero, or near zero, magnitudes. For example, consider the image shown in Fig. 6.5(a), which is called “Lena.” Fig. 6.5(b), shows a 7-level Daub 9/7 wavelet transform of the “Lena” image. This transform has been thresholded, using a threshold of 8. That is, all values with magnitudes less than 8 have been set equal to 0; they appear as a uniformly gray background in the image in Fig. 6.5(b). These large areas of gray background indicate that there is a large number of zero values in the thresholded transform. If an inverse wavelet transform is performed on this thresholded transform, then the image in Fig. 6.5(c) results (after rounding to integer values between 0 and 255). It is difficult to detect any difference between the images in Figs. 6.5(a) and (c).

The image in Fig. 6.5(c) was produced using only the 32,498 nonzero values of the thresholded transform, instead of all 262,144 values of the original transform. This represents an 8:1 compression ratio. We are, of course, ignoring difficult problems such as how to transmit concisely the positions of the nonzero values in the thresholded transform, and how to encode these nonzero values with as few bits as possible. Solutions to these problems are described below, when the various compression algorithms are discussed.

Two commonly used measures for quantifying the error between images are *mean square error* (MSE) and *peak signal to noise ratio* (PSNR). The MSE between two images f and g is defined by

$$\text{MSE} = \frac{1}{N} \sum_{j,k} (f[j, k] - g[j, k])^2 \quad (6.5)$$

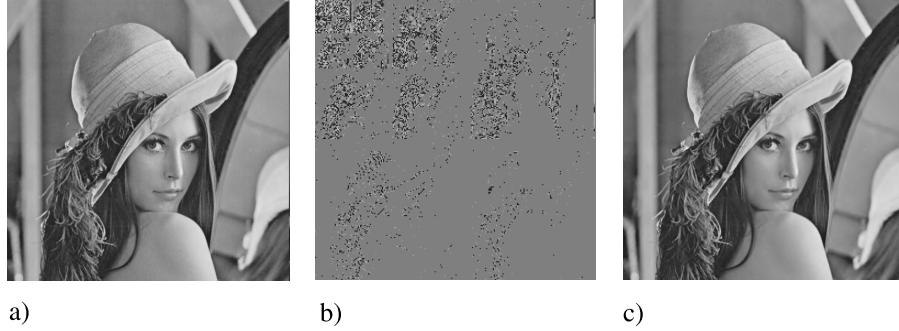


FIGURE 6.5

(a) “Lena” image, 8 bpp. (b) Wavelet transform of image, threshold = 8. (c) Inverse of thresholded wavelet transform, PSNR = 39.14 dB. Reproduced by Special Permission of *Playboy* magazine. Copyright ©1972, 2000 by Playboy.

where the sum over j, k denotes the sum over all pixels in the images, and N is the number of pixels in each image. For the images in Figs. 6.5(a) and (c), the MSE is 7.921. The PSNR between two (8 bpp) images is, in decibels,

$$\text{PSNR} = 10 \log_{10} \left(\frac{255^2}{\text{MSE}} \right). \quad (6.6)$$

PSNR tends to be cited more often since it is a logarithmic measure, and our brains seem to respond logarithmically to intensity. Increasing PSNR represents increasing fidelity of compression. For the images in Figs. 6.5(a) and (c), the PSNR is 39.14 dB. Generally, when the PSNR is 40 dB or larger, then the two images are virtually indistinguishable by human observers. In this case, we can see that 8:1 compression should yield an image almost identical to the original. The methods described below do in fact produce such results with even greater PSNR than we have just achieved with our crude approach.

Before we begin our treatment of various “state of the art” algorithms, it may be helpful to briefly outline a baseline compression algorithm of the kind described in Davis and Nosratinia [14] and Mallat [26]. This algorithm has two main parts.

First, the positions of the significant transform values — the ones having larger magnitudes than the threshold T — are determined by scanning through the transform as shown in Fig. 6.6. The positions of the significant values are then encoded using a runlength method. To be precise, it is necessary to store the values of the significance map:

$$s(m) = \begin{cases} 0 & \text{if } |w(m)| < T \\ 1 & \text{if } |w(m)| \geq T, \end{cases} \quad (6.7)$$

where m is the scanning index, and $w(m)$ is the wavelet transform value at index m . From Fig. 6.5(b) we can see that there will be long runs of $s(m) = 0$. If the scan

1	2	5	8	17	24	25	32
3	4	6	7	18	23	26	31
9	10	13	14	19	22	27	30
12	11	15	16	20	21	28	29
33	34	35	36	49	50	54	55
40	39	38	37	51	53	56	61
41	42	43	44	52	57	60	62
48	47	46	45	58	59	63	64

(a) 2-level

1	2	5	8	17	24	25	32
3	4	6	7	18	23	26	31
9	10	13	14	19	22	27	30
12	11	15	16	20	21	28	29
33	34	35	36	49	50	54	55
40	39	38	37	51	53	56	61
41	42	43	44	52	57	60	62
48	47	46	45	58	59	63	64

(b) 3-level

FIGURE 6.6

Scanning for wavelet transforms: zigzag through all-lowpass subband, column scan through vertical subbands, row scan through horizontal subbands, zigzag through diagonal subbands. (a) and (b): Order of scanned elements for 2-level and 3-level transforms of 8 by 8 image.

order illustrated in Fig. 6.6 is used, then there will also be long runs of $s(m) = 1$. The positions of significant values can then be concisely encoded by recording sequences of 6 bits according to the following pattern:

$$\begin{aligned}
 0\,abcde &: \text{run of } 0 \text{ of length } (abcde)_2 \\
 1\,abcde &: \text{run of } 1 \text{ of length } (abcde)_2 .
 \end{aligned}$$

A lossless compression, such as Huffman or arithmetic compression, of these data is also performed for a further reduction in bits.

Second, the significant values of the transform are encoded. This can be done by dividing the range of transform values into subintervals (*bins*) and rounding each transform value into the midpoint of the bin in which it lies. Fig. 6.7 shows the histogram of the frequencies of significant transform values lying in 512 bins for the 7-level Daub 9/7 transform of “Lena” shown in Fig. 6.5(b). The extremely rapid drop in the frequencies of occurrence of higher transform magnitudes implies that the very low magnitude values, which occur much more frequently, should be encoded using shorter length bit sequences. This is typically done with either Huffman encoding or arithmetic coding. If arithmetic coding is used, then the average number of bits needed to encode each significant value in this case is about 1 bit.

We have only briefly sketched the steps in this baseline compression algorithm. More details can be found in Davis and Nosratinia [14] and Mallat [26].

Our purpose in discussing the baseline compression algorithm is to introduce some basic concepts, such as scan order and thresholding, which are needed for our examination of the algorithms to follow. The baseline algorithm was one of the first to be

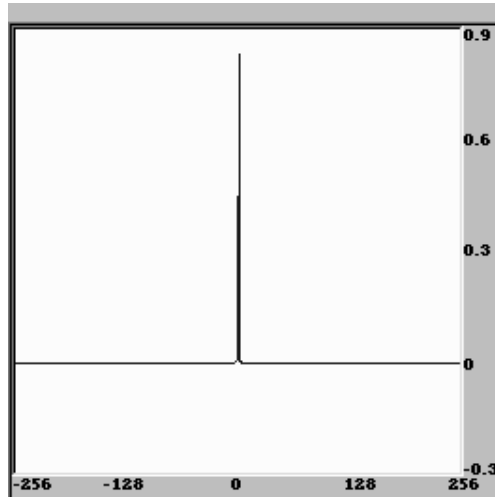


FIGURE 6.7
Histogram for 512 bins for thresholded transform of “Lena.”

proposed using wavelet methods [1]. It suffers from some defects which later algorithms have remedied. For instance, with the baseline algorithm it is very difficult, if not impossible, to specify in advance the exact compression rate or the exact error to be achieved. This is a serious defect. Another problem with the baseline method is that it does not allow for progressive transmission. In other words, it is not possible to send successive data packets (over the Internet, for instance) which produce successively increasing resolution for the received image. Progressive transmission is vital for applications that include some level of interaction with the receiver.

Let us now turn to these improved wavelet image compression algorithms. The algorithms to be discussed are the EZW, SPIHT, WDR, and ASWDR algorithms.

6.3.2 EZW Algorithm

The EZW algorithm was one of the first algorithms to show the full power of wavelet-based image compression. It was introduced in the groundbreaking paper of Shapiro [48]. We shall describe EZW in some detail because a solid understanding of it will make it much easier to comprehend the other algorithms we shall be discussing. These other algorithms build upon the fundamental concepts that were first introduced with EZW.

Our discussion of EZW will be focused on the fundamental ideas underlying it. We will not use it to compress any images because it has been superseded by a far superior algorithm, SPIHT. Since SPIHT is only a highly refined version of EZW, it makes sense to first describe EZW.

EZW stands for *embedded zerotree wavelet*. An embedded coding is a process of encoding the transform magnitudes that allows for progressive transmission of

the compressed image. Zerotrees allow for a concise encoding of the positions of significant values that result during the embedded coding process. We shall first discuss embedded coding, and then examine the notion of zerotrees.

The embedding process used by EZW is called *bit-plane encoding*. It consists of the following five-step process:

Bit-plane encoding —

Step 1: Initialize. Choose initial threshold, $T = T_0$, such that *all* transform values satisfy $|w(m)| < T_0$ and at least one transform value satisfies $|w(m)| \geq T_0/2$.

Step 2: Update threshold. Let $T_k = T_{k-1}/2$.

Step 3: Significance pass. Scan through insignificant values using baseline algorithm scan order. Test each value $w(m)$ as follows:

```

If  $|w(m)| \geq T_k$ , then
    Output sign of  $w(m)$ 
    Set  $w_Q(m) = T_k$ 
Else if  $|w(m)| < T_k$  then
    Let  $w_Q(m)$  retain its initial value of 0 .

```

Step 4: Refinement pass. Scan through significant values found with higher threshold values T_j , for $j < k$ (if $k = 1$ skip this step). For each significant value $w(m)$, do the following:

```

If  $|w(m)| \in [w_Q(m), w_Q(m) + T_k)$ , then
    Output bit 0
Else if  $|w(m)| \in [w_Q(m) + T_k, w_Q(m) + 2T_k)$ , then
    Output bit 1
    Replace value of  $w_Q(m)$  by  $w_Q(m) + T_k$  .

```

Step 5: Loop. Repeat steps 2 through 4.

This bit-plane encoding procedure can be continued for as long as necessary to obtain quantized transform magnitudes $w_Q(m)$ which are as close as desired to the transform magnitudes $|w(m)|$. During decoding, the signs and the bits output by this method can be used to construct an approximate wavelet transform to any desired degree of accuracy. If instead, a given compression ratio is desired, then it can be achieved by stopping the bit-plane encoding as soon as a given number of bits (a *bit budget*) is exhausted. In either case, the execution of the bit-plane encoding procedure can terminate at any point (not just at the end of one of the loops).

As a simple example of bit-plane encoding, suppose that we just have two transform values $w(1) = -9.5$ and $w(2) = 42$. For an initial threshold, we set $T_0 = 64$. During the first loop, when $T_1 = 32$, the output is the sign of $w(2)$, and the quantized transform magnitudes are $w_Q(1) = 0$ and $w_Q(2) = 32$. For the second loop, $T_2 = 16$, and there is no output from the significance pass. The refinement pass produces the bit 0 because $w(2) \in [32, 32 + 16)$. The quantized transform magnitudes are $w_Q(1) = 0$ and $w_Q(2) = 32$. During the third loop, when $T_3 = 8$, the significance pass outputs the

sign of $w(1)$. The refinement pass outputs the bit 1 because $w(2) \in [32+8, 32+16)$. The quantized transform magnitudes are $w_Q(1) = 8$ and $w_Q(2) = 40$.

It is not hard to see that *after n loops, the maximum error between the transform values and their quantized counterparts is less than $T_0/2^n$* . It follows that we can reduce the error to as small a value as we wish by performing a large enough number of loops. For instance, in the simple example just described, with seven loops the error is reduced to zero. The output from these seven loops, arranged to correspond to $w(1)$ and $w(2)$, is

$$\begin{array}{rcccccc} w(1): & & & - & 0 & 0 & 1 & 1 \\ w(2): & + & 0 & 1 & 0 & 1 & 0 & 0. \end{array}$$

Notice that $w(2)$ requires seven symbols, but $w(1)$ requires only five.

Bit-plane encoding consists simply of computing binary expansions — using T_0 as unit — for the transform values and recording *in magnitude order* only the significant bits in these expansions. Because the first significant bit is always 1, it is *not* encoded. Instead, the sign of the transform value is encoded first. This coherent ordering of encoding, with highest magnitude bits encoded first, is what allows for progressive transmission.

Wavelet transforms are particularly well-adapted for bit-plane encoding¹ because wavelet transforms of images of natural scenes often have relatively few high-magnitude values, which are mostly found in the highest level subbands. These high-magnitude values are first coarsely approximated during the initial loops of the bit-plane encoding, thereby producing a low-resolution, but often recognizable, version of the image. Subsequent loops encode lower magnitude values and refine the high magnitude values, adding further details to the image and refining existing details. Thus, progressive transmission is possible, and encoding/decoding can cease once a given bit budget is exhausted or a given error target is achieved.

Now that we have described the embedded coding of wavelet transform values, we will describe the zerotree method by which EZW transmits the positions of significant transform values. The zerotree method gives an implicit, very compact, description of the location of significant values by creating a highly compressed description of the location of insignificant values. For many images of natural scenes, such as the “Lena” image for example, insignificant values at a given threshold T are organized in zerotrees.

To define a zerotree we first define a *quadtree* — a tree of locations in the wavelet transform with a root $[i, j]$ and its *children* located at $[2i, 2j]$, $[2i+1, 2j]$, $[2i, 2j+1]$, and $[2i+1, 2j+1]$, and each of their children, and so on. These *descendants* of the root reach all the way back to the 1st level of the wavelet transform. For example, Fig. 6.8(a) shows two quadtrees (enclosed in dashed boxes). One quadtree has root at index 12 and children at indices $\{41, 42, 47, 48\}$. This quadtree has two levels. We denote it by $\{12 | 41, 42, 47, 48\}$. The other quadtree, which has three levels, has its

¹ Although other transforms, such as the block discrete cosine transform, can also be bit-plane encoded.

root at index 4, the children of this root at indices {13, 14, 15, 16}, and their children at indices {49, 50, ..., 64}. It is denoted by $\{4 | 13, \dots, 16 | 49, \dots, 64\}$.

1	2	5	8	17	24	25	32
3	4	6	7	18	23	26	31
9	10	13	14	19	22	27	30
12	11	15	16	20	21	28	29
33	34	35	36	49	50	54	55
40	39	38	37	51	53	56	61
41	42	43	44	52	57	60	62
48	47	46	45	58	59	63	64

(a) Scan order, with two quadtrees

63	-34	49	10	5	18	-12	7
-31	23	14	-13	3	4	6	-1
-25	-7	-14	8	5	-7	3	9
-9	14	3	-12	4	-2	3	2
-5	9	-1	47	4	6	-2	2
3	0	-3	2	3	-2	0	4
2	-3	6	-4	3	6	3	6
5	11	5	6	0	3	-4	4

(b) Wavelet transform

+	-	+	R	I	I	•	•
I	R	R	R	I	I	•	•
R	I	•	•	•	•	•	•
R	R	•	•	•	•	•	•
•	•	I	+	•	•	•	•
•	•	I	I	•	•	•	•
•	•	•	•	•	•	•	•
•	•	•	•	•	•	•	•

(c) Threshold = 32

+	-	+	R	I	+	•	•
-	+	R	R	I	I	•	•
-	R	R	R	•	•	•	•
R	R	R	R	•	•	•	•
I	I	•	+	•	•	•	•
I	I	•	•	•	•	•	•
•	•	•	•	•	•	•	•
•	•	•	•	•	•	•	•

(d) Threshold = 16

FIGURE 6.8

First two stages of EZW. (a) 3-level scan order. (b) 3-level wavelet transform. (c) Stage 1, threshold = 32. (d) Stage 2, threshold = 16.

Now that we have defined a quadtree, we can give a simple definition of a zerotree. A *zerotree* is a quadtree which, for a given threshold T , has insignificant wavelet transform values at each of its locations. For example, if the threshold is $T = 32$,

then each of the quadtrees shown in Fig. 6.8(a) is a zerotree for the wavelet transform in Fig. 6.8(b). But if the threshold is $T = 16$, then $\{12 | 41, 42, 47, 48\}$ remains a zerotree, but $\{4 | 13, \dots, 16 | 49, \dots, 64\}$ is no longer a zerotree because its root value is no longer insignificant.

Zerotrees can provide very compact descriptions of the locations of insignificant values because it is only necessary to encode one symbol, such as R , to mark the root location. The decoder can infer that all other locations in the zerotree have insignificant values, so their locations are not encoded. For the threshold $T = 32$, in the example just discussed, two R symbols are enough to specify all 26 locations in the two zerotrees.

Zerotrees can be useful only if they occur frequently. Fortunately, with wavelet transforms of natural scenes, the multiresolution structure of the wavelet transform does produce many zerotrees (especially at higher thresholds). For example, consider the images shown in Fig. 6.9. Fig. 6.9(a) shows the second all-lowpass subband of a Daub 9/7 transform of the “Lena” image. Image 6.9(b), on its right, is the third vertical subband produced from this all-lowpass subband, with a threshold of 16. Notice that there are large patches of gray pixels in this image. These represent insignificant transform values for the threshold of 16 which correspond to regions of nearly constant, or nearly linearly graded, intensities in the image in 6.9(a). Such intensities are nearly orthogonal to the analyzing Daub 9/7 wavelets. Zerotrees arise for the threshold of 16 because in image 6.9(c) — the second all-lowpass subband — there are similar regions of constant or linearly graded intensities. In fact, it was precisely these regions that were smoothed and downsampled to create the corresponding regions in image 6.9(a). These regions in image 6.9(c) produce insignificant values *in the same relative locations* (the child locations) in the second vertical subband shown in image 6.9(d).

Likewise, there are uniformly gray regions in the same relative locations in the first vertical subband [see Fig. 6.9(f)]. Because the second vertical subband in Fig. 6.9(d) is magnified by a factor of two in each dimension, and the third vertical subband in Fig. 6.9(b) is magnified by a factor of four in each dimension, it follows that the common regions of gray background shown in these three vertical subbands are all zerotrees. Similar images could be shown for horizontal and diagonal subbands, and they would also indicate a large number of zerotrees.

The “Lena” image is typical of many images of natural scenes, and the above discussion gives some background for understanding how zerotrees arise in wavelet transforms. A more rigorous, statistical discussion can be found in Shapiro [48].

Now that we have laid the foundations of zerotree encoding, we can complete our discussion of the EZW algorithm. The EZW algorithm consists simply of replacing the significance pass in the Bit-plane encoding procedure with the following step:

EZW Step 3: Significance pass. Scan through insignificant values using baseline

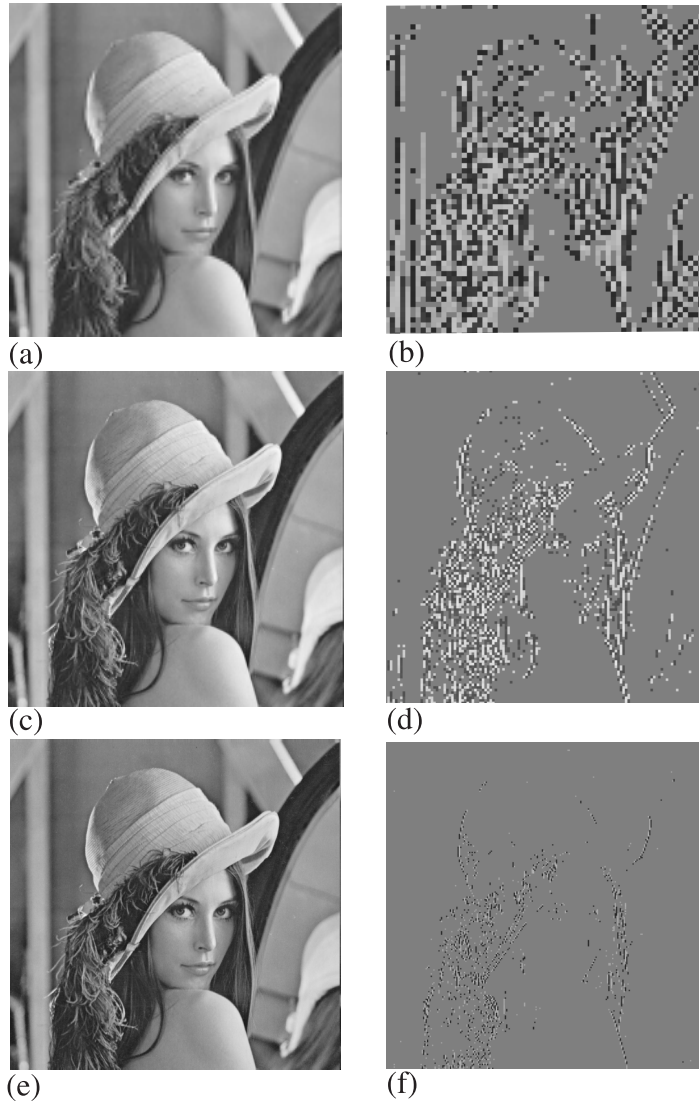


FIGURE 6.9

(a) Second all-lowpass subband. (b) Third vertical subband. (c) First all-lowpass subband. (d) Second vertical subband. (e) Original "Lena." (f) First vertical subband. Reproduced by Special Permission of *Playboy* magazine. Copyright ©1972, 2000 by Playboy.

algorithm scan order. Test each value $w(m)$ as follows:

```

If  $|w(m)| \geq T_k$ , then
    Output the sign of  $w(m)$ 
    Set  $w_Q(m) = T_k$ 
Else if  $|w(m)| < T_k$  then
    Let  $w_Q(m)$  remain equal to 0
    If  $m$  is at 1st level, then
        Output  $I$ 
    Else
        Search through quadtree having root  $m$ 
        If this quadtree is a zerotree, then
            Output  $R$ 
        Else
            Output  $I$  .

```

During a search through a quadtree, values that were found to be significant at higher thresholds are treated as zeros. All descendants of a root of a zerotree are skipped in the rest of the scanning at this threshold.

As an example of the EZW method, consider the wavelet transform shown in Fig. 6.8(b), which will be scanned through using the scan order shown in Fig. 6.8(a). Suppose that the initial threshold is $T_0 = 64$. In the first loop, the threshold is $T_1 = 32$. The results of the first significance pass are shown in Fig. 6.8(c). The coder output after this first loop would be

$$+ - I R + R R R R I R R I I I I I + I I \quad (6.8)$$

corresponding to a quantized transform having only two values: ± 32 — $+32$ at each location marked by a plus sign in Fig. 6.8(c), -32 at each location marked by a minus sign, and 0 at all other locations. In the second loop, with threshold $T_2 = 16$, the results of the significance pass are indicated in Fig. 6.8(d). Notice, in particular, that the symbol R is at the position 10 in the scan order because the plus sign which lies at a child location is from the previous loop, so it is treated as zero. Hence, position 10 is at the root of a zerotree. There is also a refinement pass done in this second loop. The output from this second loop is then

$$- + R R R - R R R R R R I I I + I I I I 1 0 1 0 \quad (6.9)$$

with corresponding quantized wavelet transform shown in Fig. 6.10(a). The MSE between this quantized transform and the original transform is 48.6875. This is a 78% reduction in error from the start of the method (when the quantized transform has all zero values).

A couple of final remarks are in order concerning the EZW method. First, it should be clear from the discussion above that the decoder, whose structure is outlined in

Fig. 6.4 above, can reverse each of the steps of the coder and produce the quantized wavelet transform. It is standard practice for the decoder to then round the quantized values to the midpoints of the intervals that they were last found to belong to during the encoding process (i.e., add half of the last threshold used to their magnitudes). This generally reduces MSE. For instance, in the example just considered, if this rounding is done to the quantized transform in Fig. 6.10(a), then the result is shown in Fig. 6.10(b). The MSE is then 39.6875, a reduction of more than 18%. A good discussion of the theoretical justification for this rounding technique can be found in Mallat [26]. *This rounding method will be employed by all of the other algorithms that we shall discuss.*

48	-32	48	0	0	16	0	0	56	-40	56	0	0	24	0	0
-16	16	0	0	0	0	0	0	-24	24	0	0	0	0	0	0
-16	0	0	0	0	0	0	0	-24	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	32	0	0	0	0	0	0	0	40	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

(a) (b)

FIGURE 6.10

(a) Quantization at end of second stage, MSE = 48.6875. (b) After rounding to midpoints, MSE = 39.6875, reduction by more than 18%.

Second, since we live in a digital world, it is usually necessary to transmit just bits. A simple encoding of the symbols of the EZW algorithm into bits would be to use a code such as $+$ = 01, $-$ = 00, R = 10, and I = 11. Since the decoder can always infer precisely when the encoding of these symbols ends (the significance pass is complete), the encoding of refinement bits can simply be as single bits 0 and 1. This form of encoding is the fastest to perform, but it does not achieve the greatest compression. In Shapiro [48], a lossless form of arithmetic coding was recommended in order to further compress the bit stream from the encoder.

6.3.3 SPIHT Algorithm

The SPIHT algorithm is a highly refined version of the EZW algorithm. It was introduced in Said and Pearlman [44, 45]. Some of the best results — highest PSNR values for given compression ratios — for a wide variety of images have been ob-

tained with SPIHT. Consequently, it is probably the most widely used wavelet-based algorithm for image compression, providing a basic standard of comparison for all subsequent algorithms.

SPIHT stands for set partitioning in hierarchical trees. The term *hierarchical trees* refers to the quadrees that we defined in our discussion of EZW. *Set partitioning* refers to the way these quadrees partition the wavelet transform values at a given threshold. By a careful analysis of this partitioning of transform values, Said and Pearlman were able to greatly improve the EZW algorithm, significantly increasing its compressive power.

Our discussion of SPIHT will consist of three parts. First, we describe a modified version of the algorithm introduced in Said and Pearlman [44]. We refer to it as the *spatial-orientation tree wavelet* (STW) algorithm. STW is essentially the SPIHT algorithm; the only difference is that SPIHT is slightly more careful in its organization of coding output. Second, we describe the SPIHT algorithm. It is easier to explain SPIHT using the concepts underlying STW. Third, we see how well SPIHT compresses images.

The only difference between STW and EZW is that STW uses a different approach to encoding the zerotree information. STW uses a *state transition model*. From one threshold to the next, the locations of transform values undergo state transitions. This model allows STW to reduce the number of bits needed for encoding. Instead of code for the symbols R and I output by EZW to mark locations, the STW algorithm uses states I_R , I_V , S_R , and S_V and outputs code for state-transitions such as $I_R \rightarrow I_V$, $S_R \rightarrow S_V$, etc. To define the states involved, some preliminary definitions are needed.

For a given index m in the baseline scan order, define the set $D(m)$ as follows. If m is either at the first level or at the all-lowpass level, then $D(m)$ is the empty set \emptyset . Otherwise, if m is at the j th level for $j > 1$, then

$$D(m) = \{\text{Descendents of index } m \text{ in quadtree with root } m\}.$$

The significance function \mathcal{S} is defined by

$$\mathcal{S}(m) = \begin{cases} \max_{n \in D(m)} |w(n)|, & \text{if } D(m) \neq \emptyset \\ \infty, & \text{if } D(m) = \emptyset. \end{cases}$$

With these preliminary definitions in hand, we can now define the states. For a given threshold T , the states I_R , I_V , S_R , and S_V are defined by

$$m \in I_R \quad \text{if and only if} \quad |w(m)| < T, \mathcal{S}(m) < T \quad (6.10)$$

$$m \in I_V \quad \text{if and only if} \quad |w(m)| < T, \mathcal{S}(m) \geq T \quad (6.11)$$

$$m \in S_R \quad \text{if and only if} \quad |w(m)| \geq T, \mathcal{S}(m) < T \quad (6.12)$$

$$m \in S_V \quad \text{if and only if} \quad |w(m)| \geq T, \mathcal{S}(m) \geq T. \quad (6.13)$$

Fig. 6.11 shows the state transition diagram for these states when a threshold is decreased from T to $T' < T$. Note that once a location m arrives in state S_V , it will remain in that state. Furthermore, there are only two transitions from each of the

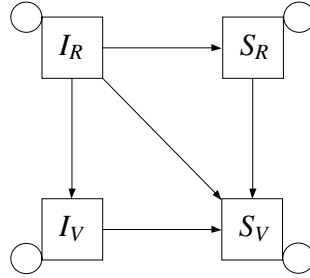


FIGURE 6.11
State transition diagram for STW.

Table 6.1 Code for State
Transitions, • Indicates that
 $S_V \rightarrow S_V$ Transition is Certain
(Hence no Encoding Needed)

Old\New	I_R	I_V	S_R	S_V
I_R	00	01	10	11
I_V		0		1
S_R			0	1
S_V				•

states I_V and S_R , so those transitions can be coded with one bit each. A simple binary coding for these state transitions is shown in Table 6.1.

Now that we have laid the groundwork for the STW algorithm, we can give its full description.

STW encoding —

Step 1: Initialize. Choose initial threshold, $T = T_0$, such that *all* transform values satisfy $|w(m)| < T_0$ and at least one transform value satisfies $|w(m)| \geq T_0/2$. Assign all indices for the L th level, where L is the number of levels in the wavelet transform, to the *dominant list* (this includes all locations in the all-lowpass subband as well as the horizontal, vertical, and diagonal subbands at the L th level). Set the *refinement list* of indices equal to the empty set.

Step 2: Update threshold. Let $T_k = T_{k-1}/2$.

Step 3: Dominant pass. Use the following procedure to scan through indices in the

dominant list (which can change as the procedure is executed).

```

Do
  Get next index  $m$  in dominant list
  Save old state  $S_{old} = S(m, T_{k-1})$ 
  Find new state  $S_{new} = S(m, T_k)$  using Eqs. (6.10) -- (6.13)
  Output code for state transition  $S_{old} \rightarrow S_{new}$ 
  If  $S_{new} \neq S_{old}$  then do the following
    If  $S_{old} \neq S_R$  and  $S_{new} \neq I_V$  then
      Append index  $m$  to refinement list
      Output sign of  $w(m)$  and set  $w_Q(m) = T_k$ 
    If  $S_{old} \neq I_V$  and  $S_{new} \neq S_R$  then
      Append child indices of  $m$  to dominant list
    If  $S_{new} = S_V$  then
      Remove index  $m$  from dominant list
  Loop until end of dominant list

```

Step 4: *Refinement pass.* Scan through indices m in the refinement list found with higher threshold values T_j , for $j < k$ (if $k = 1$ skip this step). For each value $w(m)$, do the following:

```

If  $|w(m)| \in [w_Q(m), w_Q(m) + T_k)$ , then
  Output bit 0
Else if  $|w(m)| \in [w_Q(m) + T_k, w_Q(m) + 2T_k)$ , then
  Output bit 1
  Replace value of  $w_Q(m)$  by  $w_Q(m) + T_k$  .

```

Step 5: *Loop.* Repeat steps 2 through 4.

To see how STW works — and how it improves the EZW method — it helps to reconsider the example shown in Fig. 6.8. In Fig. 6.12, we show STW states for the wavelet transform in Fig. 6.8(b) using the same two thresholds we used previously with EZW. It is important to compare the three quadrees enclosed in the dashed boxes in Fig. 6.12 with the corresponding quadrees in Figs. 6.8(c) and (d). There is a large savings in coding output for STW represented by these quadrees. The EZW symbols for these three quadrees are $+ I I I I$, $- I I I I$, and $+ R R R R$. For STW, however, they are described by the symbols $+ S_R$, $- S_R$, and $+ S_R$, which is a substantial reduction in the information that STW needs to encode.

There is not much difference between STW and SPIHT. The one thing that SPIHT does differently is to carefully organize the output of bits in the encoding of state transitions in Table 6.1, *so that only one bit is output at a time*. For instance, for the transition $I_R \rightarrow S_R$, which is coded as 1 0 in Table 6.1, SPIHT outputs a 1 first and then (after further processing) outputs a 0. Even if the bit budget is exhausted before the second bit can be output, the first bit of 1 indicates that there is a new significant value.

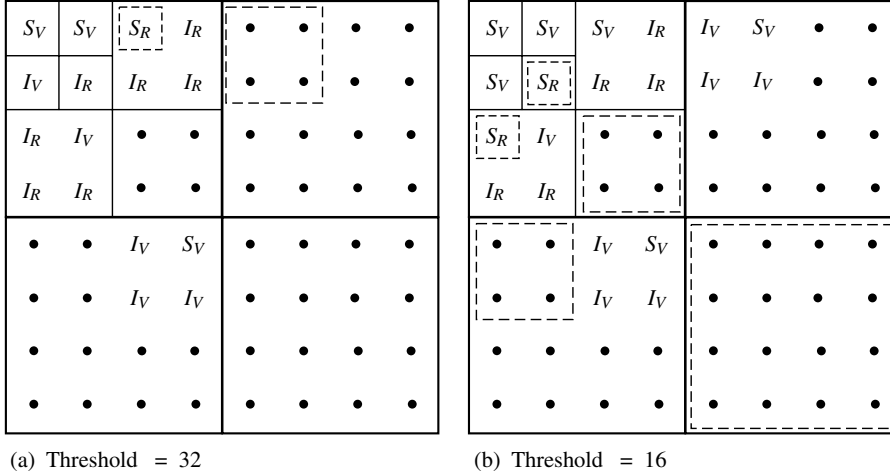


FIGURE 6.12
First two stages of STW for wavelet transform in Fig. 6.8.

The SPIHT encoding process, as described in Said and Pearlman [45], is phrased in terms of pixel locations $[i, j]$ rather than indices m in a scan order. To avoid introducing new notation, and to highlight the connections between SPIHT and the other algorithms, EZW and STW, we rephrase the description of SPIHT from Said and Pearlman [45] in terms of scanning indices. We also slightly modify their notation in the interests of clarity.

First, we need some preliminary definitions. For a given set \mathcal{I} of indices in the baseline scan order, the significance $\mathcal{S}_T[\mathcal{I}]$ of \mathcal{I} relative to a threshold T is defined by

$$\mathcal{S}_T[\mathcal{I}] = \begin{cases} 1, & \text{if } \max_{n \in \mathcal{I}} |w(n)| \geq T \\ 0, & \text{if } \max_{n \in \mathcal{I}} |w(n)| < T. \end{cases} \quad (6.14)$$

It is important to note that, for the initial threshold T_0 , we have $\mathcal{S}_{T_0}[\mathcal{I}] = 0$ for all sets of indices. If \mathcal{I} is a set containing just a single index m , then for convenience we write $\mathcal{S}_T[m]$ instead of $\mathcal{S}_T[\{m\}]$.

For a succinct presentation of the method, we need the following definitions of sets of indices:

$$\begin{aligned} \mathcal{D}(m) &= \{\text{Descendent indices of the index } m\} \\ \mathcal{C}(m) &= \{\text{Child indices of the index } m\} \\ \mathcal{G}(m) &= \mathcal{D}(m) - \mathcal{C}(m) \\ &= \{\text{Grandchildren of } m, \text{ i.e., descendants which are not children}\}. \end{aligned}$$

In addition, the set H consists of indices for the L th level, where L is the number of levels in the wavelet transform (this includes all locations in the all-lowpass subband as well as the horizontal, vertical, and diagonal subbands at the L th level). It is important to remember that the indices in the all-lowpass subband have no descendants. If m marks a location in the all-lowpass subband, then $D(m) = \emptyset$.

SPIHT keeps track of the states of sets of indices by means of three lists. They are the *list of insignificant sets* (LIS), the *list of insignificant pixels* (LIP), and the *list of significant pixels* (LSP). For each list a set is identified by a single index, in the LIP and LSP these indices represent the singleton sets $\{m\}$ where m is the identifying index. An index m is called either significant or insignificant, depending on whether the transform value $w(m)$ is significant or insignificant with respect to a given threshold. For the LIS, the index m denotes either $D(m)$ or $G(m)$. In the former case, the index m is said to be of type D and, in the latter case, of type G.

The following is the pseudocode for the SPIHT algorithm. For simplicity, we write the significance function \mathcal{S}_{T_k} as \mathcal{S}_k .

SPIHT encoding —

Step 1: Initialize. Choose initial threshold T_0 such that *all* transform values satisfy $|w(m)| < T_0$ and at least one value satisfies $|w(m)| \geq T_0/2$. Set LIP equal to H , set LSP equal to \emptyset , and set LIS equal to all the indices in H that have descendants (assigning them all type D).

Step 2: Update threshold. Let $T_k = T_{k-1}/2$.

Step 3: Sorting pass. Proceed as follows:

```

For each  $m$  in LIP do:
  Output  $\mathcal{S}_k[m]$ 
  If  $\mathcal{S}_k[m] = 1$  then
    Move  $m$  to end of LSP
    Output sign of  $w(m)$ ; set  $w_Q(m) = T_k$ 
Continue until end of LIP
For each  $m$  in LIS do:
  If  $m$  is of type D then
    Output  $\mathcal{S}_k[D(m)]$ 
    If  $\mathcal{S}_k[D(m)] = 1$  then

```

```

For each  $n \in C(m)$  do:
  Output  $S_k[n]$ 
  If  $S_k[n] = 1$  then
    Append  $n$  to LSP
    Output sign of  $w(n)$ ; set  $w_Q(n) = T_k$ 
  Else If  $S_k[n] = 0$  then
    Append  $n$  to LIP
If  $G(m) \neq \emptyset$  then
  Move  $m$  to end of LIS as type G
Else
  Remove  $m$  from LIS
Else If  $m$  is of type G then
  Output  $S_k[G(m)]$ 
  If  $S_k[G(m)] = 1$  then
    Append  $C(m)$  to LIS, all type D indices
    Remove  $m$  from LIS
Continue until end of LIS

```

Notice that the set LIS can undergo many changes during this procedure, it typically does not remain fixed throughout.

Step 4: Refinement pass. Scan through indices m in LSP found with higher threshold values T_j , for $j < k$ (if $k = 1$ skip this step). For each value $w(m)$, do the following:

```

If  $|w(m)| \in [w_Q(m), w_Q(m) + T_k)$ , then
  Output bit 0
Else if  $|w(m)| \in [w_Q(m) + T_k, w_Q(m) + 2T_k)$ , then
  Output bit 1
Replace value of  $w_Q(m)$  by  $w_Q(m) + T_k$  .

```

Step 5: Loop. Repeat steps 2 through 4.

It helps to carry out this procedure on the wavelet transform shown in Fig. 6.8. Then one can see that SPIHT simply performs STW with the binary code for the states in Table 6.1 being output one bit at a time.

Now comes the payoff. We shall see how well SPIHT performs in compressing images. To do these compressions we used the public domain SPIHT programs that can be downloaded from the Internet [46]. In Fig. 6.13 we show several SPIHT compressions of the “Lena” image. The original “Lena” image is shown in Fig. 6.13(f). Five SPIHT compressions are shown with compression ratios of 128:1, 64:1, 32:1, 16:1, and 8:1.

Several things are worth noting about these compressed images. First, they were all produced from one file containing the 1 bpp compression of the “Lena image.”

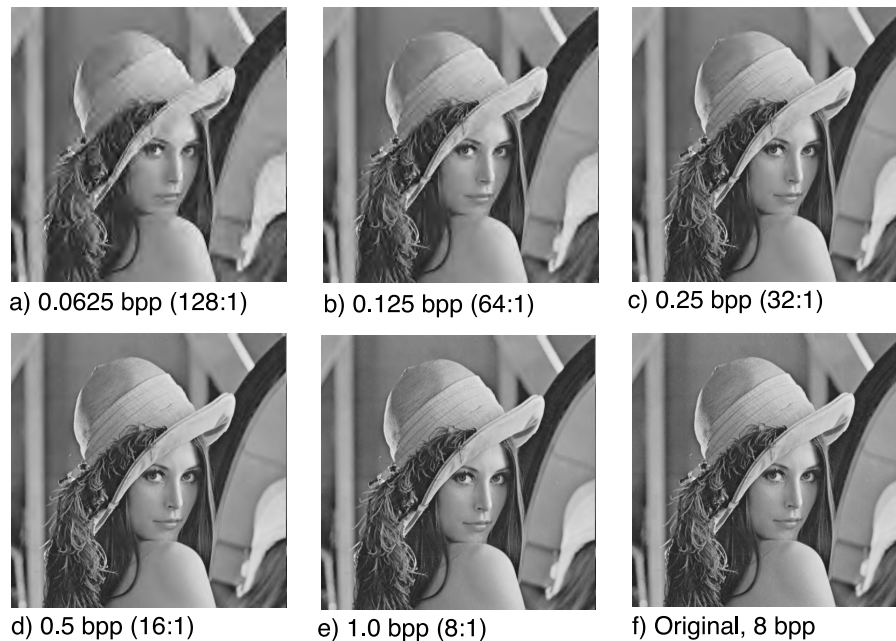


FIGURE 6.13

SPIHT compressions of “Lena” image. PSNR values: (a) 27.96 dB. (b) 30.85 dB. (c) 33.93 dB. (d) 37.09 dB. (e) 40.32 dB. Reproduced by Special Permission of *Playboy* magazine. Copyright ©1972, 2000 by Playboy.

By specifying a bit budget, a certain bpp value up to 1, the SPIHT decompression program will stop decoding the 1 bpp compressed file once the bit budget is exhausted. This illustrates the embedded nature of SPIHT.

Second, the rapid convergence of the compressed images to the original is nothing short of astonishing. Even the 64: 1 compression in Fig. 6.13(b) is almost indistinguishable from the original. A close examination of the two images is needed in order to see some differences, e.g., the blurring of details in the top of Lena’s hat. The image in (b) would be quite acceptable for some applications, such as the first image in a sequence of video telephone images or as a thumbnail display within a large archive.

Third, notice that the 1 bpp image has a 40.32 dB PSNR value and is virtually indistinguishable — even under very close examination — from the original. Here we find that SPIHT is able to exceed the simple thresholding compression we first discussed (see Fig. 6.5). For reasons of space, we cannot show SPIHT compressions of many test images, so in Table 6.2 we give PSNR values for several test images [19]. These data show that SPIHT produces higher PSNR values than the two other algorithms that we shall describe below. SPIHT is well-known for its superior performance

when PSNR is used as the error measure. High PSNR values, however, are not the sole criterion for the performance of lossy compression algorithms. We discuss other criteria below.

Table 6.2 PSNR Values, *With Arithmetic Compression*

Image/Method	SPIHT	WDR	ASWDR
Lena, 0.5 bpp	37.09	36.45	36.67
Lena, 0.25 bpp	33.85	33.39	33.64
Lena, 0.125 bpp	30.85	30.42	30.61
Goldhill, 0.5 bpp	33.10	32.70	32.85
Goldhill, 0.25 bpp	30.49	30.33	30.34
Goldhill, 0.125 bpp	28.39	28.25	28.23
Barbara, 0.5 bpp	31.29	30.68	30.87
Barbara, 0.25 bpp	27.47	26.87	27.03
Barbara, 0.125 bpp	24.77	24.30	24.52
Airfield, 0.5 bpp	28.57	28.12	28.36
Airfield, 0.25 bpp	25.90	25.49	25.64
Airfield, 0.125 bpp	23.68	23.32	23.50

Fourth, these SPIHT compressed images were obtained using SPIHT's arithmetic compression option. The method that SPIHT uses for arithmetic compression is quite involved and space does not permit a discussion of the details here. Some details are provided in Said and Pearlman [47].

Finally, it is interesting to compare SPIHT compressions with compressions obtained with the JPEG method². The JPEG method is a sophisticated implementation of block discrete cosine transform encoding [67, 38]. It is used extensively for compression of images, especially for transmission over the Internet. In Fig. 6.14, we compare compressions of the "Lena" image obtained with JPEG and with SPIHT at three different compression ratios. (JPEG does not allow for specifying the bpp value in advance; the 59:1 compression was the closest we could get to 64:1.) It is clear from these images that SPIHT is far superior to JPEG. It is better both in perceptual quality and in terms of PSNR. Notice, in particular, that the 59:1 JPEG compression is very distorted (exhibiting blocking artifacts stemming from coarse quantization within the blocks making up the block DCT used by JPEG). The SPIHT compression, even at the slightly higher ratio of 64:1, exhibits none of these objectionable features. In fact, for quick transmission of a thumbnail image (say, as part of a much larger webpage), this SPIHT compression would be quite acceptable. The 32:1 JPEG image might be

²JPEG stands for Joint Photographic Experts Group, a group of engineers who developed this compression method.

acceptable for some applications, but it also contains some blocking artifacts. The 32:1 SPIHT compression is almost indistinguishable (at these image sizes) from the original “Lena” image. The 16:1 compressions for both methods are nearly indistinguishable. In fact, they are both nearly indistinguishable from the original “Lena” image.

Although we have compared JPEG with SPIHT using only one image, the results we have found are generally valid. SPIHT compressions are superior to JPEG compressions both in perceptual quality and in PSNR values. In fact, all of the wavelet-based image compression techniques that we discuss here are superior to JPEG. Hence, we will not make any further comparisons with the JPEG method.

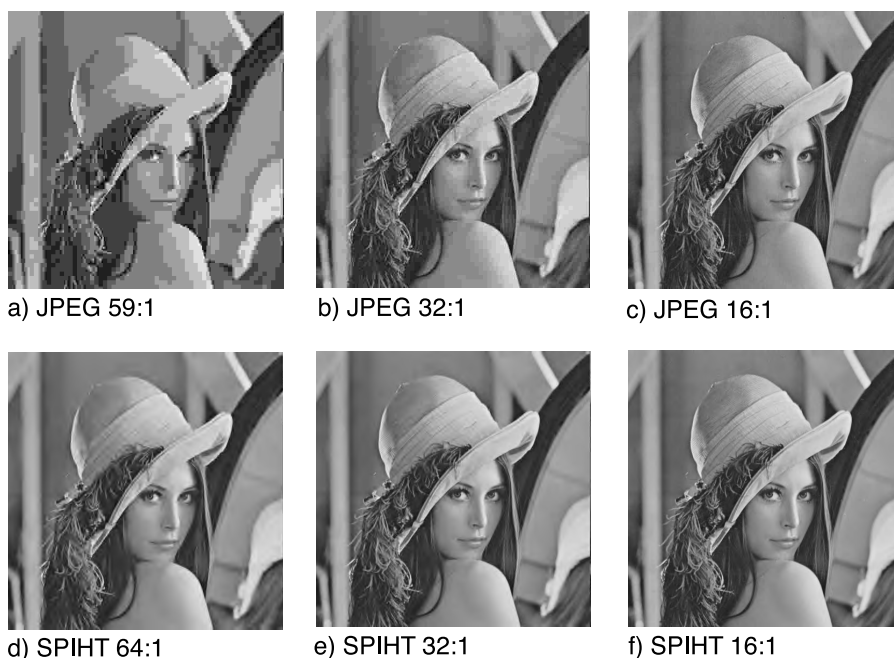


FIGURE 6.14

Comparison of JPEG and SPIHT compressions of “Lena” image. PSNR values: (a) 24.16 dB. (b) 30.11 dB. (c) 34.12 dB. (d) 30.85 dB. (e) 33.93 dB. (f) 37.09 dB. Reproduced by Special Permission of *Playboy* magazine. Copyright ©1972, 2000 by Playboy.

6.3.4 WDR Algorithm

One of the defects of SPIHT is that it only *implicitly* locates the position of significant coefficients. This makes it difficult to perform operations which depend on the exact position of significant transform values, such as region selection on compressed data. By *region selection*, also known as *region of interest* (ROI), we mean selecting

a portion of a compressed image that requires increased resolution. This can occur, for example, with a portion of a low resolution medical image that has been sent at a low bpp rate in order to arrive quickly.

Such compressed data operations are possible with the wavelet difference reduction (WDR) algorithm of Tian and Wells [56]–[58]. The term *difference reduction* refers to the way in which WDR encodes the locations of significant wavelet transform values, which we describe below. Although WDR will not typically produce higher PSNR values than SPIHT (see Table 6.2), we will see that WDR can produce perceptually superior images, especially at high compression ratios.

The only difference between WDR and the bit-plane encoding described above is in the significance pass. In WDR, the output from the significance pass consists of the signs of significant values along with sequences of bits which concisely describe the precise locations of significant values. The best way to see how this is done is to consider a simple example.

Suppose that the significant values are $w(2) = +34.2$, $w(3) = -33.5$, $w(7) = +48.2$, $w(12) = +40.34$, and $w(34) = -54.36$. The indices for these significant values are 2, 3, 7, 12, and 34. Rather than working with these values, WDR works with their successive differences: 2, 1, 4, 5, 22. In this latter list, the first number is the *starting index*, and each successive number is the *number of steps* needed to reach the next index. The binary expansions of these successive differences are $(10)_2$, $(1)_2$, $(100)_2$, $(101)_2$, and $(10110)_2$. Since the most significant bit for each of these expansions is always 1, this bit can be dropped and the signs of the significant transform values can be used instead as separators in the symbol stream. The resulting symbol stream for this example is then $+0 - +00 + 01 - 0110$.

When this most significant bit is dropped, we will refer to the binary expansion that remains as the reduced binary expansion. Notice, in particular, that the reduced binary expansion of 1 is empty. The reduced binary expansion of 2 is just the 0 bit, the reduced binary expansion of 3 is just the 1 bit, and so on.

The WDR algorithm simply consists of replacing the significance pass in the bit-plane encoding procedure with the following step:

WDR Step 3: Significance pass. Perform the following procedure on the insignificant indices in the baseline scan order:

```

Initialize step-counter  $C = 0$ 
Let  $C_{\text{old}} = 0$ 
Do
    Get next insignificant index  $m$ 
    Increment step-counter  $C$  by 1

```

```

If  $|w(m)| \geq T_k$  then
    Output sign  $w(m)$  and set  $w_Q(m) = T_k$ 
    Move  $m$  to end of sequence of significant indices
    Let  $n = C - C_{\text{old}}$ 
    Set  $C_{\text{old}} = C$ 
    If  $n > 1$  then
        Output reduced binary expansion of  $n$ 
    Else if  $|w(m)| < T_k$  then
        Let  $w_Q(m)$  retain its initial value of 0.
Loop until end of insignificant indices
Output end-marker

```

The output for the end-marker is a plus sign, followed by the reduced binary expansion of $n = C + 1 - C_{\text{old}}$, and a final plus sign.

It is not hard to see that WDR is of no greater computational complexity than SPIHT. For one thing, WDR does not need to search through quadrees as SPIHT does. The calculations of the reduced binary expansions adds some complexity to WDR, but they can be done rapidly with bit-shift operations. As explained in Tian and Wells [56]–[58], the output of the WDR encoding can be arithmetically compressed. The method that they describe is based on the elementary arithmetic coding algorithm described in Witten, Neal, and Cleary [68]. This form of arithmetic coding is substantially less complex (at the price of poorer performance) than the arithmetic coding employed by SPIHT.

As an example of the WDR algorithm, consider the scan order and wavelet transform shown in Fig. 6.8. For the threshold $T_1 = 32$, the significant values are $w(1) = 63$, $w(2) = -34$, $w(5) = 49$, and $w(36) = 47$. The output of the WDR significance pass will then be the following string of symbols:

+ - + 1 + 1 1 1 1 + 1 1 0 1 +

which compares favorably with the EZW output in Eq. (6.8). The last six symbols are the code for the end-marker. For the threshold $T_2 = 16$, the new significant values are $w(3) = -31$, $w(4) = 23$, $w(9) = -25$, and $w(24) = 18$. Since the previous indices 1, 2, 5, and 36, are removed from the sequence of insignificant indices, the values of n in the WDR significance pass will be 1, 1, 4, and 15. In this case, the value of n for the end-marker is 40. Adding on the four refinement bits, which are the same as in Eq. (6.9), the WDR output for this second threshold is

- + - 0 0 + 1 1 1 + 0 1 0 0 0 + 1 0 1 0

which is also a smaller output than the corresponding EZW output. It is also clear that, for this simple case, WDR does *not* produce as compact an output as STW does.

As an example of WDR performance for a natural image, Fig. 6.15 shows several compressions of the “Lena” image. These compressions were produced with free software [16].

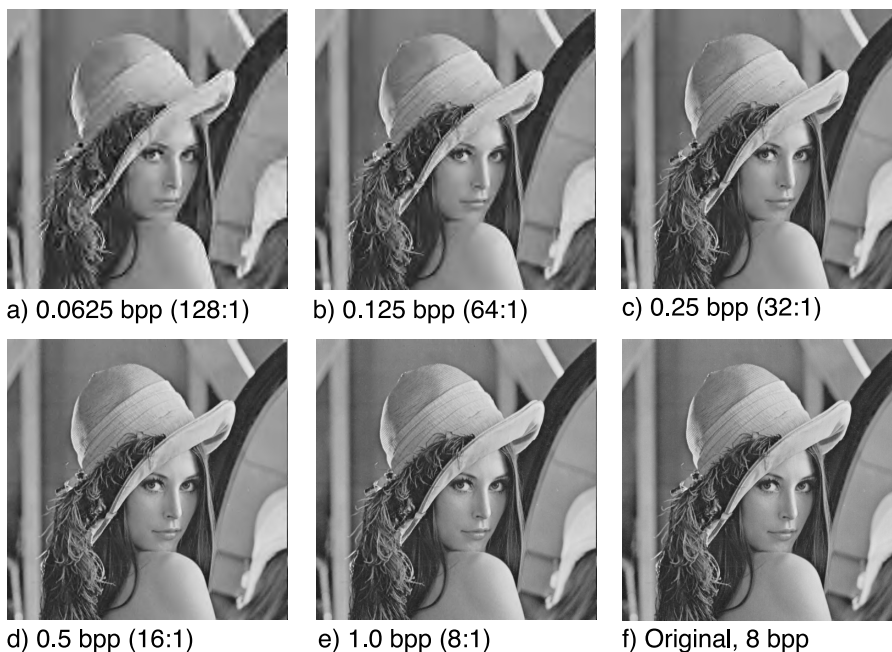


FIGURE 6.15

WDR compressions of “Lena” image. PSNR values: (a) 27.63 dB. (b) 30.42 dB. (c) 33.39 dB. (d) 36.45 dB. (e) 39.62 dB. Reproduced by Special Permission of *Playboy* magazine. Copyright ©1972, 2000 by Playboy.

There are a couple things to observe about these compressions. First, the PSNR values are lower than for SPIHT. This is typically the case. In Table 6.2 we compare PSNR values for WDR and SPIHT on several images at various compression ratios. In every case, SPIHT has higher PSNR values.

Second, at high compression ratios, the visual quality of WDR compressions of “Lena” are superior to those of SPIHT. For example, the 0.0625 bpp and 0.125 bpp compressions have higher resolution with WDR. This is easier to see if the images are magnified as in Fig. 6.16. At 0.0625 bpp, the WDR compression does a better job in preserving the shape of Lena’s nose and in retaining some of the striping in the band around her hat. Similar remarks apply to the 0.125 bpp compressions. SPIHT, however, does a better job in preserving parts of Lena’s eyes. These observations point to the need for an objective, quantitative measure of image quality.

There is no universally accepted objective measure for image quality. We shall now describe a simple measure that we have found useful. There is some evidence that the visual system of humans concentrates on analyzing edges in images [30, 40].

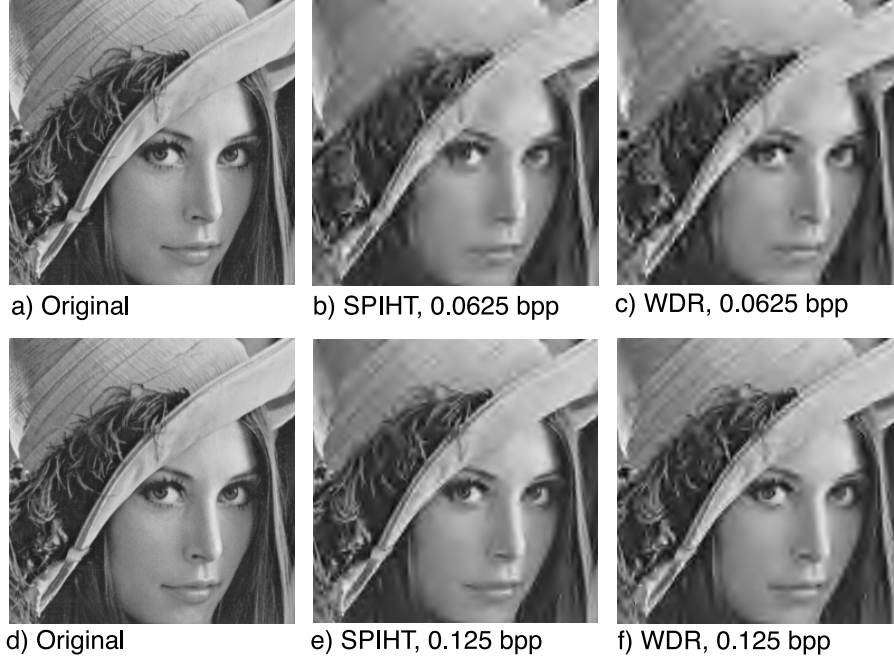


FIGURE 6.16

SPIHT and WDR compressions of “Lena” at low bpp. Reproduced by Special Permission of *Playboy* magazine. Copyright ©1972, 2000 by Playboy.

To produce an image that retains only edges, we proceed as follows. First, a 3-level Daub 9/7 transform of an image f is created. Second, the all-lowpass subband is subtracted away from this transform. Third, an inverse transform is performed on the remaining part of the transform. This produces a highpass filtered image, which exhibits edges from the image f . A similar highpass filtered image is created from the compressed image. Both of these highpass filtered images have mean values that are approximately zero. We define the *edge correlation* γ_3 by

$$\gamma_3 = \frac{\sigma_c}{\sigma_o}$$

where σ_c denotes the standard deviation of the values of the highpass filtered version of the compressed image, and σ_o denotes the standard deviation of the values of the highpass filtered version of the original image. Thus γ_3 measures how well the compressed image captures the variation of edge details in the original image.

Using this edge correlation measure, we obtained the results shown in Table 6.3. In every case, the WDR compressions exhibit higher edge correlations than the SPIHT compressions. These numerical results are also consistent with the increased preservation of details within WDR images, and with the informal reports of human observers.

Table 6.3 Edge Correlations, *With Arithmetic Compression*

Image/Method	SPIHT	WDR	ASWDR
Lena, 0.5 bpp	.966	.976	.978
Lena, 0.25 bpp	.931	.946	.951
Lena, 0.125 bpp	.863	.885	.894
Goldhill, 0.5 bpp	.920	.958	.963
Goldhill, 0.25 bpp	.842	.870	.871
Goldhill, 0.125 bpp	.747	.783	.781
Barbara, 0.5 bpp	.932	.955	.959
Barbara, 0.25 bpp	.861	.894	.902
Barbara, 0.125 bpp	.739	.767	.785
Airfield, 0.5 bpp	.922	.939	.937
Airfield, 0.25 bpp	.857	.871	.878
Airfield, 0.125 bpp	.766	.790	.803

Although WDR is simple, competitive with SPIHT in PSNR values, and often provides better perceptual results, there is still room for improvement. We now turn to a recent enhancement of the WDR algorithm.

6.3.5 ASWDR Algorithm

One of the most recent image compression algorithms is the adaptively scanned wavelet difference reduction (ASWDR) algorithm of Walker [66]. The adjective adaptively scanned refers to the fact that this algorithm modifies the scanning order used by WDR in order to achieve better performance.

ASWDR adapts the scanning order so as to predict locations of new significant values. If a prediction is correct, then the output specifying that location will just be the sign of the new significant value — the reduced binary expansion of the number of steps will be empty. Therefore a good prediction scheme will significantly reduce the coding output of WDR.

The prediction method used by ASWDR is the following: if $w(m)$ is significant for threshold T , then the values of the children of m are predicted to be significant for half-threshold $T/2$. For many natural images, this prediction method is a reasonably good one. As an example, Fig. 6.17 shows two vertical subbands for a Daub 9/7 wavelet transform of the “Lena” image. The image in Fig. 6.17(a) is of those significant values in the second level vertical subband for a threshold of 16 (significant values shown in white). In Fig. 6.17(b), we show the *new* significant values in the first vertical subband for the half-threshold of 8. Notice that there is a great deal of similarity in the two images. Since the image in Fig. 6.17(a) is magnified by two in each dimension, its white pixels actually represent the predictions for the locations of new significant values in the first vertical subband. Although these predictions are not

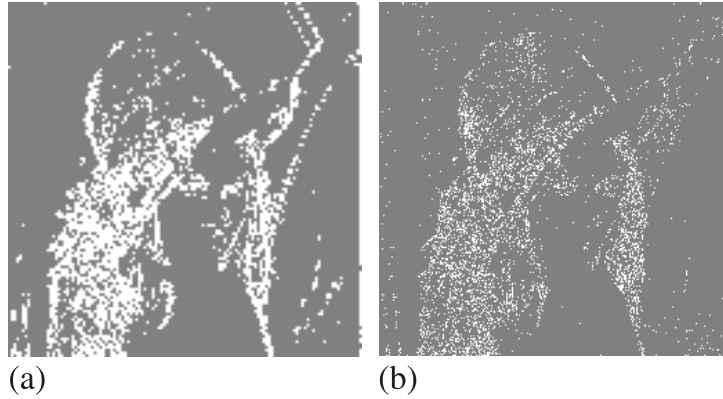


FIGURE 6.17

(a) Significant values, second vertical subband, threshold 16. (b) *New* significant values, first vertical subband, threshold 8. Reproduced by Special Permission of *Playboy* magazine. Copyright ©1972, 2000 by Playboy.

perfectly accurate, there is a great deal of overlap between the two images. Notice also how the locations of significant values are highly correlated with the location of edges in the “Lena” image. The scanning order of ASWDR dynamically adapts to the locations of edge details in an image, and this enhances the resolution of these edges in ASWDR compressed images.

Table 6.4 Number of Significant Values Encoded, *No* Arithmetic Coding

Image\Method	WDR	ASWDR	% increase
Lena, 0.125 bpp	5,241	5,458	4.1%
Lena, 0.25 bpp	10,450	11,105	6.3%
Lena, 0.5 bpp	20,809	22,370	7.5%
Goldhill, 0.125 bpp	5,744	5,634	−1.9%
Goldhill, 0.25 bpp	10,410	10,210	−1.9%
Goldhill, 0.5 bpp	22,905	23,394	2.1%
Barbara, 0.125 bpp	5,348	5,571	4.2%
Barbara, 0.25 bpp	11,681	12,174	4.2%
Barbara, 0.5 bpp	23,697	24,915	5.1%
Airfield, 0.125 bpp	5,388	5,736	6.5%
Airfield, 0.25 bpp	10,519	11,228	6.7%
Airfield, 0.5 bpp	19,950	21,814	9.3%

A complete validation of the prediction method just described would require assembling statistics for a large number of different subbands, thresholds, and images.

Rather than attempting such an a priori argument (see [6, 66]), we instead argue from an a posteriori standpoint. We present statistics that show that the prediction scheme employed by ASWDR does, in fact, encode more significant values than are encoded by WDR for a number of different images. As the pseudocode presented below shows, the only difference between ASWDR and WDR is in the predictive scheme employed by ASWDR to create new scanning orders. Consequently, if ASWDR typically encodes more values than WDR does, this must be due to the success of the predictive scheme.

Table 6.4 shows the numbers of significant values encoded by WDR and ASWDR for four different images. In almost every case, ASWDR was able to encode more values than WDR. This gives an a posteriori validation of the predictive scheme employed by ASWDR.

We now present the pseudocode description of ASWDR encoding. Notice that the significance pass portion of this procedure is the same as the WDR significance pass described above, and that the refinement pass is the same as for bit-plane encoding (hence the same as for WDR). The one new feature is the insertion of a step for creating a new scanning order.

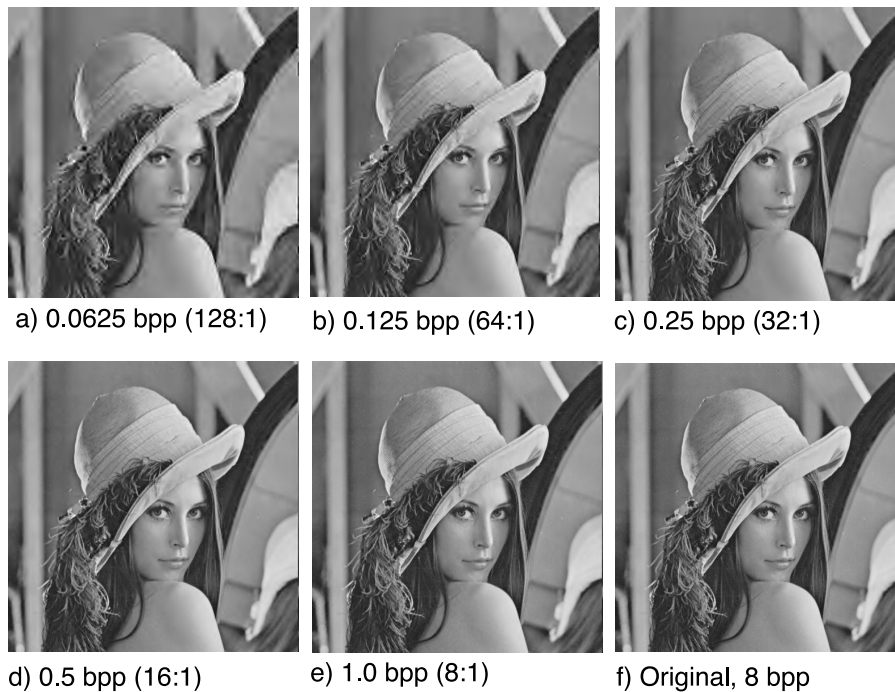


FIGURE 6.18

ASWDR compressions of “Lena image.” PSNR values: (a) 27.73 dB. (b) 30.61 dB. (c) 33.64 dB. (d) 36.67 dB. (e) 39.90 dB. Reproduced by Special Permission of *Playboy* magazine. Copyright ©1972, 2000 by Playboy.

ASWDR encoding —

Step 1: Initialize. Choose initial threshold, $T = T_0$, such that *all* transform values satisfy $|w(m)| < T_0$ and at least one transform value satisfies $|w(m)| \geq T_0/2$. Set the initial scan order to be the baseline scan order.

Step 2: Update threshold. Let $T_k = T_{k-1}/2$.

Step 3: Significance pass. Perform the following procedure on the insignificant indices in the scan order:

```
Initialize step-counter  $C = 0$ 
Let  $C_{\text{old}} = 0$ 
Do
  Get next insignificant index  $m$ 
  Increment step-counter  $C$  by 1
  If  $|w(m)| \geq T_k$  then
    Output sign  $w(m)$  and set  $w_Q(m) = T_k$ 
    Move  $m$  to end of sequence of significant indices
    Let  $n = C - C_{\text{old}}$ 
    Set  $C_{\text{old}} = C$ 
    If  $n > 1$  then
      Output reduced binary expansion of  $n$ 
  Else if  $|w(m)| < T_k$  then
    Let  $w_Q(m)$  retain its initial value of 0.
Loop until end of insignificant indices
Output end-marker as per WDR Step 3
```

Step 4: Refinement pass. Scan through significant values found with higher threshold values T_j , for $j < k$ (if $k = 1$ skip this step). For each significant value $w(m)$, do the following:

```
If  $|w(m)| \in [w_Q(m), w_Q(m) + T_k)$ , then
  Output bit 0
Else if  $|w(m)| \in [w_Q(m) + T_k, w_Q(m) + 2T_k)$ , then
  Output bit 1
  Replace value of  $w_Q(m)$  by  $w_Q(m) + T_k$ .
```

Step 5: Create new scan order. For the highest-scale level (the one containing the all-lowpass subband), use the indices of the remaining insignificant values as the scan order at that level. Use the scan order at level j to create the new scan order at level $j - 1$ as follows. The first part of the new scan order at level $j - 1$ consists of the insignificant children of the significant values at level j . The second part of the new scan order at level $j - 1$ consists of the insignificant children of the insignificant values at level j . Use this new scan order for level $j - 1$ to create the new scan order at level $j - 2$, until all levels are exhausted.

Step 6: Loop. Repeat steps 2 through 5.

The creation of the new scanning order only adds a small degree of complexity to the original WDR algorithm. Moreover, ASWDR retains all of the attractive features of WDR: simplicity, progressive transmission capability, and ROI capability.



FIGURE 6.19

SPIHT, WDR, and ASWDR compressions of “Lena” at low bpp. (a)–(c) 0.0625 bpp, 128:1. (d)–(f) 0.125 bpp, 64:1. Reproduced by Special Permission of *Playboy* magazine. Copyright ©1972, 2000 by Playboy.

Fig. 6.18 shows how ASWDR performs on the Lena image. The PSNR values for these images are slightly better than those for WDR, and almost as good as those for SPIHT. More importantly, the perceptual quality of ASWDR compressions are better than SPIHT compressions and slightly better than WDR compressions. This is especially true at high compression ratios. Fig. 6.19 shows magnifications of 128:1 and 64:1 compressions of the “Lena” image. The ASWDR compressions better preserve the shape of Lena’s nose and details of her hat, and show less distortion along the side of her left cheek (especially for the 0.125 bpp case). These subjective observations are borne out by the edge correlations in Table 6.3. In almost every case, the ASWDR compressions produce slightly higher edge correlation values.

As a further example of the superior performance of ASWDR at high compression ratios, in Fig. 6.20 we show compressions of the “airfield” image at 128:1. The WDR and ASWDR algorithms preserve more of the fine details in the image. Look especially along the top of the images: SPIHT erases many fine details such as

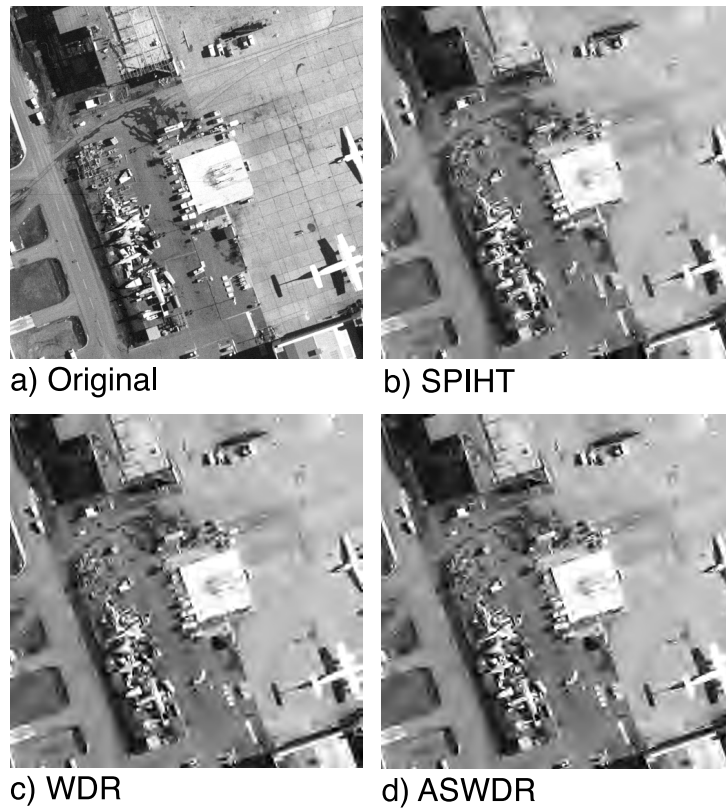


FIGURE 6.20

Comparisons of 128:1 compressions of “airfield” image. (From Walker, James S., A lossy image codec based on adaptively scanned wavelet difference reduction, in *Optical Engineering*, July 2000. With permission.)

the telephone pole and two small square structures to the right of the thin black rectangle. These details are preserved, at least partially, by both WDR and ASWDR. The ASWDR image does the best job in retaining some structure in the telephone pole. ASWDR is also superior in preserving the structure of the swept-back winged aircraft, especially its thin nose, located to the lower left of center. These are only a few of the many details in the airplane image which are better preserved by ASWDR.

As quantitative support for the superiority of ASWDR in preserving edge details, we show in [Table 6.5](#) the values for three different edge correlations γ_k , $k = 3, 4$, and 5. Here k denotes how many levels in the Daub 9/7 wavelet transform were used. A higher value of k means that edge detail at lower resolutions was considered in computing the edge correlation. These edge correlations show that ASWDR is superior over several resolution levels in preserving edges in the “airfield” image at the low bit rate of 0.0625 bpp.

Table 6.5 Edge Correlations for 128:1
Compressions of “Airfield” Image

Corr./Method	SPIHT	WDR	ASWDR
γ_3	.665	.692	.711
γ_4	.780	.817	.827
γ_5	.845	.879	.885

High compression ratio images like these are used in reconnaissance and in medical applications, where fast transmission and ROI (region selection) are employed, as well as multiresolution detection. The WDR and ASWDR algorithms do allow for ROI while SPIHT does not. Furthermore, their superior performance in displaying edge details at low bit rates facilitates multiresolution detection.

Further research is being done on improving the ASWDR algorithm. One important enhancement will be the incorporation of an improved predictive scheme, based on weighted values of neighboring transform magnitudes as described in Buccigrossi and Simoncelli [6].

6.3.6 Lossless Compression

A novel aspect of the compression/decompression methods diagrammed in Figs. 6.3 and 6.4 is that *integer-to-integer* wavelet transforms can be used in place of the ordinary wavelet transforms (such as Daub 9/7) described so far. An integer-to-integer wavelet transform produces an integer-valued transform from the gray-scale, integer-valued image [7]. Since n loops in bit-plane encoding reduces the quantization error to less than $T_0/2^n$, it follows that *once 2^n is greater than T_0 , there will be zero error*. In other words, the bit-plane encoded transform will be exactly the same as the original wavelet transform; hence lossless encoding is achieved (with progressive transmission as well). Of course, for many indices, the zero error will occur sooner than with the maximum number of loops n . Consequently, some care is needed in order to efficiently encode the minimum number of bits in each binary expansion. A discussion of how SPIHT is adapted to achieve lossless encoding can be found in Said and Pearlman [47]. The algorithms WDR and ASWDR can also be adapted in order to achieve lossless encoding (public versions of these adaptations are available [16].)

6.3.7 Color Images

Following the standard practice in image compression research, we have concentrated here on methods of compressing gray-scale images. For color images, this corresponds to compressing the *intensity* portion of the image. That is, if the color image is a typical RGB image, with 8 bits for red, 8 bits for green, and 8 bits for blue, then the intensity I is defined by $I = (R + B + G)/3$, which rounds to an 8-bit gray-scale image. The human eye is most sensitive to variations in intensity, so the most

difficult part of compressing a color image lies in the compressing of the intensity. Usually, the two color channels are denoted Y and C and are derived from the R , G , and B values [43]. Much greater compression can be done on the Y and C versions of the image since the human visual system is much less sensitive to variations in these two variables. Each of the algorithms described above can be modified so as to compress color images. For example, the public domain SPIHT coder [46] does provide programs for compressing color images. For reasons of space, we cannot describe compression of color images in any more detail.

6.3.8 Other Compression Algorithms

There is a wide variety of wavelet-based image compression algorithms besides the ones that we focused on here. Some of the most promising are algorithms that minimize the amount of memory which the encoder and/or decoder must use [20, 29]. A new algorithm which is embedded and which minimizes PSNR is described by Li and Lei [24]. Many other algorithms are cited in the review article by Davis and Nosratinia [14]. In evaluating the performance of any new image compression algorithm, one must take into account not only PSNR values but also the following factors: (1) perceptual quality of the images (edge correlation values can be helpful here); (2) whether the algorithm allows for progressive transmission; (3) the complexity of the algorithm (including memory usage); and (4) whether the algorithm has ROI capability.

6.3.9 Ringing Artifacts and Postprocessing Algorithms

As observed from the simulation for low bit rate (high compression ratio) compression, the decompressed image has ringing artifacts at the strong edges in the image. This is caused by the quantization process and by the overlapping nature of the wavelet transform. Edges have significant coefficients in all detailed subbands (horizontal, vertical, and diagonal subbands) and at low bit rate compression, these subband coefficients are quantized heavily. The ringing artifact at an edge is the linear combination of the (overlapping) wavelets and the quantization errors. Several post-processing algorithms are proposed to reduce the ringing artifacts [22, 36, 72] and improve the perceptual quality of the decompressed image. Simulation results, software, and further details can be found at <http://mmsplab.ece.wisc.edu/post/index.html>.

References

- [1] Antonini, M., Barlaud, M., Mathieu, P., and Daubechies, I., Image coding using the wavelet transform, *IEEE Trans. on Image Processing*, 1, 205–220, 1992.

- [2] Bamberger, R.H., Eddins, S.L., and Nuri, V., Generalized symmetric extension for size-limited multirate filter banks, *IEEE Trans. on Image Processing*, 3, 82–86, 1994.
- [3] Brislawn, C., Classification of symmetric wavelet transforms, *Los Alamos Tech. Report*, 1993.
- [4] Brislawn, C., A simple lattice architecture for even-order linear-phase perfect reconstruction filter banks, *Proc. IEEE-SP Intl. Symp. Time-Frequency and Time-Scale Analysis*, Philadelphia, PA, 124–127, 1994.
- [5] Brower, B.V., Low-bit-rate image compression evaluations, *Proc. SPIE*, Orlando, FL, April 4–9, 1994.
- [6] Buccigrossi, R.W. and Simoncelli, E.P., Image compression via joint statistical characterization in the wavelet domain, *IEEE Trans. on Image Processing*, 8(12), 1999.
- [7] Calderbank, A.R., Daubechies, I., Sweldens, W., and Yeo, B.-L., Wavelet transforms that map integers to integers, *Applied and Computational Harmonic Analysis*, 5(3), 332–369, 1998.
- [8] Cohen, A., *Ondelettes, analyses multirésolutions et traitement numérique du signal*, Ph.D. thesis, Université Paris IX, Dauphine, 1990.
- [9] Cohen, A., Daubechies, I., and Feauveau, J.-C., Biorthogonal bases of compactly supported wavelets, *Comm. Pure Appl. Math.*, 45, 1992.
- [10] Compression with Reversible Embedded Wavelets, RICOH Company Ltd. submission to ISO/IEC JTC1/SC29/WG1 for the JTC1.29.12 work item, 1995. Can be obtained on the World Wide Web, address: <http://www.crc.ricoh.com/CREW>.
- [11] Daubechies, I., *Ten Lectures on Wavelets*, CBMS Conference Series, SIAM, Philadelphia, 1992.
- [12] Daubechies, I., Orthonormal bases of compactly supported wavelets, *Comm. Pure Appl. Math.*, 41, 909–996, 1988.
- [13] Daubechies, I. and Lagarias, J., Two-scale difference equations I. Existence and global regularity of solutions, *SIAM J. Math. Anal.*, 22, 1388–1410, 1991.
- [14] Davis, G.M. and Nosratinia, A., Wavelet-based image coding: an overview, *Applied and Computational Control, Signals and Circuits*, 1(1), 1998.
- [15] Eiroa, T., Sobolev characterization of solutions of dilation equations, *SIAM J. Math. Anal.*, 23, 1015–1030, 1992.
- [16] WDR and ASWDR compressors are part of the FAWAV software package at <http://www.crcpress.com/edp/download/fawav/fawav.htm/>.

- [17] Gopinath, R.A., Odegard, J.E., and Burrus, C.S., Optimal wavelet representation of signals and the wavelet sampling theorem, *IEEE Transaction on Circuits & Systems II*, 41, 262–277, 1994.
- [18] Heller, P.N. and Wells, Jr., R.O., Spectral theory of multiresolution operators and applications, in *Wavelets: Theory, Algorithms, and Applications*, Chui, C.K., Ed., Academic Press, San Diego, CA, 13–31, 1994.
- [19] Go to <ftp://ipl.rpi.edu/pub/image/still/usc/gray/> for “Lena,” “Goldhill,” and “Barbara.” Go to <http://www.image.cityu.edu.hk/imagedb/> for “airfield.”
- [20] Islam, A. and Pearlman, W.A., An embedded and efficient low-complexity hierarchical image coder, *Proc. SPIE 3653, Visual Communications and Image Processing '99*, San Jose, CA, Jan. 1999.
- [21] Kiya, H., Nishikawa, K., and Iwahashi, M., A development of symmetric extension method for subband image coding, *IEEE Trans. on Image Processing*, 3, 78–81, 1994.
- [22] Shen, M. and Jay Kuo, C.C., Artifact removal in low bit rate wavelet coding with robust nonlinear filtering, *MMSP98*, 480–485, 1998.
- [23] Lawton, W., Necessary and sufficient conditions for construction orthonormal wavelet bases, *J. Math. Phys.*, 32, 57–61, 1991.
- [24] Li, J. and Lei, S., An embedded still image coder with rate-distortion optimization, *IEEE Trans. on Image Processing*, 8(7), 913–924, 1999.
- [25] Majani, E. and Lightstone, M., Biorthogonal wavelets for image compression, *Proc. 1994 Data Compression Conference*, Snowbird, Utah, 462, 1994.
- [26] Mallat, S., *A Wavelet Tour of Signal Processing*, Academic Press, New York, 1998.
- [27] Mallat, S., A theory for multiresolution signal decomposition: the wavelet representation, *IEEE Trans. PAMI*, 11, 674–693, 1989.
- [28] Mallat, S., Multifrequency channel decomposition of images and wavelet models, *IEEE Trans. on Acoust. Speech and Signal Processing*, 37(12), 2091–2110, 1989.
- [29] Malvar, H., Progressive wavelet coding of images, *Proc. of IEEE Data Compression Conference*, Salt Lake City, UT, 336–343, March 1999.
- [30] Marr, D., *Vision*, W.H. Freeman, San Francisco, CA, 1982.
- [31] Recommendation H.262, ISO/IEC 13818. Generic coding of moving picture and associated audio, Draft International Standard of MPEG-2.
- [32] Mintzer, F., Filters for distortion-free two-band multirate filter banks, *IEEE Trans. on ASSP*, 626–630, 1985.

- [33] Nayebi, K., Barnwell, III, T.P., and Smith, M.J.T., Time-domain filter bank analysis: a new design theory, *IEEE Trans. on Signal Processing*, 40, 1992.
- [34] Nguyen, T.Q. and Vaidyanathan, P.P., Two-channel perfect-reconstruction FIR QMF structures which yield linear-phase analysis and synthesis filters, *IEEE Trans. on ASSP*, 37, 676–690, 1989.
- [35] Nguyen, T.Q., A quadratic constrained least-squares approach to the design of digital filter banks, *Proc. IEEE ISCAS*, San Diego, 1344–1347, May 1992.
- [36] Oguz, S.H., Hu, Y.H., and Nguyen, T.Q., Morphological post-filtering of ringing and lost data concealment in generalized lapped orthogonal transform based image and video coding, Ph.D. thesis, University of Wisconsin, 1999. Additional informations can be found at <http://mmsplab.ece.wisc.edu/post/index.html>.
- [37] Orchard, M. and Ramchandran, K., An investigation of wavelet-based image coding using an entropy-constrained quantization framework, *Proc. Data Compression Conf.*, Snowbird, Utah, 341–350, 1994.
- [38] Pennebaker, W.B. and Mitchell, J.L., *JPEG: Still Image Compression Standard*, Van Nostrand Reinhold, New York, 1993.
- [39] Ramchandran, K. and Vetterli, M., Best wavelet packet bases in a rate-distortion sense, *IEEE Trans. on Image Processing*, 2, 160–175, 1993.
- [40] Ramos, M.G. and Hemami, S.S., Activity selective SPIHT coding, *Proc. SPIE 3653, Visual Communications and Image Processing '99*, San Jose, CA, Jan. 1999. See also errata for this paper at <http://foulard.ee.cornell.edu/marcia/asspiht2.html>.
- [41] Rioul, O., A discrete-time multiresolution theory, *IEEE Trans. on Signal Processing*, 41, 2591–2606, 1993.
- [42] Rioul, O. and Vetterli, M., Wavelets and signal processing, *IEEE Signal Processing Magazine*, 8(3), 14–38, 1991.
- [43] Russ, J.C., *The Image Processing Handbook*, CRC Press, Boca Raton, FL, 1995.
- [44] Said, A. and Pearlman, W.A., Image compression using the spatial-orientation tree, *IEEE Int. Symp. on Circuits and Systems*, Chicago, IL, 279–282, 1993.
- [45] Said, A. and Pearlman, W.A., A new, fast, and efficient image codec based on set partitioning in hierarchical trees, *IEEE Trans. on Circuits and Systems for Video Technology*, 6(3), 243–250, 1996.
- [46] SPIHT programs can be downloaded from <ftp://ipl.rpi.edu/pub/>.
- [47] Said, A. and Pearlman, W.A., An image multi-resolution representation for lossless and lossy image compression, *IEEE Trans. Image Processing*, 5(9), 1303–1310, 1996.

- [48] Shapiro, J.M., Embedded image coding using zerotrees of wavelet coefficients, *IEEE Trans. on Signal Processing*, 41, 3445–3462, 1993.
- [49] Shoham, Y. and Gersho, A., Efficient bit allocation for an arbitrary set of quantizers, *IEEE Trans. on ASSP*, 36, 1445–1453, 1988.
- [50] Smith, M.J.T. and Barnwell, III, T.P., Exact reconstruction techniques for tree-structured subband coders, *IEEE Trans. ASSP*, 34, 434–441, 1986.
- [51] Smith, M.J.T. and Eddins, S., Analysis-synthesis techniques for subband image coding, *IEEE Trans. ASSP*, 38, 1446–1456, 1990.
- [52] Strang, G., Wavelets and dilation equations, *SIAM Review*, 31, 614–627, 1989.
- [53] Strang, G. and Nguyen, T., *Wavelets and Filter Banks*, Wellesley-Cambridge Press, Wellesley, MA, 1997.
- [54] Vaidyanathan, P.P., *Multirate Systems and Filter Banks*, Prentice-Hall, Englewood Cliffs, NJ, 1993.
- [55] Vaidyanathan, P.P. and Hoang, P.Q., Lattice structures for optimal design and robust implementation of two-channel perfect-reconstruction QMF banks, *IEEE Trans. on ASSP*, 36, 81–94, 1988.
- [56] Tian, J. and Wells, Jr., R.O., A lossy image codec based on index coding, *IEEE Data Compression Conference, DCC '96*, 456, 1996.
- [57] Tian, J. and Wells, Jr., R.O., Embedded image coding using wavelet-difference-reduction, in *Wavelet Image and Video Compression*, Topiwala, P., Ed., 289–301, Kluwer Academic, Norwell, MA, 1998.
- [58] Tian, J. and Wells, Jr., R.O., Image data processing in the compressed wavelet domain, *3rd International Conference on Signal Processing Proc.*, Yuan, B. and Tang, X., Eds., 978–981, Beijing, China, 1996.
- [59] Vetterli, M., A theory of multirate filter banks, *IEEE Trans. on ASSP*, 35, 356–372, 1987.
- [60] Vetterli, M., Multidimensional subband coding: some theory and algorithms, *Signal Processing*, 6, 97–112, 1984.
- [61] Vetterli, M. and Herley, C., Wavelets and filter banks, *IEEE Trans. on Signal Processing*, 40, 2207–2233, 1992.
- [62] Vetterli, M. and LeGall, D., Perfect reconstruction FIR filter banks: some properties and factorization, *IEEE Trans. on ASSP*, 37, 1057–1071, 1989.
- [63] Villasenor, J.D., Belzer, B., and Liao, J., Wavelet filter evaluation for image compression, *IEEE Trans. on Image Processing*, 4, 1053–1060, 1995.
- [64] Villemoes, L., Energy moments in time and frequency for two-scale difference equation solutions and wavelets, *SIAM J. Math. Anal.*, 23, 1519–1543, 1992.

- [65] Volkmer, H., On the regularity of wavelets, *IEEE Trans. on Information Theory*, 38, 872–876, 1992.
- [66] Walker, J.S., A lossy image codec based on adaptively scanned wavelet difference reduction, *Optical Engineering*, in press.
- [67] Wallace, G.K., The JPEG still picture compression standard, *Comm. of the ACM*, 34(4), 30–44, 1991.
- [68] Witten, I., Neal, R., and Cleary, J., Arithmetic coding for compression, *Comm. of the ACM*, 30(6), 1278–1288, 1986.
- [69] Woods, J. and O’Neil, S.D., Subband coding of images, *IEEE Trans. on ASSP*, 34, 1278–1288, 1986.
- [70] Wavelet scalar quantization gray scale fingerprint image compression specification, Criminal Justice Information Services, FBI, Washington, DC, 1993.
- [71] Xiong, Z., Ramchandran, K., and Orchard, M., Joint optimization of scalar and tree-structured quantization of wavelet image decomposition, *27th Asilomar Conf.*, Pacific Grove, CA, November 1993.
- [72] Yang, S., Tull, D., Hu, Y.H., and Nguyen, T., Maximum a posteriori parameter estimation for image ringing artifact removal, submitted to the *IEEE Transaction on Image Processing*, 1999. Additional information can be found at <http://mmsplab.ece.wisc.edu/post/index.html>.
- [73] Zettler, W.R., Huffman, J., and Linden, D., The application of compactly supported wavelets to image compression, *Proc. SPIE*, 1244, 150–160, 1990.
- [74] Zhu, B., Tewfik, A.H., Colestock, M.A., Gerek, O.N., and Cetin, A.E., Image coding with wavelet representations, edge information and visual masking, *Proc. IEEE ICIP*, Washington, DC, 1995.