CHAPTER 14

3D WAVELET-BASED COMPRESSION OF HYPERSPECTRAL IMAGERY

James E. Fowler and Justin T. Rucker
Department of Electrical & Computer Engineering
GeoResources Institute
Mississippi State University

14.1 INTRODUCTION

Since hyperspectral imagery is generated by collecting hundreds of contiguous bands, uncompressed hyperspectral imagery can be very large, with a single image potentially occupying hundreds of megabytes. For instance, the Airborne Visible InfraRed Imaging Spectrometer (AVIRIS) sensor is capable of collecting several gigabytes of data per day. Compression is thus necessary to facilitate both the storage and the transmission of hyperspectral images. Since hyperspectral imagery is typically collected on remote acquisition platforms, such as satellites, the transmission of such data to central, often terrestrial, reception sites can be a critical issue. Thus, compression schemes oriented to the task of remote transmission are becoming increasingly of interest in hyperspectral applications.

Although there have been a number of approaches to the compression of hyperspectral imagery proposed in recent years—prominent techniques would include vector quantization (VQ) (e.g., [1, 2]) or principal component analysis (PCA) (e.g., [3, 4]) applied to spectral pixel vectors, as well as 3D extensions of common image-compression methods such as the discrete cosine transform (DCT) (e.g., [5])—most of the approaches as proposed are not particularly well-suited to the image-transmission task. That is, in many applications involving the communication of images, *progressive transmission* is desired in that successive reconstructions of the image are possible. In such a scenario, the receiver can produce a low-quality repre-

Hyperspectral Data Exploitation: Theory and Applications. By Chien-I Chang

sentation of the image after having received only a small portion of the transmitted bitstream, and this "preview" representation can be successively refined in quality as more and more of the bitstream is received. In many hyperspectral compression techniques, such progressive transmission is not supported, and, if the bitstream is not received in its entirety, no dataset can be reconstructed. In this case, the bits that are received are generally useless in the application. It is anticipated that progressive-transmission capabilities will be of increasing interest, particularly for hyperspectral applications involving satellite-to-ground communications which are inherently susceptible to transmission failure due to high noise levels and limited bandwidth.

Wavelet-based compression schemes have garnered significant attention in recent years in part due to their widespread support for progressive transmission. Wavelet-based compression techniques typically implement progressive transmission through the use of embedded coding. An *embedded coding* of a dataset can be defined as any coding such that 1) any prefix of length N bits of an M-bit coding is also a valid coding of the entire dataset, $0 < N \leq M$; and 2) if $N' > N$, then the quality upon reconstructing from the length-N' prefix is greater than or equal to that associated with the length-N prefix. Figs. 14.1 and 14.2 illustrate the difference between transmission of typical nonembedded and embedded codings. With an embedded coding, applications may be able to process partially reconstructed datasets—for example, in the case of a bitstream being truncated prematurely due to a communication failure—whereas the nonembedded bitstream is generally of little use unless received in its entirety.

In this chapter, we overview embedded wavelet-based algorithms as applied to the compression of hyperspectral imagery. First, we review the major components of which modern wavelet-based coders are composed in Sec. 14.2 as well as various measures of compression performance in Sec. 14.3. We then overview specific compression algorithms in Sec. 14.4. In Sec. 14.5, we consider several issues concerning encoder design for JPEG2000 [6–8], perhaps the most prominent wavelet-based coder used for hyperspectral compression. We follow with a body of experimental results in Sec. 14.6 that compares the relative compression performance of the various wavelet-based approaches considered. Finally, we make some concluding observations in Sec. 14.7.

14.2 EMBEDDED WAVELET-BASED COMPRESSION OF 3D IMAGERY

The general philosophy behind embedded coding lies in the recognition that each successive bit of the bitstream that is received improves the quality of the reconstructed image by a certain amount. Consequently, in order to achieve an embedded coding, we must organize information in the bitstream in decreasing order of importance, where the most important information is defined to be that which produces the greatest increase in quality upon reconstruction. Although it is usually not possible to exactly achieve this ordering in practice, modern embedded compression algorithms do come close to approximating this optimal embedded ordering. Embedded wavelet-based coders are based upon four major precepts: a wavelet transform; significance-map encoding; successive-approximation coding, i.e., bitplane coding; and some form of entropy coding, most often arithmetic coding. These components are described in detail below.

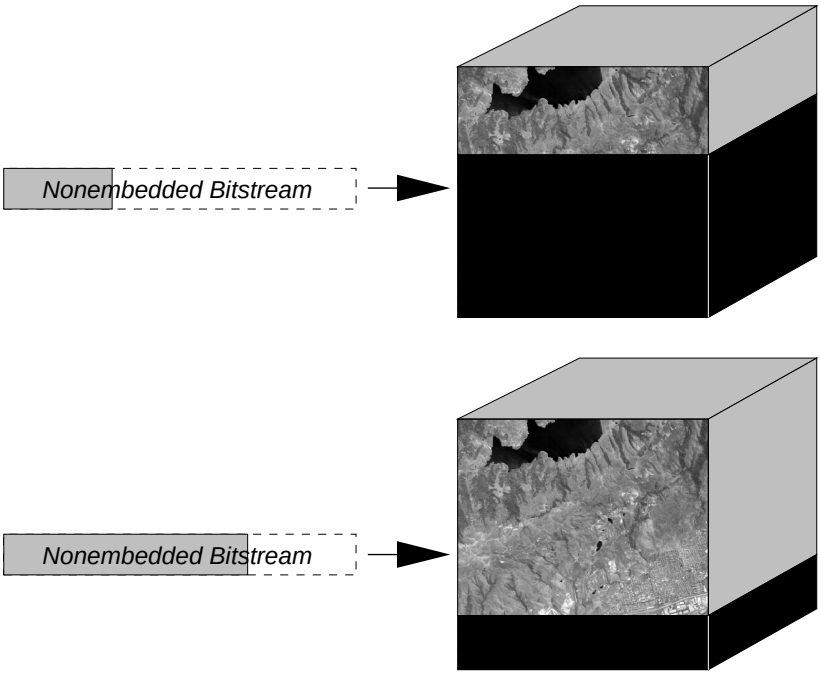

Figure 14.1. Transmission of a nonembedded coding.

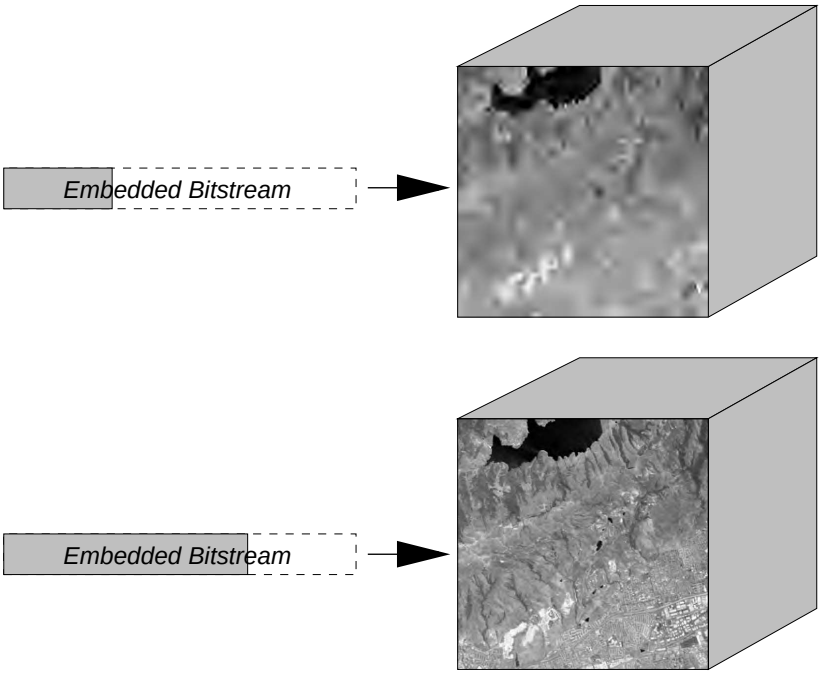

Figure 14.2. Transmission of an embedded coding.

14.2.1 Discrete Wavelet Transform (DWT)

Transforms aid the establishment of an embedded coding in that low-frequency components typically contain the majority of signal energy and are thus more important than high-frequency components to reconstruction. Wavelet transforms are currently the transform of choice for modern 2D image coders, since they not only provide this partitioning of information in terms of frequency but also retain much of the spatial structure of the original image. Wavelet-based coders for hyperspectral imagery extend the 2D transform structure into three dimensions.

A 2D *discrete wavelet transform* (DWT) can be implemented as a filter bank as illustrated in Fig. 14.3. This filter bank decomposes the original image into horizontal (H), vertical (V), diagonal (D), and baseband (B) subbands, each being one-fourth the size of the original image. Wavelet theory provides filter-design methods such that the filter bank is perfectly reconstructing (i.e., there exists a reconstruction filter bank that will generate exactly the original image from the decomposed subbands H, V, D, and B), and such that the lowpass and highpass filters have finite impulse responses (which aids practical implementation). Multiple stages of decomposition can be cascaded together by recursively decomposing the baseband; the subbands in this case are usually arranged in a pyramidal form as illustrated in Fig. 14.4.

For hyperspectral imagery, the 2D-transform decomposition of Fig. 14.4 is extended to three dimensions to accommodate the addition of the spectral dimension. A 3D wavelet transform, like the 2D transform, is implemented in separable fashion, employing 1D transforms separately in the spatial-row, spatial-column, and spectral-slice directions. However, the addition of a third dimension permits several options for the order of decomposition. For instance, we can perform one scale of decomposition along each direction, then further decompose the lowpass subband, leading to the dyadic decomposition, as is illustrated in Fig. 14.5. This dyadic decomposition structure is the most straightforward 3D generalization of the 2D dyadic decomposition of Fig. 14.4. However, in 3D, we can alternatively use a so-called wavelet-packet transform, in which we first decompose each spectral slice using a separable 2D transform and then follow with a 1D decomposition in the spectral direction. With this approach, we employ an m-scale decomposition spatially, followed by an n-scale decomposition spectrally, where it is possible for $m \neq n$. For example, the wavelet-packet transform depicted in Fig. 14.6 uses a three-scale decomposition ($m = n = 3$) in all directions. In comparing the two decomposition structures, the wavelet-packet transform is more flexible, because the spectral decomposition can be better tailored to the data at hand than in the dyadic transform. In Sec. 14.6.1, we will see that this wavelet-packet decomposition typically yields more efficient coding for hyperspectral datasets than the dyadic decomposition does. Additionally, it has been shown [9] that the particular wavelet-packet decomposition of Fig. 14.6 is typically very close in performance to the optimal 3D transform structure selected in a best-basis sense for the dataset at hand.

Wavelet-based coders, 2D or 3D, base their operation on the following observations about the DWT: 1) since most images are lowpass in nature, most signal energy is compacted into the baseband and lower-frequency subbands; 2) most coefficients are zero in the higher-frequency subbands; 3) small- or zero-valued coefficients tend to be clustered together within a given subband; and 4) clusters of

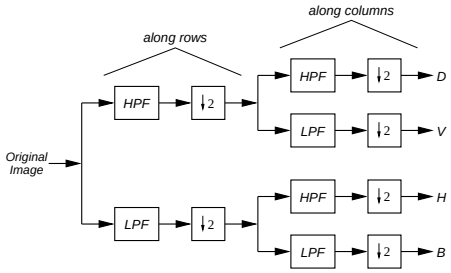

Figure 14.3. One stage of 2D DWT decomposition composed of lowpass (LPF) and highpass (HPF) filters applied to the columns and rows independently.

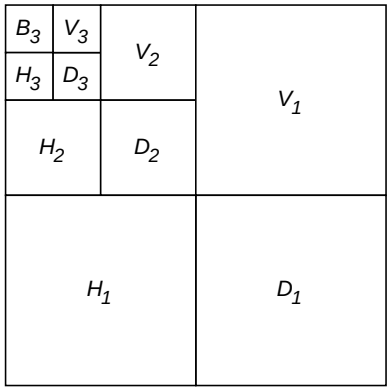

Figure 14.4. A 3-scale, 2D DWT pyramid arrangement of subbands.

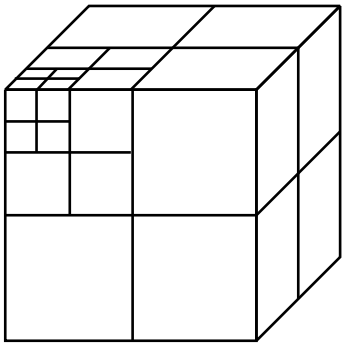

Figure 14.5. 2-level, 3D dyadic DWT

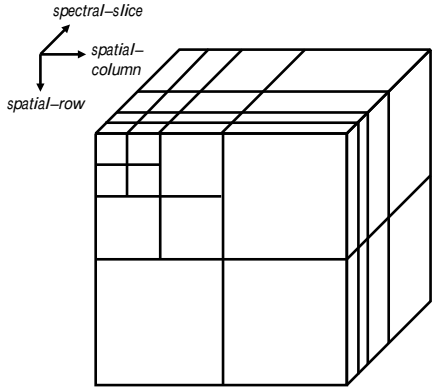

Figure 14.6. 3D packet DWT, with $m = 3$ spatial decompositions and $n = 3$ spectral decompositions

small- or zero-valued coefficients in one subband tend to be located in the same relative spatial/spectral position as similar clusters in subbands of the next decomposition scale. The techniques we describe in Sec. 14.4 exploit one or more of these DWT properties to achieve efficient coding performance.

14.2.2 Bitplane Coding

The partitioning of information into DWT subbands somewhat inherently supports embedded coding in that transmitting coefficients by ordering the subbands from the low-resolution baseband subband toward the high-resolution highpass subbands implements a decreasing order of importance. However, more is needed to produce a truly embedded bitstream—even if some coefficient is more important than some

other coefficient, not every bit of the first coefficient is necessarily more important than every bit of second. That is, not only should the coefficients be transmitted in decreasing order of importance, but also the individual bits that constitute the coefficients should be ordered as well.

Specifically, to effectuate an embedded coding of a set of coefficients, we represent the coefficients in sign-magnitude form as illustrated in Fig. 14.7 and code the sign and magnitude of the coefficients separately. For coefficient-magnitude coding, we transmit the most significant bit (MSB) of *all* coefficient magnitudes, then the next-most significant bit of all coefficient magnitudes, etc., such that each coefficient is successively approximated. This *bitplane-coding* scheme is contrary to the usual binary representation which would output all bits of a coefficient at once. The net effect of bitplane coding is that each coefficient magnitude is successively quantized by dividing the interval in which it is known to reside in half and outputting a bit to designate the appropriate subinterval, as illustrated in Fig. 14.8.

In practice, bitplane coding is usually implemented by performing two passes through the set of coefficients for each bitplane—the significance pass and the refinement pass. Suppose the coefficient located at position $[x_1, x_2, x_3]$ in the 3D hyperspectral volume is $c[x_1, x_2, x_3]$. We define the significance state with respect to threshold t of the coefficient as

$$s[x_1, x_2, x_3] = \begin{cases} 1, & \left|c[x_1, x_2, x_3]\right| \geq t, \\ 0, & \text{otherwise.} \end{cases} \tag{14.1}$$

We say that $c[x_1, x_2, x_3]$ is a *significant* coefficient when $s[x_1, x_2, x_3] = 1$; otherwise, $c[x_1, x_2, x_3]$ is *insignificant*. The significance pass describes $s[x_1, x_2, x_3]$ for all the coefficients in the DWT that are currently known to be insignificant but may become significant for the current threshold. On the other hand, the refinement pass produces a successive approximation to those coefficients that are already known to be significant by coding the current coefficient-magnitude bitplane for those significant coefficients. After each iteration of the significance and refinement passes, the significance threshold is divided in half, and the process is repeated for the next bitplane.

14.2.3 Significance-Map Coding

The collection of $s[x_1, x_2, x_3]$ values for all the coefficients in the DWT of an image is called the *significance map* for a particular threshold value. Given our observations in Sec. 14.2.1 of the nature of DWT coefficients, we see that for most of the bitplanes (particularly for large t), the significance map will be only sparsely populated with nonzero values. Consequently, the task of the significance pass is to create an efficient coding of this sparse significance map at each bitplane; the efficiency of this coding will be crucial to the overall compression efficiency of the coder. Sec. 14.4 is devoted to reviewing approaches that prominent algorithms have taken for the efficient coding of significance-map information. These algorithms are largely 2D image coders which have been extended to 3D and modified to accommodate the addition of spectral information.

Bitplane		Coefficients 11	2	−3	6
	Sign	0	0	1	0
MSB	3	1	0	0	0
	2	0	0	0	1
	1	1	1	1	1
LSB	0	1	0	1	0

Figure 14.7. Coefficients in sign-magnitude bitplane representation.

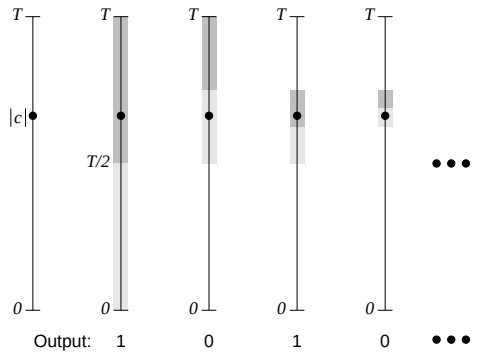

Figure 14.8. Successive-approximation quantization of a coefficient magnitude $|c|$ in interval $[0, T]$ where T is an integer power of 2.

14.2.4 Refinement and Sign Coding

In most embedded image coders, after the significance map is coded for a particular bitplane, a refinement pass proceeds through the coefficients, coding the current bitplane value of each coefficient that is already known to be significant but did not become significant in the immediately preceding significance pass. These *refinement bits* permit the reconstruction of the significant coefficients with progressively greater accuracy. It is usually assumed that the occurrence of a 0 or 1 is equally likely in bitplanes other than the MSB for a particular coefficient; consequently, most algorithms take little effort to code the refinement bits and may simply output them unencoded into the bitstream. Recently, it has been recognized that the refinement bits typically possess some correlation to their neighboring coefficients [10], particularly for the more significant bitplanes; consequently, some coders (e.g., JPEG2000) employ entropy coding for refinement bits.

The significance and refinement passes encode the coefficient magnitudes; to reconstruct the wavelet coefficients, the coefficient signs must also be encoded. As with the refinement bits, most algorithms assume that any given coefficient is equally likely to be positive or negative; however, recent work [10–12] has shown that there is some structure to the sign information that can be exploited to improve coding efficiency. Thus, certain coders (e.g., JPEG2000) also employ entropy coding for coefficient signs as well as for refinement bits.

14.2.5 Arithmetic Coding

Most wavelet-based coders incorporate some form of lossless entropy coding at the final stage before producing the compressed bitstream. In essence, such entropy coders assign shorter bitstream codewords to more frequently occurring symbols in order to maximize the compactness of the bitstream representation.

Most wavelet-based coders use *adaptive arithmetic coding* (AAC) [13] for lossless entropy coding. AAC codes a stream of symbols into a bitstream with length very close to its theoretical minimum limit. Suppose source X produces symbol i with

probability p_i. The *entropy* of source X is defined to be

$$H(X) = -\sum_i p_i \log_2 p_i, \tag{14.2}$$

where $H(X)$ has units of bits per symbol (bps). One of the fundamental tenets of information theory is that the average bit rate in bps of the most efficient lossless (i.e., invertible) compression of source X cannot be less than $H(X)$. In practice, AAC often produces compression quite close to $H(X)$ by estimating the probabilities of the source symbols with frequencies of occurrence as it codes the symbol stream. Essentially, the better able AAC can estimate p_i, the closer it will come to the $H(X)$ lower bound on compression efficiency. Oftentimes, the efficiency of AAC can be improved by *conditioning* the coder with known *context* information and maintaining separate symbol-probability estimates for each context. That is, limiting attention of AAC to a specific context usually reduces the variety of symbols, thus permitting better estimation of the probabilities within that context and producing greater compression efficiency. From a mathematical standpoint, the *conditional entropy* of source X with known information Y is $H(X|Y)$. Since it is well known from information theory that

$$H(X|Y) \leq H(X), \tag{14.3}$$

conditioning AAC with Y as the context will (usually) produce a bitstream with a smaller bit rate.

14.3 PERFORMANCE MEASURES FOR HYPERSPECTRAL COMPRESSION

Traditionally, performance for lossy compression is determined by simultaneously measuring both *distortion* and *rate*. Distortion measures the fidelity of the reconstructed data to the original data, while rate essentially measures the amount of compression incurred. Distortion is commonly measured via a *signal-to-noise ratio* (SNR) between the original and reconstructed data. Let $c[x_1, x_2, x_3]$ be an $N_1 \times N_2 \times N_3$ hyperspectral dataset with variance of σ^2. Let $\widehat{c}[x_1, x_2, x_3]$ be the dataset as reconstructed from the compressed bitstream. The *mean squared error* (MSE) is defined as

$$\text{MSE} = \frac{1}{N_1 N_2 N_3} \sum_{x_1, x_2, x_3} \Big(c[x_1, x_2, x_3] - \widehat{c}[x_1, x_2, x_3] \Big)^2, \tag{14.4}$$

while the SNR in decibels (dB) is defined in terms of the MSE as

$$\text{SNR} = 10 \log_{10} \frac{\sigma^2}{\text{MSE}}. \tag{14.5}$$

Both the MSE and SNR provide a measure of the performance of a coder in an *average* sense over the entire volume. Such an average measure may or may not be of the greatest use depending on the application to be made of the reconstructed data. Hyperspectral imagery is often used in applications involving extensive analysis; consequently, it is paramount that the compression of hyperspectral data does

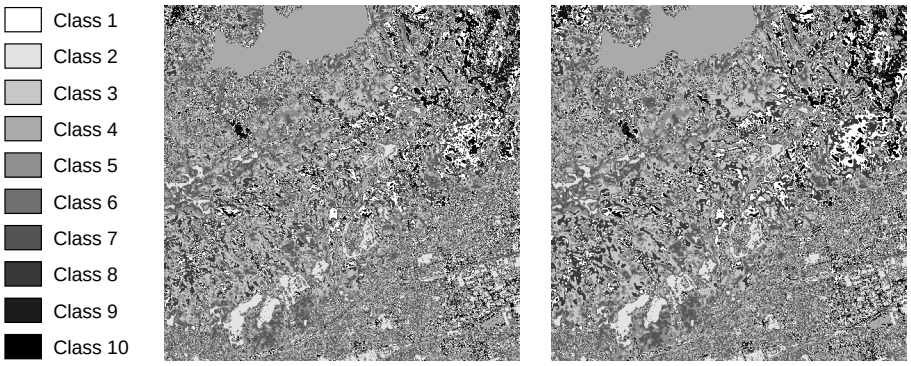

Figure 14.9. Classification map for the Moffett image using k-means classification. (a) Map for original image, (b) map after JPEG2000 compression.

not alter the outcome of such analysis. As an alternative to the SNR measure for distortion, one can examine the difference between performance of application-specific analysis as applied to the original data and the reconstructed data. As an example, unsupervised classification of hyperspectral pixel vectors is representative of methods that segment an image into multiple constituent classes. To form a distortion measure, we can apply unsupervised classification on the original hyperspectral image as well as on the reconstructed image, counting the number of pixels that change assigned class as a result of the compression. We call the resulting distortion measure *preservation of classification* (POC), which is measured as the percentage of pixels that do not change class due to compression.

In the subsequent experimental results reported in Sec. 14.6, all POC results are calculated using the ISODATA and k-means unsupervised classification as implemented in ENVI Version 4.0. A maximum of ten classes are used, and POC performance is determined by applying the classification to the original dataset as well as to the reconstructed volume and comparing the classification map produced for reconstructed volume to that of the original dataset. In this manner, the classification map of the original dataset is effectively used as "ground truth." Fig. 14.9 depicts typical classification maps generated in this manner.

In addition to distortion, it is necessary to gauge compression techniques according to the amount of compression incurred, due to the inherent trade-off between distortion and compression—the more highly compressed a reconstructed dataset is, the greater is the expected distortion between the original and reconstructed data. Typically, for hyperspectral imagery, one measures the rate as the number of *bits per pixel per band* (bpppb) which gives the average number of bits to represent a single sample of the hyperspectral dataset. A *compression ratio* can then be determined as the ratio of the bpppb of the original dataset (usually 16 bpppb) to the bpppb of the compressed dataset.

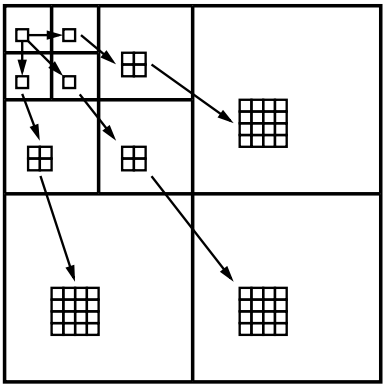

Figure 14.10. Parent-child relationships between subbands of a 2D DWT.

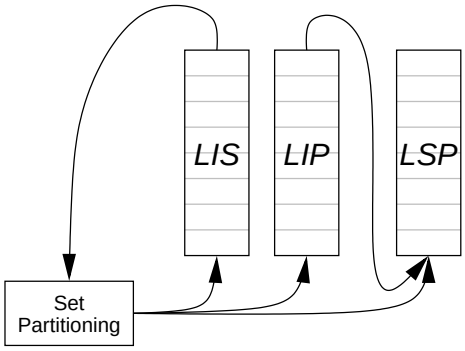

Figure 14.11. Processing of sorted lists in SPIHT.

14.4 PROMINENT TECHNIQUES FOR SIGNIFICANCE-MAP CODING

The primary difference between wavelet-based coding algorithms is how coding of the significance map is performed. Several techniques for significance-map coding that have been used for hyperspectral imagery are discussed below. These techniques were typically developed originally for 2D images and then subsequently extended and modified for 3D coding. As a consequence, we briefly overview the original 2D algorithm—which is usually more easily conceptualized—before discussing its 3D extension for each of the techniques considered below.

14.4.1 Zerotrees

Zerotrees are one of the most widely used techniques for coding significance maps in wavelet-based coders. Zerotrees capitalize on the fact that insignificant coefficients tend to cluster together within a subband, and clusters of insignificant coefficients tend to be located in the same location within subbands of different scales. As illustrated for a 2D DWT in Fig. 14.10, "parent" coefficients in a subband can be related to *four* "children" coefficients in the same relative spatial location in a subband at the next scale. A *zerotree* is formed when a coefficient and all of its descendants are insignificant with respect to the current threshold, while a *zerotree root* is defined to be a coefficient that is part of a zerotree yet is not the descendant of another zerotree root.

The Embedded Zerotree Wavelet (EZW) algorithm [14] was the first 2D image coder to make use of zerotrees for the coding of significance-map information. This coder is based on the observation that if a coefficient is found to be insignificant, it is likely that its descendants are also insignificant. Consequently, the occurrence of a zerotree root in the baseband or in the lower-frequency subbands can lead to substantial coding efficiency since we can denote the zerotree root as a special "Z" symbol in the significance map, and not code all of the descendants which are known then to be insignificant by definition. The EZW algorithm then proceeds to code the significance map in a raster scan within each subband, starting with

the baseband and progressing to the high-frequency subbands. A lossless entropy coding of symbols from this raster scan then produces a compact representation of the significance map.

The Set Partitioning in Hierarchical Trees (SPIHT) algorithm [15] improves upon the zerotree concept by replacing the raster scan with a number of sorted lists that contain sets of coefficients (i.e., zerotrees) and individual coefficients. These lists are illustrated in Fig. 14.11. In the significance pass of the SPIHT algorithm, the list of insignificant sets (LIS) is examined in regard to the current threshold; any set in the list that is no longer a zerotree with respect to the current threshold is then partitioned into one or more smaller zerotree sets, isolated insignificant coefficients, or significant coefficients. Isolated insignificant coefficients are appended to the list of insignificant pixels (LIP), while significant coefficients are appended to the list of significant pixels (LSP). The LIP is also examined, and, as coefficients become significant with respect to the current threshold, they are appended to the LSP. Binary symbols are encoded to describe motion of sets and coefficients between the three lists. Since the lists remain implicitly sorted in an importance ordering, SPIHT achieves a high degree of embedding and compression efficiency.

Originally developed for 2D images, SPIHT has been extended to 3D in several contexts [16–21]. In the case of a dyadic transform such as in Fig. 14.5, the 3D zerotree is a straightforward extension of the parent-child relationship of 2D zerotrees; that is, one coefficient is the parent to a $2 \times 2 \times 2$ cube of eight offspring coefficients in the next scale. However, in the case of a wavelet-packet transform, there are several approaches to fitting a zerotree structure to the wavelet coefficients. The first, proposed in [18], recognizes that wavelet-packet subbands appear as "split" versions of their dyadic counterparts; consequently, one should "split" the $2 \times 2 \times 2$ offspring nodes of the dyadic zerotree structure appropriately. An alternative zerotree structure for packet transforms was proposed in [19] and used subsequently in [20, 21]. In essence, this zerotree structure consists of 2D zerotrees within each "slice" of the subband-pyramid volume, with parent-child relationships setup between the tree-root coefficients of the 2D trees. Cho and Pearlman [20] called this alternative structure an *asymmetric packet zerotree*, with the original splitting-based packet structure of [18] then being a *symmetric packet zerotree*. The asymmetric structure, which is depicted in Fig. 14.12, usually offers somewhat more efficient compression performance than the symmetric packet structure [19–21]. Additionally, the wavelet-packet transform can have the number of spectral decomposition levels different from the number of spatial decomposition levels when the asymmetric tree is used, whereas the number of spatial and spectral decompositions must be the same in order to use the symmetric packet zerotree.

14.4.2 Spatial-Spectral Partitioning

Another approach to significance-map coding is *spatial-spectral partitioning*. The Set-Partitioning Embedded Block Coder (SPECK) [22, 23], originally developed as a 2D image coder, employs quadtree partitioning (see Fig. 14.13) in which the significance state of an entire block of coefficients is tested and coded. Then, if the block contains at least one significant coefficient, the block is subdivided into four subblocks of approximately equal size, and the significance-coding process is repeated recursively on each of the subblocks.

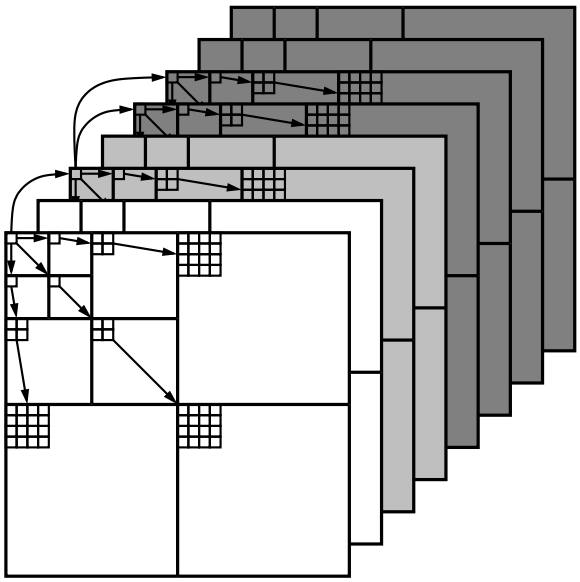

Figure 14.12. The asymmetric packet zerotree in a 3D packet DWT of $m = 3$ spatial decompositions and $n = 2$ spectral decompositions (adapted from [20]). The spectral subbands are indicated by different shades of gray.

In 2D-SPECK, there are two types of sets: S sets and I sets. The first S set is the baseband, and the first I set contains everything that remains. There are also two linked lists in SPECK: the List of Insignificant Sets (LIS), which contains sorted lists of decreasing sizes that have not been found to contain a significant pixel as compared with the current threshold, and the List of Significant Pixels (LSP), which contains single pixels that have been found to be significant through sorting and refinement passes. An S set remains in the LIS until it is found to be significant against the current threshold. The set is then divided into four approximately equal-sized sets, and the significance of each of the resulting four sets is tested. If the set is not significant, then it is placed in its appropriate place in the LIS. If the set is significant and contains single pixel, it is appended to the LSP; otherwise, the set is recursively split into four subsets. Following the significant pass, the coefficients in the LSP go through a refinement pass in which coefficients that have been previously found to be significant are refined.

The SPECK algorithm was extended to 3D in [24, 25] by replacing quadtrees with octrees as illustrated in Fig. 14.14. Unlike the original 2D-SPECK algorithm, the 3D-SPECK algorithm uses only one type of set, rather than having S and I sets as in 2D-SPECK. Consequently, each subband in the DWT decomposition is added to an LIS at the start of the 3D-SPECK algorithm, whereas the 2D algorithm initializes with only the baseband subband in an LIS. An advantage of the set-partitioning processing of 3D-SPECK is that sets are confined to reside within a single subband at all times throughout the algorithm, whereas sets in SPIHT (i.e., the zerotrees) span across scales. This is beneficial from a computational standpoint as the coder need buffer only a single subband at a given time, leading

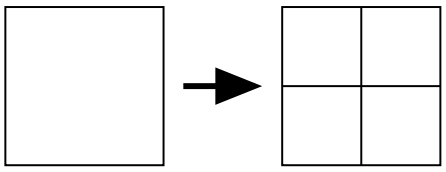

Figure 14.13. 2D quadtree block partitioning as performed in 2D SPECK.

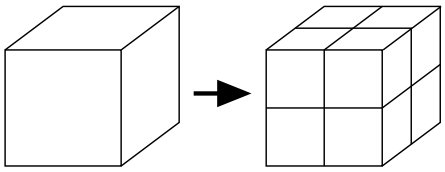

Figure 14.14. 3D octree cube partitioning as performed in 3D SPECK.

to reduced dynamic memory needed [23]. Furthermore, 3D-SPECK is easily applied to both the dyadic and packet transform structures of Figs. 14.5 and 14.6 with no algorithmic differences.

14.4.3 Conditional Coding

Recent work [26] has indicated that typically the ability to predict the insignificance of a coefficient through parent-child relationships, such as those employed by zerotree algorithms, is somewhat limited compared to the predictive ability of neighboring coefficients within the same subband. Consequently, recent algorithms, such as SPECK above, have focused on coding significance-map information using only within-subband information. Another approach to within-subband coding is to employ extensively conditioned, multiple-context AAC to capitalize on the theoretical advantages conditioning provides for entropy coding as discussed in Sec. 14.2.5.

The usual approach to employing AAC with context conditioning for the significance-map coding of an image is to use the known significance states of neighboring coefficients to provide the context for the coding of the significance state of the current coefficient. Assuming a 2D image, the eight neighboring significance states to x_i are shown in Fig. 14.15. Given that each neighbor takes on a binary value, there are $2^8 = 256$ possible contexts.

JPEG2000 [6–8], the most prominent *conditional-coding* technique, uses contexts derived from the neighbors depicted in Fig. 14.15, but reduces the number of distinct contexts to nine, since not all possible contexts were found to be useful. The context definitions, which vary from subband to subband, are shown in Fig. 14.16. To further improve the context conditioning, as well as to increase the degree of embedding, JPEG2000 splits the coding of the significance map into two separate passes rather than employing one significance pass as do most other algorithms. Specifically, JPEG2000 uses a significance-propagation pass that codes those coefficients that are currently insignificant but have at least one neighbor that is already significant. This pass accounts for all coefficients that are likely to become significant in the current bitplane. The remaining insignificant coefficients are coded in the cleanup pass; these coefficients, which are surrounded by insignificant coefficients, are likely to remain insignificant. Both passes use the same nine contexts depicted in Fig. 14.16. contexts. In addition, the cleanup pass includes one additional context used to encode four successive insignificant coefficients together with a single "insignificant run" symbol.

To code a single-band (i.e., 2D) image, a JPEG2000 encoder first performs a 2D wavelet transform on the image and then partitions each transform subband into small, 2D rectangular blocks called codeblocks, which are typically of size

d_0	v_0	d_1
h_0	x_i	h_1
d_2	v_1	d_3

Figure 14.15. Significance-state neighbors to x_i.

32×32 or 64×64 pixels. Subsequently, the JPEG2000 encoder independently generates an embedded bitstream for each codeblock. To assemble the individual codeblock bitstreams into a single, final bitstream, each codeblock bitstream is truncated in some fashion, and the truncated bitstreams are concatenated together to form the final bitstream. The method for codeblock-bitstream truncation is an implementation issue concerning only the encoder as codeblock-bitstream lengths are conveyed to the decoder as header information. Consequently, this truncation process is not covered by the JPEG2000 standard.

It is highly likely that, for codeblocks residing in a single spectral band, any given JPEG2000 encoder with perform a Lagrangian rate-distortion optimal truncation as described as part of Taubman's EBCOT algorithm [8, 10]. This optimal truncation technique, *post-compression rate-distortion (PCRD) optimization*, is a primary factor in the excellent rate-distortion performance of the EBCOT algorithm. PCRD optimization is performed simultaneously across all of the codeblocks from the image, producing an optimal truncation point for each codeblock. The truncated codeblocks are then concatenated together to form a single bitstream. The PCRD optimization, in effect, distributes the total rate for the image spatially across the codeblocks in a rate-distortion-optimal fashion such that codeblocks with higher energy, which tend to more heavily influence the distortion measure, tend to receive greater rate.

As described in the standard, JPEG2000 is, in essence, a 2D image coder. Although the standard does make a few provisions for multiband imagery such as hyperspectral data, the core coding procedure is based on within-band coding of 2D blocks as described above. Furthermore, the exact procedure employed for 3D imagery (e.g., the 3D wavelet transform and PCRD optimization across multiple bands) largely entails design issues for the encoder and thus lies outside the realm of the JPEG2000 standard, which covers only the decoder. Given the increasing prominence that JPEG2000 is garnering for the coding of hyperspectral imagery, we return to consider these encoder-centric issues in depth in Sec. 14.5. Finally, we note that JPEG2000 with truly 3D coding, consisting of AAC coding of 3D codeblocks as in [27], has been proposed as JPEG2000 Part 10 (JP3D), an extension to the core JPEG2000 standard. However, at the time of this writing, this proposed extension is in the preliminary stages of development, and currently, JPEG2000 for hyperspectral imagery is employed as discussed in Sec. 14.5.

14.4.4 Runlength Coding

Since, for a given significance threshold, the significance map is essentially a binary image, techniques that have long been employed for the coding of bilevel images are applicable. Specifically, *runlength coding* is the fundamental compression algorithm behind the Group 3 fax standard; the Wavelet Difference Reduction (WDR) [28] algorithm combines runlength coding of the significance map with an efficient

B_j and V_j subbands			H_j subbands			D_j subbands		
$\sum h_i$	$\sum v_i$	$\sum d_i$	$\sum h_i$	$\sum v_i$	$\sum d_i$	$\sum(h_i + v_i)$	$\sum d_i$	Context
2				2			≥ 3	8
1	≥ 1		≥ 1	1		≥ 1	2	7
1	0	≥ 1	0	1	≥ 1	0	2	6
1	0	0	0	1	0	≥ 2	1	5
0	2		2	0		1	1	4
0	1		1	0		0	1	3
0	0	≥ 2	0	0	≥ 2	≥ 2	0	2
0	0	1	0	0	1	1	0	1
0	0	0	0	0	0	0	0	0

Figure 14.16. The AAC contexts for JPEG2000.

lossless representation of the runlength symbols to produce an embedded image coder. Originally developed for 2D imagery in [28], WDR was extended to 3D as an implementation in QccPack [29]; this 3D extension merely deploys the runlength scanning as a 3D raster scan of each subband of the 3D DWT, which is easily accomplished in either dyadic or packet DWT decompositions.

14.4.5 Density Estimation

An all-together different approach to significance-map coding was proposed in [30] wherein an explicit estimate of the probability of significance of wavelet coefficients is used to code the significance map. Specifically, the significance state of a set of coefficients for a given threshold is coded via a raster scan through the coefficients. For coding efficiency, an entropy coder codes the significance state for each coefficient, using the probability that the coefficient is significant as determined by the density-estimation procedure. The density estimate is in the form of a multidimensional convolution implemented as a sequence of 1D filtering operations coined *tarp filtering*. In [30], the tarp filtering procedure was originally developed for 2D image coding; 3D tarp, with the tarp-filtering procedure suitably extended to three dimensions, was proposed in [31, 32].

Of the various significance-map coding techniques considered in this section, conditional coding in the form of JPEG2000 has achieved the most widespread prominence for the coding of hyperspectral imagery. In the next section, we explore several issues concerning JPEG2000 encoding that lie outside the scope of the JPEG2000 standard yet yield significant impact on compression performance.

14.5 JPEG2000 ENCODING STRATEGIES

JPEG2000 is increasingly being considered for the coding of hyperspectral imagery as well as other types of volumetric data, such as medical imagery. JPEG2000 is attractive because of its proven state-of-the art performance for the compression of grayscale and color photographic imagery. However, its performance for hyperspectral compression can vary greatly depending on how the JPEG2000 encoder

handles multiple-band images, i.e., images with multiple spectral bands. In effect, the JPEG2000 standard specifies the syntax and semantics of the compressed bitstream and, consequently, the operation of the *decoder*. The exact architecture of the *encoder*, on the other hand, is left largely to the designer of the compression system.

In deploying JPEG2000 on hyperspectral imagery, there are two primary issues that must be considered in the implementation of the JPEG2000 encoder: 1) spectral decorrelation, and 2) rate allocation between spectral bands. The first issue arises due to the fact that there tends to exist significant correlation between consecutive bands in a hyperspectral image. As a consequence, spectral decorrelation, via a wavelet transform, yields significant performance improvement.

The second encoder-design issue—rate allocation between spectral bands—arises from the fact that, essentially, JPEG2000 is a 2D compression algorithm. Consequently, given a specific target rate of R bpppb, the JPEG2000 encoder must determine how to allocate this total rate appropriately between spectral bands. It is usually the case that certain bands have significantly higher energy than other bands and thus will weigh more heavily in distortion measures than the other, weaker-energy bands. Consequently, it is likely that the JPEG2000 encoder will need to allocate proportionally greater rate to the higher-energy bands in order to maximize distortion performance for a given total rate R. Below, we explore several rate-allocation strategies; we will find significant performance difference between these strategies later in experimental results in Sec. 14.6.2.

14.5.1 Spectral Decorrelation for Multiple-Component Images

The JPEG2000 standard allows for images with up to 16,385 spectral bands to be included in a single bitstream; however, the standard does not specify how these spectral bands should be encoded for best performance. Whereas Part I of the JPEG2000 standard [6] permits spectral decorrelation only in the case of three-band images (i.e., red-green-blue), Annexes I and N of Part II of the standard [7] make provisions for arbitrary spectral decorrelation, including wavelet transforms.

By applying a 1D wavelet transform spectrally, and then subsequently employing a 2D wavelet transform spatially, we effectively implement the wavelet-packet transform of Fig. 14.6. We note that many JPEG2000 implementations are not yet fully compliant with Part II of the standard. In this case, we can "simulate" the spectral decorrelation permitted under Part II by employing a 1D wavelet transform spectrally on each pixel in the scene before the image cube is sent to the Part-I-compliant JPEG2000 encoder. Such an external spectral transform as been used previously [32, 33] to implement a "2D spatial + 1D spectral" wavelet-packet transform with Part-I-compliant coders.

14.5.2 Rate-Allocation Strategies Across Multiple Image Components

The PCRD optimization procedure of EBCOT described in Sec. 14.4.3 produces a rate-distortion-optimal bitstream for a single-band image by optimally truncating the independent codeblock bitstreams from the spectral band. However, there are several ways that this single-band truncation procedure can be extended to the multiband case, and the resulting multiband truncation procedure, in effect,

dictates how the total rate available for coding the hyperspectral image is allocated between the individual spectral bands.

That is, for a multiple-band image, a JPEG2000 encoder will partition each spectral band into 2D codeblocks which are coded into independent bitstreams identically to the process used for single-band imagery. To assemble a final bitstream, these individual codeblock bitstreams are truncated and concatenated together. Although the method for codeblock-bitstream truncation is an implementation issue concerning only the encoder and is thus not covered by the JPEG2000 standard, it is highly likely that, any given multiband JPEG2000 encoder will perform PCRD optimization for *at least* the codeblocks originating from a single spectral band. How this truncation process is extended across the multiple bands may vary with encoder implementation. Below, we describe three possible multiband rate-allocation strategies. In the following, let a hyperspectral image volume X be composed of N bands X_i, i.e., $X = \{X_1, X_2, \ldots, X_N\}$. We code X with a total rate of R bpppb. Assume that $B_i = \texttt{JPEG2000\_Encode}(R_i, X_i)$ is a single-band JPEG2000 encoder that encodes spectral band X_i with rate R_i using PCRD optimization, producing a bitstream B_i.

JPEG2000-BIFR The most straightforward method of allocating rate between multiple spectral bands is to simply code each band independently and assign to each an identical rate. This JPEG2000 band-independent fixed-rate (JPEG2000-BIFR), strategy operates as follows, where the "$\circ$" operator denotes bitstream concatenation:

$\texttt{JPEG2000\_BIFR}(R, \{X_1, \ldots, X_N\})$
  $B = \emptyset$
  `for` $i = 1, 2, ..., N$
    B_i `=` $\texttt{JPEG2000\_Encode}(R, X_i)$
    B `=` $B \circ B_i$
  `return` B

JPEG2000-BIRA JPEG2000 band-independent rate allocation (JPEG2000-BIRA) also codes each band independently; however, rates are allocated explicitly so that more important bands are coded with higher rate, and less important bands are coded at a lower rate.

$\texttt{JPEG2000\_BIRA}(R, \{X_1, \ldots, X_N\})$
  $B = \emptyset$
  `for` $i = 1, 2, ..., N$
    $\sigma_i^2 = \text{variance}\,[X_i]$
  `for` $i = 1, 2, ..., N$
    $R_i = \dfrac{\log_2 \sigma_i}{\sum_{j=1}^{N} \log_2 \sigma_j} \cdot RN$
    B_i `=` $\texttt{JPEG2000\_Encode}(R_i, X_i)$
    B `=` $B \circ B_i$
  `return` B

The rates, R_i, are determined so that bands with larger variances (i.e., higher energy) are coded at a higher rate than those with lower variances, while the total rate for the entire volume is R. This approach is, in essence, an *ad-hoc* variant of classical optimal rate allocation for a set of quantizers based on log variances (chap. 8 of [34], [35]).

JPEG2000-MC The final approach, JPEG2000 multi-component (JPEG2000-MC), can be employed when the JPEG2000 encoder is capable of performing PCRD optimization across multiple bands. That is, all of the spectral bands are input to the

encoder which produces codeblock bitstreams for every codeblock in every subband of every spectral band. Then, PCRD optimal truncation is applied to all codeblock bitstreams from all bands simultaneously, rather than simply the codeblock bitstreams for a single band. In this way, the PCRD optimization performs to the maximum of its potential, implicitly allocating rate in a rate-distortion fashion, not only spatially within each spectral band, but also spectrally across the multiple bands.

14.6 COMPRESSION PERFORMANCE

All the datasets used in the experiments were collected by AVIRIS, an airborne hyperspectral sensor with data in 224 contiguous bands from 400 nm to 2500 nm. For the results here, we crop the first scene in each dataset to produce image cubes with dimensions of $512 \times 512 \times 224$. In all cases, unprocessed radiance data was used. All coders use the popular biorthogonal 9-7 wavelet [36] with symmetric extension as used extensively in image-compression applications, and a transform decomposition of 4 spatial and spectral levels is employed. All rate measurements are expressed in bits per pixel per band (bpppb). All JPEG2000 coding uses Kakadu[1] Version 4.3 with a quantization step size of 0.0000001. Since Kakadu is not yet fully compliant with Part II of the JPEG2000 standard, the spectral transform is applied externally as described in Sec. 14.2.1 and in [32, 33]. We note that the results below are selected from extensive empirical evaluations we have conducted; a more complete presentation of results is available in [37].

14.6.1 Performance of Dyadic and Packet Transforms

As was discussed in Sec.14.2.1, there are two contending transform arrangements for the 3D DWT. The 3D dyadic transform (Fig. 14.5) is a direct extension of the 2D dyadic transform in which we transform once in each direction and then further decompose the baseband. In the case of the 3D packet transform (Fig. 14.6), the coefficients in each spectral slice are transformed with a 2D dyadic transform, which is then followed by a spectral transform. Fig. 14.17 depicts the typical rate-distortion performance achieved by a coder using these two transform structures. We see that performance for the packet transform is greatly superior to that for the dyadic transform. As we have observed similar results for other coders and other datasets, we use the packet transform exclusively for all subsequent results.

14.6.2 Performance of JPEG2000 Encoding Strategies

Sec. 14.5 presents several strategies for the design of a JPEG2000 encoder. We evaluate these strategies now, focusing first on rate-distortion performance before considering POC performance as described in Sec. 14.3. In Fig. 14.18, we plot the rate-distortion performance of JPEG2000 for a range of rates, while in Table 14.1, distortion performance at a single rate is tabulated. In these results, techniques labeled as "2D" do not use any spectral transform (i.e., only 2D wavelet transforms are applied spatially), while the other techniques use the 3D wavelet-packet transform which includes a spectral transform. For each dataset, we present performance

[1] `http://www.kakadusoftware.com`

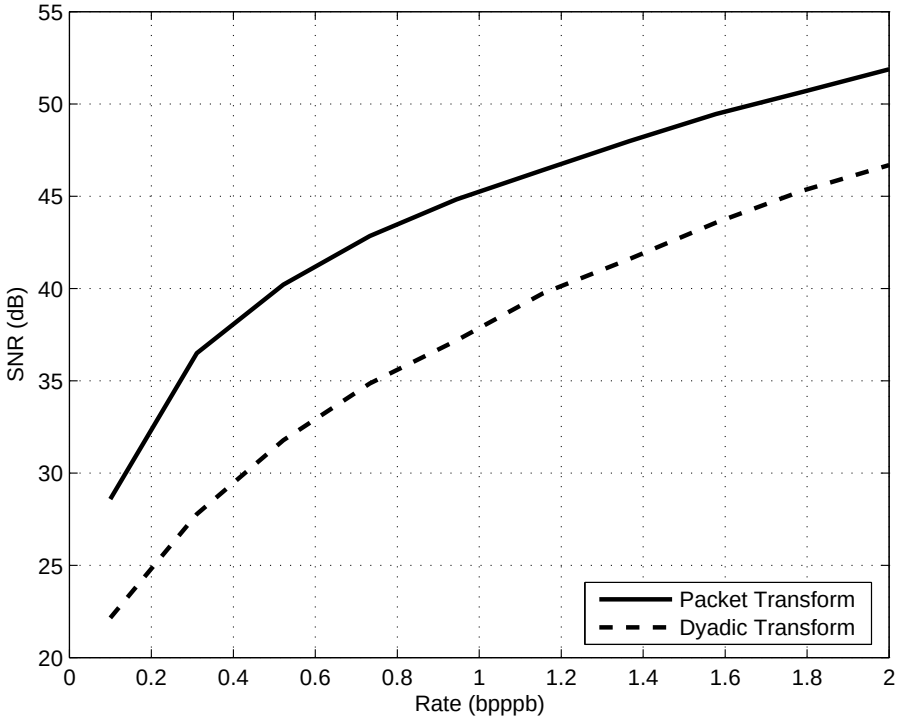

Figure 14.17. Comparison of the typical rate-distortion performance for the dyadic transform of Fig. 14.5 versus that of the packet transform of Fig. 14.6. This plot is for the Moffett image using 3D-SPIHT.

for the three rate-allocation techniques described in Sec. 14.5.2, both with and without the spectral-decorrelation transform. With the exception of JPEG2000-BIFR, all the rate-allocation techniques perform significantly better when a spectral transform is performed. We see that JPEG2000-MC substantially outperforms the other techniques by at least 5–10 dB.

We now turn our attention to POC performance to gauge the preservation of unsupervised-classification performance. We see that the POC performances in Table 14.2 correlate well with SNR figures of Table 14.1 in that, if one technique outperforms another in the rate-distortion realm, then it will mostly likely have higher POC performance as well. As expected, JPEG2000-MC performs substantially better than the other techniques in terms of POC.

We note that both Kakadu Version 4.3 and the JPEG2000 encoder in ENVI Version 4.1 (which uses the Kakadu coder) implement JPEG2000-MC rate allocation, yet neither support the use of a spectral transform since they are not fully compliant with Part II of the JPEG2000 standard. Thus, the performance of these coders is equivalent to that of the 2D JPEG2000-MC approach considered here. As our results indicate, adding a spectral transform would significantly enhance the performance of these coders.

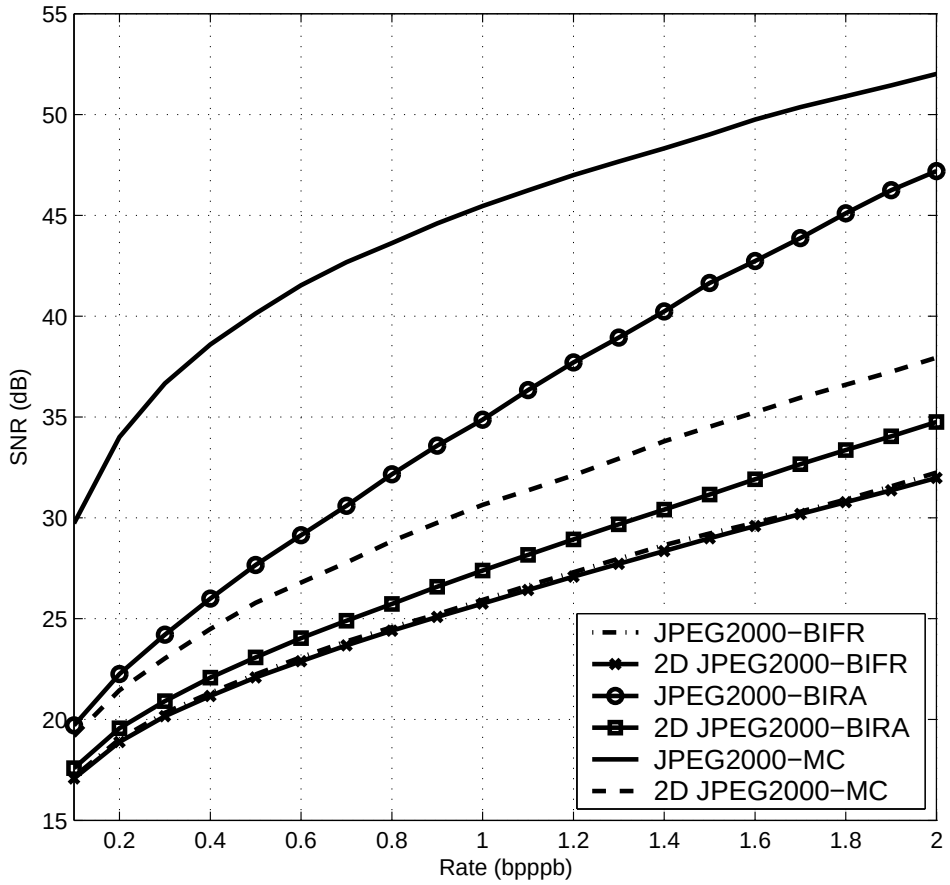

Figure 14.18. Rate-distortion performance for Moffett for the JPEG2000 encoding strategies.

Table 14.1. SNR Performance in dB at 1.0 bpppb for the JPEG2000 Encoding Strategies

Dataset	2D BIFR	BIFR	2D BIRA	BIRA	2D MC	MC
Moffett	25.8	25.9	27.4	34.9	30.6	**45.5**
Jasper Ridge	24.0	23.8	25.7	33.4	29.8	**44.8**
Cuprite	32.9	32.8	34.9	42.6	38.3	**51.0**

14.6.3 Algorithm Performance

Rate-distortion performance for a variety of the algorithms described in Sec. 14.4 (3D-WDR, 3D-tarp, 3D-SPECK, 3D-SPIHT, and JPEG2000-MC) is shown in Figs. 14.19–14.21 as well as in Table 14.3. In these results, we see that all five techniques provide largely similar rate-distortion performance for the datasets considered, with JPEG2000-MC usually slightly outperforming the others. Similar conclusions are drawn from the POC results of Table 14.4.

14.7 SUMMARY

In this chapter, we overviewed the major concepts in 3D embedded wavelet-based compression for hyperspectral imagery. We reviewed several popular compression techniques that have been considered for the coding of hyperspectral imagery, focusing on the primary difference between techniques—how the significance map is coded. We found that the different techniques offered performance that is roughly similar, both in terms of rate-distortion performance as well as a more application-specific preservation of performance at unsupervised classification. We discussed that the most prominent of the algorithms considered, JPEG2000, is subject to an international standard that covers only the decoder, leaving many design details regarding the encoder unspecified, particularly as pertaining to the coding of multiband imagery. We presented experimental results that demonstrate how a JPEG2000 encoder allocates rate between spectral bands substantially affects performance. Additionally, we saw that JPEG2000 performance almost always benefits greatly from the application of a 1D spectral wavelet transform to remove correlation in the spectral direction.

As a final note, we observe that, in many situations, it may be necessary to store hyperspectral datasets in their original state, i.e., without any compression loss. Such archival applications may necessitate the *lossless* compression of hyperspectral imagery, whereas the discussion in this chapter has focused exclusively on *lossy* compression algorithms. However, it is fairly straightforward to modify the lossy algorithms considered here to render them lossless, while still preserving their progressive-transmission capability. Such embedded wavelet-based coders then provide "lossy-to-lossless" performance, such that any truncation of the bitstream can be reconstructed to a lossy representation, yet, if the entire bitstream is decoded, a lossless reconstruction of the original dataset is obtained. Such lossy-to-lossless coding has been proposed in several contexts [38, 39], including hyperspectral-image compression [40], by adding an integer-to-integer wavelet transform [41] to a lossy technique. JPEG2000 supports such lossy-to-lossless coding in Part I of the standard in exactly this way.

REFERENCES

1. S. Gupta and A. Gersho, "Feature predictive vector quantization of multispectral images," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 30, no. 3, pp. 491–501, May 1992.
2. S.-E. Qian, A. B. Hollinger, S. Williams, and D. Manak, "Vector quantization using spectral index-based multiple subcodebooks for hyperspectral data compression,"

Table 14.2. POC Performance at 1.0 bpppb for the JPEG2000 Encoding Strategies

	ISODATA POC (%)					
Dataset	2D BIFR	BIFR	2D BIRA	BIRA	2D MC	MC
Moffett	83.4	94.5	86.6	94.5	93.2	**99.7**
Jasper Ridge	77.3	75.5	82.2	93.7	93.9	**99.7**
Cuprite	80.3	78.1	85.1	94.7	94.7	**99.8**
	k-means POC (%)					
Moffett	75.4	73.2	79.9	91.7	89.8	**99.6**
Jasper Ridge	67.2	64.7	73.9	90.4	91.0	**99.5**
Cuprite	71.3	68.3	77.6	92.2	92.1	**99.6**

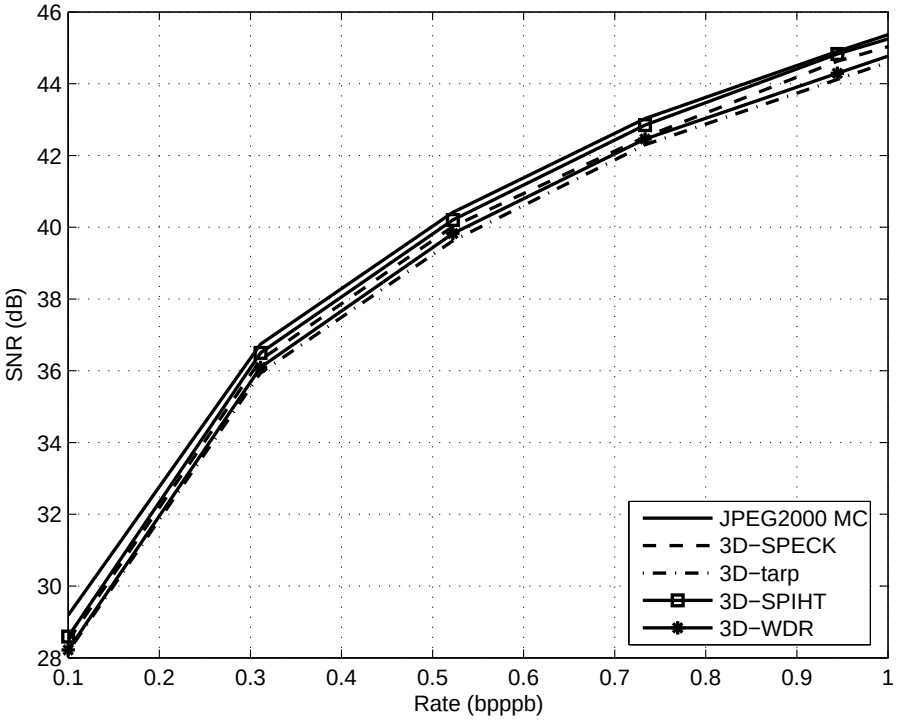

Figure 14.19. Rate-distortion performance for Moffett.

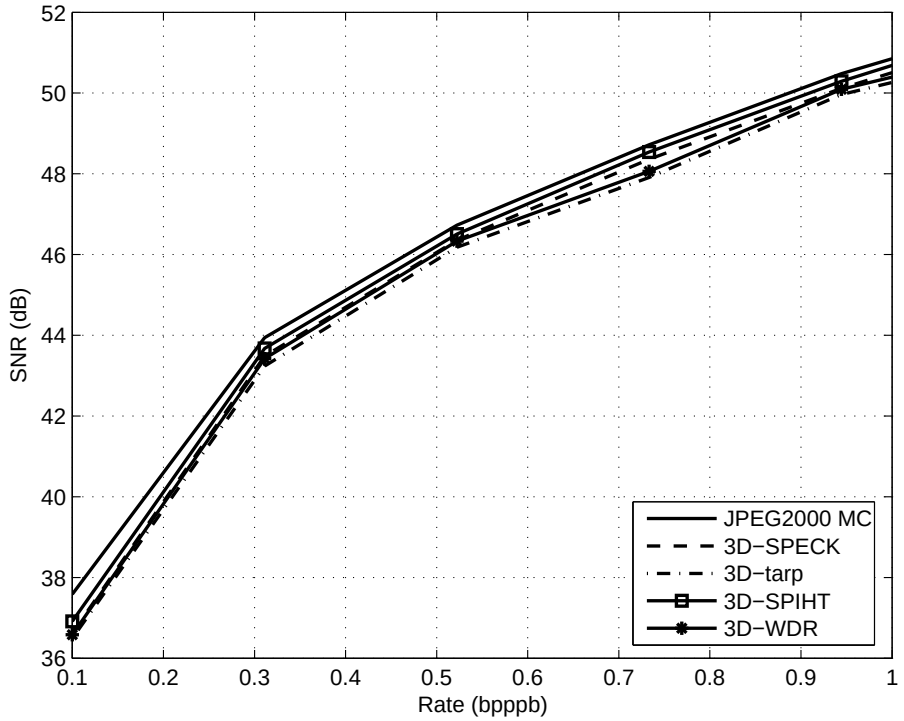

Figure 14.20. Rate-distortion performance for Cuprite.

Table 14.3. SNR at 1.0 bpppb

	SNR (dB)				
Dataset	JPEG2000	SPECK	SPIHT	TARP	WDR
Moffett	**45.4**	45.1	45.3	44.5	44.7
Jasper Ridge	**44.9**	44.4	44.7	43.7	44.2
Cuprite	**50.8**	50.5	50.7	50.3	50.4
Low Altitude	**27.6**	27.3	27.4	25.2	27.1
Lunar Lake	**46.4**	45.9	46.1	43.7	45.9

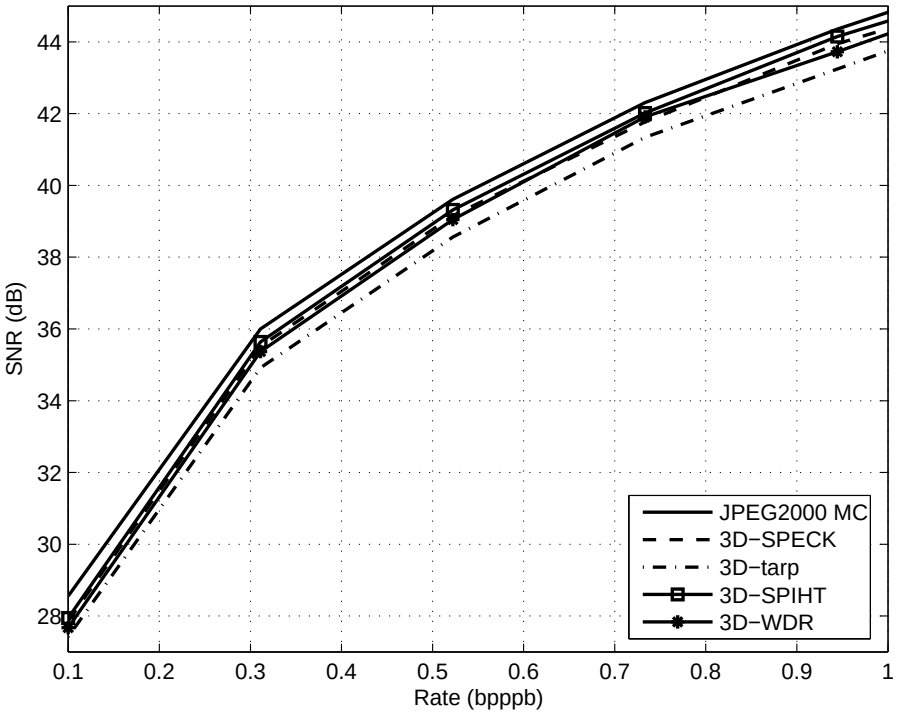

Figure 14.21. Rate-distortion performance for Jasper Ridge.

Table 14.4. POC Performance at 1.0 bpppb

	ISODATA POC (%)				
Dataset	JPEG2000	SPECK	SPIHT	TARP	WDR
Moffett	**99.8**	99.7	99.7	99.7	99.7
Jasper Ridge	**99.8**	99.7	99.7	99.7	99.7
Cuprite	**99.8**	**99.8**	**99.8**	**99.8**	**99.8**
Low Altitude	97.9	**98.1**	97.9	97.3	97.9
Lunar Lake	**99.7**	99.5	**99.7**	**99.7**	**99.7**
	k-Means POC (%)				
Moffett	**99.7**	99.6	99.6	99.6	99.6
Jasper Ridge	**99.6**	99.5	99.5	99.5	99.5
Cuprite	**99.7**	**99.7**	99.6	**99.7**	**99.7**
Low Altitude	96.6	**96.8**	96.6	96.1	96.7
Lunar Lake	99.5	**99.6**	99.5	99.2	**99.6**

IEEE Transactions on Geoscience and Remote Sensing, vol. 38, no. 3, pp. 1183–1190, May 2000.

3. B. R. Epstein, R. Hingorani, J. M. Shapiro, and M. Czigler, "Multispectral KLT-wavelet data compression for Landsat thematic mapper images," in *Proceedings of the IEEE Data Compression Conference*, J. A. Storer and M. Cohn, Eds., Snowbird, UT, March 1992, pp. 200–208.
4. J. A. Saghri, A. G. Tescher, and J. T. Reagan, "Practical transform coding of multispectral imagery," *IEEE Signal Processing Magazine*, vol. 12, no. 1, pp. 32–43, January 1995.
5. G. P. Abousleman, M. W. Marcellin, and B. R. Hunt, "Compression of hyperspectral imagery using the 3-D DCT and hybird DPCM/DCT," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 33, no. 1, pp. 26–34, January 1995.
6. *Information Technology—JPEG 2000 Image Coding System—Part 1: Core Coding System*, ISO/IEC 15444-1, 2000.
7. *Information Technology—JPEG 2000 Image Coding System—Part 2: Extensions*, ISO/IEC 15444-2, 2004.
8. D. S. Taubman and M. W. Marcellin, *JPEG2000: Image Compression Fundamentals, Standards and Practice*. Boston, MA: Kluwer Academic Publishers, 2002.
9. B. Penna, T. Tillo, E. Magli, and G. Olmo, "Progressive 3-D coding of hyperspectral images based on JPEG 2000," *IEEE Geoscience and Remote Sensing Letters*, vol. 3, no. 1, pp. 125–129, January 2006.
10. D. Taubman, "High performance scalable image compression with EBCOT," *IEEE Transactions on Image Processing*, vol. 9, no. 7, pp. 1158–1170, July 2000.
11. A. Deever and S. S. Hemami, "What's your sign?: Efficient sign coding for embedded wavelet image coding," in *Proceedings of the IEEE Data Compression Conference*, J. A. Storer and M. Cohn, Eds., Snowbird, UT, March 2000, pp. 273–282.
12. A. T. Deever and S. S. Hemami, "Efficient sign coding and estimation of zero-quantized coefficients in embedded wavelet image codecs," *IEEE Transactions on Image Processing*, vol. 12, no. 4, pp. 420–430, April 2003.
13. I. H. Witten, R. M. Neal, and J. G. Cleary, "Arithmetic coding for data compression," *Communications of the ACM*, vol. 30, no. 6, pp. 520–540, June 1987.
14. J. M. Shapiro, "Embedded image coding using zerotrees of wavelet coefficients," *IEEE Transactions on Signal Processing*, vol. 41, no. 12, pp. 3445–3462, December 1993.
15. A. Said and W. A. Pearlman, "A new, fast, and efficient image codec based on set partitioning in hierarchical trees," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 6, no. 3, pp. 243–250, June 1996.
16. B.-J. Kim and W. A. Pearlman, "An embedded wavelet video coder using three-dimensional set partitioning in hierarchical trees (SPIHT)," in *Proceedings of the IEEE Data Compression Conference*, J. A. Storer and M. Cohn, Eds., Snowbird, UT, March 1997, pp. 251–257.
17. P. L. Dragotti, G. Poggi, and A. R. P. Ragozini, "Compression of multispectral images by three-dimensional SPIHT algorithm," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 38, no. 1, pp. 416–428, January 2000.
18. B.-J. Kim, Z. Xiong, and W. A. Pearlman, "Low bit-rate scalable video coding with 3-D set partitioning in hierarchical trees (3-D SPIHT)," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 10, no. 8, pp. 1374–1387, December 2000.
19. C. He, J. Dong, Y. F. Zheng, and Z. Gao, "Optimal 3-D coefficient tree structure for 3-D wavelet video coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 10, pp. 961–972, October 2003.

20. S. Cho and W. A. Pearlman, "Error resilient video coding with improved 3-D SPIHT and error concealment," in *Image and Video Communications and Processing*, B. Vasudev, T. R. Hsing, and A. G. Tescher, Eds. Santa Clara, CA: Proc. SPIE 5022, January 2003, pp. 125–136.

21. X. Tang, S. Cho, and W. A. Pearlman, "3D set partitioning coding methods in hyperspectral image compression," in *Proceedings of the International Conference on Image Processing*, vol. 2, Barcelona, Spain, September 2003, pp. 239–242.

22. A. Islam and W. A. Pearlman, "An embedded and efficient low-complexity hierarchical image coder," in *Visual Communications and Image Processing*, K. Aizawa, R. L. Stevenson, and Y.-Q. Zhang, Eds. San Jose, CA: Proc. SPIE 3653, January 1999, pp. 294–305.

23. W. A. Pearlman, A. Islam, N. Nagaraj, and A. Said, "Efficient, low-complexity image coding with a set-partitioning embedded block coder," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 14, no. 11, pp. 1219–1235, November 2004.

24. X. Tang, W. A. Pearlman, and J. W. Modestino, "Hyperspectral image compression using three-dimensional wavelet coding," in *Image and Video Communications and Processing*, B. Vasudev, T. R. Hsing, A. G. Tescher, and T. Ebrahimi, Eds. Santa Clara, CA: Proc. SPIE 5022, January 2003, pp. 1037–1047.

25. X. Tang and W. A. Pearlman, "Three-dimensional wavelet-based compression of hyperspectral images," in *Hyperspectral Data Compression*, G. Motta, F. Rizzo, and J. A. Storer, Eds. Norwell, MA: Kluwer Academic Publishers, 2006, ch. 10, pp. 273–308.

26. M. W. Marcellin and A. Bilgin, "Quantifying the parent-child coding gain in zerotree-based coders," *IEEE Signal Processing Letters*, vol. 8, no. 3, pp. 67–69, March 2001.

27. P. Schelkens, J. Barbarien, and J. Cornelis, "Compression of volumetric medical data based on cube-splitting," in *Applications of Digital Image Processing XXII*. San Diego, CA: Proc. SPIE 4115, August 2000, pp. 91–101.

28. J. Tian and R. Wells, Jr., "Embedded image coding using wavelet difference reduction," in *Wavelet Image and Video Compression*, P. N. Topiwala, Ed. Boston, MA: Kluwer Academic Publishers, 1998, ch. 17, pp. 289–301.

29. J. E. Fowler, "QccPack: An open-source software library for quantization, compression, and coding," in *Applications of Digital Image Processing XXIII*, A. G. Tescher, Ed. San Diego, CA: Proc. SPIE 4115, August 2000, pp. 294–301.

30. P. Simard, D. Steinkraus, and H. Malvar, "On-line adaptation in image coding with a 2-D tarp filter," in *Proceedings of the IEEE Data Compression Conference*, J. A. Storer and M. Cohn, Eds., Snowbird, UT, April 2002, pp. 23–32.

31. Y. Wang, J. T. Rucker, and J. E. Fowler, "Embedded wavelet-based compression of hyperspectral imagery using tarp coding," in *Proceedings of the International Geoscience and Remote Sensing Symposium*, vol. 3, Toulouse, France, July 2003, pp. 2027–2029.

32. ——, "3D tarp coding for the compression of hyperspectral images," *IEEE Geoscience and Remote Sensing Letters*, vol. 1, no. 2, pp. 136–140, April 2004.

33. H. S. Lee, N. H. Younan, and R. L. King, "Hyperspectral image cube compression combining JPEG-2000 and spectral decorrelation," in *Proceedings of the International Geoscience and Remote Sensing Symposium*, vol. 6, Toronto, Canada, June 2002, pp. 3317–3319.

34. A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression*. Norwell, MA: Kluwer Academic Publishers, 1992.

35. J. J. Y. Huang and P. M. Schultheiss, "Block quantization of correlated Gaussian random vectors," *IEEE Transactions on Communications*, vol. 11, no. 3, pp. 289–296, September 1963.

36. M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies, "Image coding using wavelet transform," *IEEE Transactions on Image Processing*, vol. 1, no. 2, pp. 205–220, April 1992.

37. J. T. Rucker, "3D wavelet-based algorithms for the compression of geoscience data," Master's thesis, Mississippi State University, 2005.

38. A. Bilgin, G. Zweig, and M. W. Marcellin, "Three-dimensional image compression with integer wavelet transforms," *Applied Optics*, vol. 39, no. 11, pp. 1799–1814, April 2000.

39. Z. Xiong, X. Wu, S. Cheng, and J. Hua, "Lossy-to-lossless compression of medical volumetric data using three-dimensional integer wavelet transforms," *IEEE Transactions on Medical Imaging*, vol. 22, no. 3, pp. 459–470, March 2003.

40. X. Tang and W. A. Pearlman, "Lossy-to-lossless block-based compression of hyperspectral volumetric data," in *Proceedings of the International Conference on Image Processing*, vol. 3, Singapore, October 2004, pp. 3283–3286.

41. A. R. Calderbank, I. Daubechies, W. Sweldens, and B.-L. Yeo, "Lossless image compression using integer to integer wavelet transforms," in *Proceedings of the International Conference on Image Processing*, vol. 1, Lausanne, Switzerland, September 1997, pp. 596–599.