JPEG2000: Standard for Interactive Imaging

DAVID S. TAUBMAN AND MICHAEL W. MARCELLIN, FELLOW, IEEE

Contributed Paper

JPEG2000 is the latest image compression standard to emerge from the Joint Photographic Experts Group (JPEG) working under the auspices of the International Standards Organization. Although the new standard does offer superior compression performance to JPEG, JPEG2000 provides a whole new way of interacting with compressed imagery in a scalable and interoperable fashion. This paper provides a tutorial-style review of the new standard, explaining the technology on which it is based and drawing comparisons with JPEG and other compression standards. The paper also describes new work, exploiting the capabilities of JPEG2000 in client–server systems for efficient interactive browsing of images over the internet.

Keywords*—*Coding, compression standards, image compression, interactive imaging, JPEG2000, remote image browsing, scalability, wavelets.

I. INTRODUCTION

JPEG2000 [1]–[6] is the latest image compression standard to emerge from the body popularly known as the Joint Photographic Experts Group (JPEG). More formally, this body is denoted ISO/IEC JTC1/SC29/WG1, which stands for Working Group 1 of Study Committee 29 of Joint Technical Committee 1 of ISO/IEC. Here, ISO is the International Organization for Standardization, IEC is the International Electrotechnical Commission, and the word "Joint" refers to the fact that the standard is developed and published jointly with the International Telecommunication Union (ITU).

This new standard has been developed to meet the demand for efficient, flexible, and interactive image representations. JPEG2000 is much more than a compression algorithm, opening up new paradigms for interacting with digital imagery. At the same time, the features offered by JPEG2000 derive from a single algorithm rather than a family of different algorithms. In particular, an important goal of JPEG2000 is that all implementations, from the simplest to the most sophisticated, should be able to effectively interact with the same, efficiently compressed image, regardless of the resolution, bit depth, bit rate, or number of components in that image. With few exceptions, this goal has been achieved. This is a direct consequence of JPEG2000's emphasis on scalable compressed representations, as described below.

In addition to highly scalable compressed data streams, JPEG2000 offers numerous advantages over its predecessor, JPEG [7]–[9]. Among these are

- improved compression efficiency;
- progressive lossy to lossless performance within a single data stream;
- the ability to resequence compressed data to suit a wide range of different applications;
- the ability to arbitrarily crop images in the compressed domain without compression noise buildup;
- the ability to enhance the quality associated with selected spatial regions in selected "quality layers";
- the ability to work with truly enormous images without breaking them into independently compressed "tiles."

Section VI of this paper is devoted to a more careful comparison of JPEG with JPEG2000. We also take the opportunity to point out key differences between the two standards at various other junctures in the technical discussion which follows.

A. Scalability

As already suggested, a central concept in JPEG2000 is that of scalability. In the image compression literature, scalability means much more than the ability to compress images with a variety of different sizes at a variety of different bit rates. Instead, scalability is a property of the compressed data stream itself. A compressed data stream is said to be scalable if it consists of an embedded collection of smaller streams, each representing an efficient compression of the original image or some portion thereof.

More specifically, a compressed representation is said to be "resolution scalable" if it contains identifiable subsets which represent successively lower resolution versions of the source image; it is "distortion scalable" (or SNR scalable)

Manuscript received December 7, 2001; revised April 18, 2002.

D. S. Taubman is with the School of Electrical Engineering and Telecommunications, The University of New South Wales, UNSW Sydney 2052, Australia (e-mail: d.taubman@unsw.edu.au).

M. W. Marcellin is with the Department of Electrical and Computer Engineering, The University of Arizona, Tucson, AZ 85721 USA (e-mail: marcellin@ece.arizona.edu).

Publisher Item Identifier 10.1109/JPROC.2002.800725.

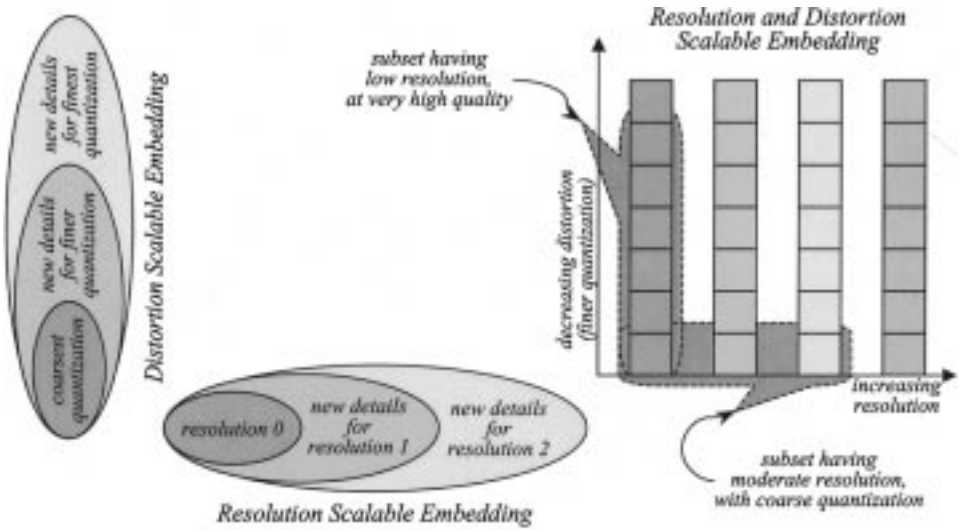

Fig. 1. One- and two-dimensional embedded compressed data streams with dimensions of resolution (image size) and distortion (image quality).

if it contains identifiable subsets which represent the image at full resolution but with successively lower quality (more distortion). A compressed data stream which possesses both forms of scalability contains subsets which represent the image at any of a number of different resolutions, each at any of a number of different qualities. The embedded components of a resolution and distortion scalable data stream are illustrated stylistically in Fig. 1.

The value of scalability may be summarized as the ability to *compress once but decompress in many ways.* At the point of compression, there is no need to know the resolution and quality which will be required by a consumer. The complete data stream, as well as each of its subsets, represents the image as efficiently as could be expected if we had known the consumer's requirements at the point of compression. In addition to resolution and distortion, JPEG2000 compressed images offer spatial accessibility and component accessibility. Starting with a single compressed data stream, it is possible to extract a subset (itself a valid JPEG2000 compressed data stream) which represents a selected spatial region of interest within one or more selected image components[1] at a selected resolution and with a selected quality. We may think of the spatial region, the image components, the resolution, and the distortion as constituting four "dimensions" of scalability. Fig. 1 shows only two of these four dimensions.

Scalable representations have obvious benefits for applications where the image must be distributed to multiple clients with different display resolutions, regions of interest, or communication capacities. Alternatively, a single client may access the compressed image interactively, where the spatial region, resolution, and image quality may change dynamically with the user's interest or the properties of the communication channel. Such applications are discussed in Section V of this paper.

[1]Individual color channels, such as luminance (Y) and chrominance (Cb and Cr), are compressed as separate image components. Image components may also play more exotic roles as layers (text, graphics, etc.) in a compound document or as slices in MRI or CT data sets.

Scalable representations also provide elegant solutions to a number of problems which regularly arise in the deployment of media compression algorithms. If the compressed file must conform to a given length constraint, a single compressed data stream can simply be truncated to the desired size, achieving comparable compression efficiency to the more traditional approach of repeatedly modifying quantization parameters until the desired size is achieved.

Scalability largely obviates the need to impose limits on the image dimensions, compressed bit rate, and number of colors. The reader may be familiar with the use of such limits in the MPEG family of standards where they serve to bound the complexity of compliant decoders. In JPEG2000, however, the decoder is free to decompress a reduced size image, possibly at reduced quality, in accordance with its advertised capabilities. This in turn means that content providers are free to include as much information as they wish in a single compressed data stream. Only a fraction of this information might be exploited by any given application, but content providers need not make prejudicial assumptions concerning the information which a potentially unknown application might be able to exploit.

B. Technology Overview

Scalability in JPEG2000 is based on the discrete wavelet transform (DWT) and embedded block coding with optimized truncation (EBCOT). It is appropriate at this point to provide a cursory introduction to these concepts. A more detailed exposition follows in Sections II and III. The reader may find Figs. 2–4 helpful in understanding the discussion which follows.

If multiple image components are present, they may optionally be subjected to a decorrelating "color" transform. JPEG2000 Part 1 describes two such transforms, one of which is the conventional RGB to YCbCr transform: the Y component represents image luminance and may be independently decompressed to recover a suitable gray-scale rendition of the image; Cb and Cr represent blue and red

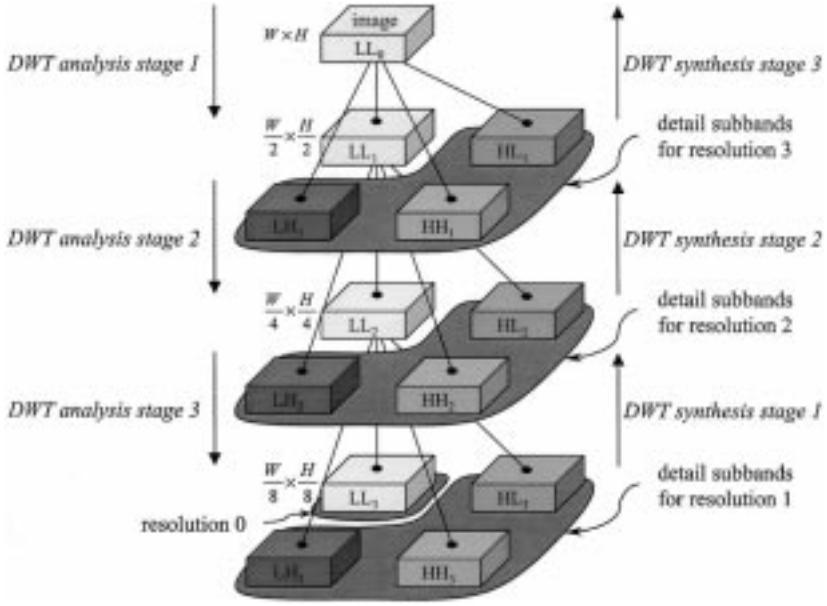

Fig. 2. DWT analysis and synthesis, showing three levels of decomposition. The dimensions of the image (LL_d) are halved between successive stages in the transform, d.

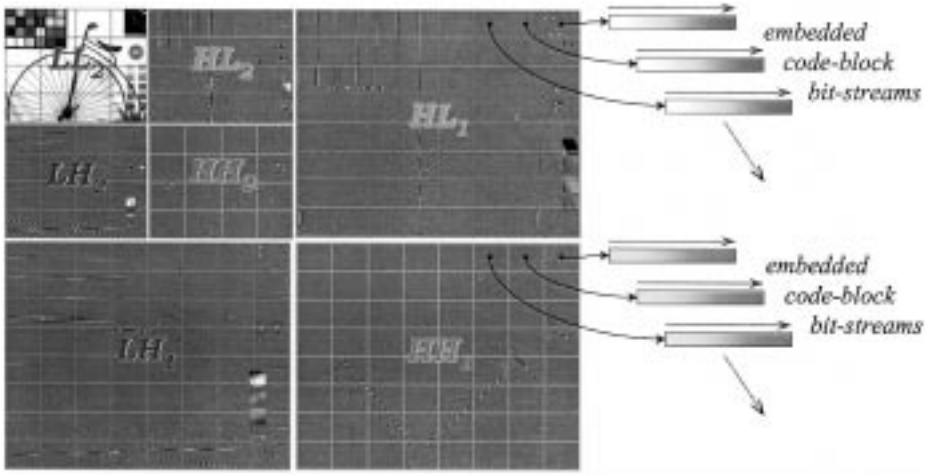

Fig. 3. Partition of image subbands into code-blocks, each with its own embedded bit-stream.
Original image source belongs to the standard color image data (SCID) described in ISO 12640, 1995.

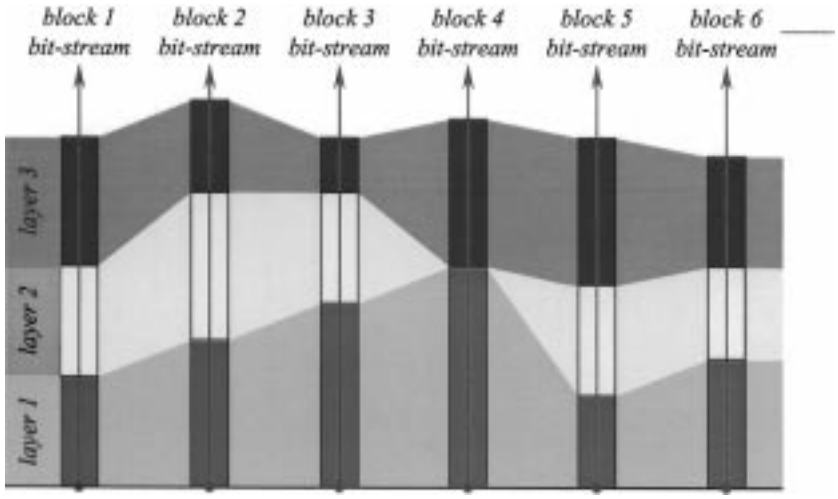

Fig. 4. Quality layer abstraction in JPEG2000.

color difference components, respectively. While the YCbCr transform is invertible, it is not amenable to efficient lossless compression. A second decorrelating transform known as the reversible color transform (RCT) [10] is defined by JPEG2000 Part 1 for applications requiring lossless compression. It has similar properties to the YCbCr transform, but produces integer-valued luminance and chrominance components with similar dynamic range to the original image samples. These two decorrelating transforms may be applied to the first three components of any image, regardless of whether or not the components have an interpretation in terms of color. In any event, the output components are compressed independently.

For the sake of simplicity, we henceforth restrict our discussion to the compression of a single image component, using the term "image" to refer to that component. We also temporarily ignore the fact that images may be further subdivided into tiles prior to compression, a matter which we take up again in Section IV-A. As shown in Fig. 2, the image is first decomposed (DWT analysis) into a collection of subsampled spatial frequency bands known as subbands. The subbands belong to a multiresolution hierarchy, from which each successively higher resolution version of the image may be reconstructed by composing the immediate lower resolution version (denoted LL_d) with three spatial detail subbands (denoted LH_d, HL_d and HH_d). The composition operator is identified as "DWT synthesis," and the image dimensions double between successive synthesis stages.

The samples describing each subband are partitioned into rectangular blocks, known as "code-blocks," each of which is independently coded into a finely embedded bitstream, as suggested by Fig. 3. Truncating the embedded bitstream associated with any given code-block has the effect of quantizing the samples in that block more coarsely. By *finely* embedded, we mean that each block's bitstream offers numerous useful truncation points, with rate-distortion characteristics comparable to those which could be obtained by adjusting the quantization parameters associated with an efficient nonembedded coding scheme. Each block of each subband in each image component may be independently truncated to any desired length after the coding is complete.

Resolution scalability in JPEG2000 is a direct consequence of the multiresolution properties of the DWT. By dropping the code-blocks corresponding to the highest resolution detail subbands and omitting the final stage of DWT synthesis, a half resolution image is reconstructed from the remaining subbands. Dropping the next lower resolution subbands leaves a quarter resolution image, and so forth.

Even though each individual code-block has its own embedded bitstream, this is not sufficient to ensure that the overall compressed representation is distortion scalable. To obtain an efficient compressed representation at a lower overall bit rate, it is necessary to know how each code-block's bitstream should be truncated in relation to the others. JPEG2000 provides this information through a "quality layer" abstraction. As illustrated in Fig. 4, each quality layer represents an incremental contribution (possibly empty) from the embedded bitstream associated with each code-block in the image. The sizes of these incremental layer contributions are determined (optimized) during compression in a manner which ensures that any leading set of quality layers corresponds to an efficient compressed representation of the original image.

Distortion scalability is realized by discarding one or more final quality layers. It is also possible, although somewhat suboptimal, to discard partial layers. If the compressed data stream is organized in layer progressive fashion,[2] all of the code-block contributions to layer 1 appear first, followed by the contributions to layer 2, and so forth. Such streams may be truncated at any point and meaningfully reconstructed to yield a decompressed image whose quality is similar to that which could be achieved from a nonprogressive stream of the same size. It is important to note that the JPEG2000 standard imposes no practical restriction on the way in which a compressor may assign code-block contributions to quality layers. If a particular region of the image or particular frequency bands are known to be more important than others, the relevant code-blocks may be assigned higher priority, making larger contributions to the initial quality layers. In this way, a decompressor which receives only the first few quality layers will reconstruct the most important portions of the image with higher fidelity. The quality layer abstraction thus allows distortion scalability to be applied to domain-specific measures of distortion (or importance).

Spatial accessibility in JPEG2000 arises from the fact that each code-block is associated with a limited spatial region and is coded independently. Typical code-block dimensions are 32×32 or 64×64 subband samples. The size of the reconstructed image region which is affected by any given code-block depends upon the particular subband to which it belongs. Also, adjacent code-blocks from any given subband have overlapping regions of influence in the reconstructed image. This is because wavelet synthesis is a spatially expansive operation. This property tends to blur the boundaries between code-blocks in the reconstructed image, avoiding the appearance of hard boundary artifacts when individual block bitstreams are aggressively truncated.

II. Multiresolution Transforms

Multiresolution transforms are the key to resolution scalable compression. Such transforms are typically based around either a Laplacian pyramid structure or a tree-structured subband structure, also called a discrete wavelet transform. The former is used by JPEG in its "hierarchical refinement" mode, while the latter is foundational to JPEG2000.

A. Laplacian Pyramids

Fig. 5 illustrates the Laplacian pyramid paradigm for resolution-scalable image compression. A resolution "reduction" operator is used to construct a family of successively lower resolution images, each having half the width and height of its predecessor. The lowest resolution image, at the base of the pyramid, is passed through a (usually lossy) compression

[2]JPEG2000 supports a rich family of information progression orders, some of which are discussed further in Section IV-B.

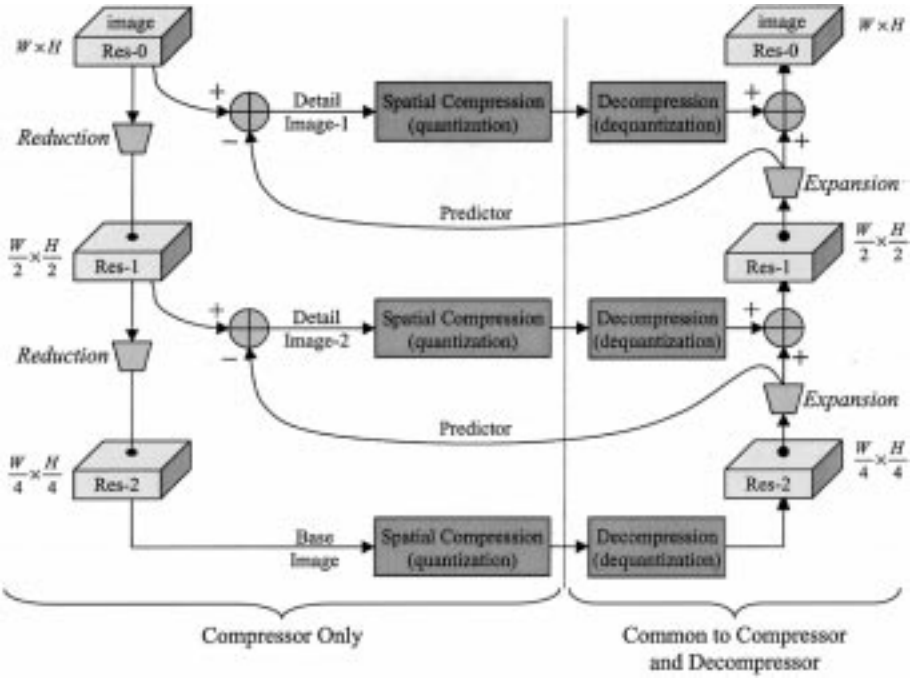

Fig. 5. Multiresolution compression using the Laplacian pyramid structure.

algorithm. After decompression, a resolution "expansion" operator is used to form a "predictor" for the next higher resolution image. The prediction residual, or "detail image," is again spatially compressed. After decompression and addition of the predictor, the second lowest resolution in the pyramid is recovered (with some distortion).

In this way, the entire multiresolution family is represented by the base image, together with a sequence of detailed images. JPEG's "hierarchical refinement" mode operates in exactly this way, with the base image and each detail image compressed using the baseline JPEG algorithm. Evidently, a reduced resolution may be obtained simply by discarding one or more of the detail images from the compressed data stream.

Perhaps the most significant drawback of the Laplacian pyramid is its redundancy. The largest detail image has as many samples as the original image. The next detail image has $1/4$ as many samples as the original image, and so forth. In this way, the total number of samples which must be compressed is larger than the original number of image samples by a factor of $1 + (1/4) + (1/16) + \cdots = 4/3$. This redundancy works against efficient compression.

A second drawback of the multiresolution structure shown in Fig. 5 is that the compressor and decompressor are expected to form exactly the same predictors from the decompressed base and detail image components. This poses a problem if resolution and distortion scalability are both required, since the amount of distortion in the base and detail image components of a distortion scalable representation depends upon the portion of the data stream which is actually decompressed—something the compressor cannot know *a priori*. Although it is possible to relax the requirement that the compressor and decompressor form identical predictors, this leads to further reduction in compression efficiency.

B. Subband Transforms

JPEG uses the discrete cosine transform (DCT) to exploit spatial redundancy, adding hierarchical refinement as an optional mode. By contrast, the DWT at the heart of JPEG2000 serves both to exploit spatial redundancy and to impart resolution scalability.

The DWT belongs to the more general class of tree-structured subband transforms, which are constructed from the one-dimensional (1-D) building block illustrated in Fig. 6. An input sequence $x[n]$ is passed through low- and high-pass analysis filters, with impulse responses $h_0[n]$ and $h_1[n]$, respectively. The filtered outputs $y_0'[n]$ and $y_1'[n]$ are each subsampled by a factor of 2, yielding low- and high-pass subband sequences. In particular, the low-pass subband is formed by retaining even indexed samples, $y_0[k] = y_0'[2k]$, while the high-pass subband is formed by retaining odd indexed samples, $y_1[k] = y_1'[2k+1]$.[3]

We say that the subbands are "critically sampled" since the combined sample rate of the low- and high-pass subbands is identical to that of the original input sequence $x[n]$. As suggested by Fig. 6, it is possible to recover the input sequence from its subbands. Upsampled sequences $y_0''[n]$ and $y_1''[n]$ are formed by inserting zeros into the odd (even) locations which were discarded by the subsampling operators. Low- and high-pass synthesis filters $g_0[n]$ and $g_1[n]$ are applied to the upsampled sequences, and the results are summed.

1) Finite Support Filters: The fact that it is possible to perfectly reconstruct $x[n]$ from its subbands is not particularly surprising. What is surprising, however, is that it is pos-

[3]We note that subband transforms are more commonly described in terms of subsampling operators which retain only the even indexed samples. The use of two different subsampling operators, one for each type of subband, is more convenient for our purposes, leading to simpler descriptions of the filter properties.

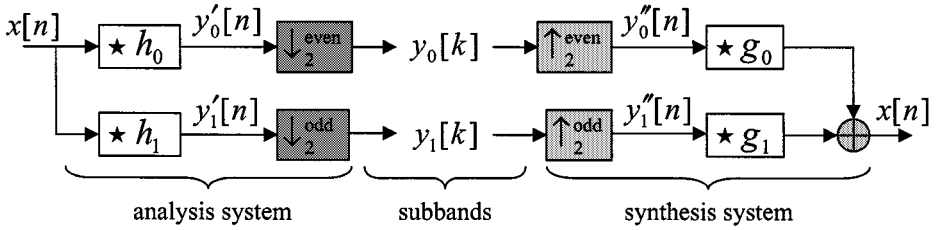

Fig. 6. One-dimensional, two-channel subband transform.

sible to find filters h_0, h_1, g_0, and g_1 all with finite support, such that this "perfect reconstruction" property holds. When the filters all have finite support, it can be shown that the analysis and synthesis filters must be related through

$$g_0[n] = \alpha^{-1}(-1)^n h_1[n] \quad \text{and} \quad g_1[n] = \alpha^{-1}(-1)^n h_0[n]. \tag{1}$$

That is, the low-pass synthesis filter is obtained by modulating the high-pass analysis filter and the high-pass synthesis filter is obtained by modulating the low-pass analysis filter. The factor α serves to compensate for the gains of the analysis filters, whose selection is implementation dependent.

In JPEG2000 Part 1, the filter impulse responses have odd lengths and are symmetric about the origin.[4] That is, $h_q[n] = h_q[-n]$, $q = 0, 1$. Symmetric (or linear phase) filters are used almost exclusively in image processing due to their edge preservation properties [11, Sec. 4.1]. Filters with nonlinear phase characteristics introduce visually disturbing distortion at image edges, which would have an adverse impact on the quality of the reduced resolution images produced by the final multiresolution transform.

It can be shown that odd-length symmetric filters satisfying the perfect reconstruction property must have lengths which differ by an odd multiple of 2 [12], [2]. Part 1 of the JPEG2000 standard [1] prescribes two different sets of filters, each having lengths which differ by the minimum of two samples. The 5/3 transform has low- and high-pass analysis filters with lengths 5 and 3; the filter coefficients are

$$h_0[n] = \begin{cases} 0.75, & \text{if} n = 0 \\ 0.25, & \text{if } n = \pm 1 \\ -0.125, & \text{if } n = \pm 2 \end{cases}$$

$$h_1[n] = \begin{cases} 0.5, & \text{if } n = 0 \\ -0.25, & \text{if } n = \pm 1. \end{cases} \tag{2}$$

The 9/7 transform has analysis filters

$$h_0[n] = \begin{cases} 0.602\,949\,018\,236, & \text{if } n = 0 \\ 0.266\,864\,118\,443, & \text{if } n = \pm 1 \\ -0.078\,223\,266\,529, & \text{if } n = \pm 2 \\ -0.016\,864\,118\,443, & \text{if } n = \pm 3 \\ 0.026\,748\,757\,411, & \text{if } n = \pm 4 \end{cases}$$

$$h_1[n] = \begin{cases} 0.557\,543\,526\,229, & \text{if } n = 0 \\ -0.295\,635\,881\,557, & \text{if } n = \pm 1 \\ -0.028\,771\,763\,114, & \text{if } n = \pm 2 \\ 0.045\,635\,881\,557, & \text{if } n = \pm 3. \end{cases} \tag{3}$$

[4]The fact that both filters are symmetric about $n = 0$ is a consequence of the simplifying yet less common subsampling convention adopted in this paper.

As described here, the low-pass filters each have a dc gain of $\hat{h}_0(0) = 1$ and a Nyquist gain of $\hat{h}_1(\pi) = 0$, while the high-pass filters each have a dc gain of $\hat{h}_1(0) = 0$ and a Nyquist gain of $\hat{h}_1(\pi) = 1$. Under these conditions,[5] the synthesis filters are found by setting $\alpha = 1/2$ in (1).

While Fig. 6 clarifies the role of subband transforms in decomposing a signal into its low- and high-frequency components, the equivalent representation in Fig. 7 provides insight into the distinction between subband transforms and block transforms, such as the block DCT used by JPEG. The low-pass subband samples may be associated with the even indexed locations in an interleaved sequence, $y[n]$, while the high-pass samples may be associated with the odd indexed locations in $y[n]$. Subband analysis and synthesis may then be expressed as time-varying convolution operators, mapping $x[n]$ to $y[n]$ and vice versa, according to

$$y[n] = \sum_p x[p] h_{n \bmod 2}[n - p]$$

and

$$x[n] = \sum_p y[p] g_{p \bmod 2}[n - p].$$

Fig. 7 illustrates these operations for the specific case of the 5/3 transform. Each subband sample $y[p]$ is obtained as a weighted average of the input samples $x[n]$, where the weights are the samples of the time reversed impulse response, $h_{p \bmod 2}[-n]$, shifted to location p. These time-reversed and shifted impulse responses are the transform's "analysis functions." As seen in the figure, the analysis functions overlap with one another. By contrast, block transforms such as the DCT break the input signal into blocks, analyzing the samples in each block without reference to other blocks.

During synthesis, the signal is reconstructed by summing a collection of overlapping contributions, one from each subband sample. The contribution due to subband sample $y[p]$ is obtained by shifting the synthesis impulse response $g_{p \bmod 2}[n]$ to location p and scaling it by $y[p]$. These shifted impulse responses are the transform's "synthesis functions" and their overlapping regions of support are evident from Fig. 7. By contrast, block transforms such as the DCT reconstruct the signal in independent blocks.

This distinction between overlapping and independent block transforms plays a major role in shaping the appearance of the compression artifacts which appear in lossy

[5]The JPEG2000 standard [1] adopts the slightly different normalization of $\hat{h}_0(0) = 1$ and $\hat{h}_1(\pi) = 2$ in describing these two transforms, although that convention is slightly less convenient for fixed point implementations.

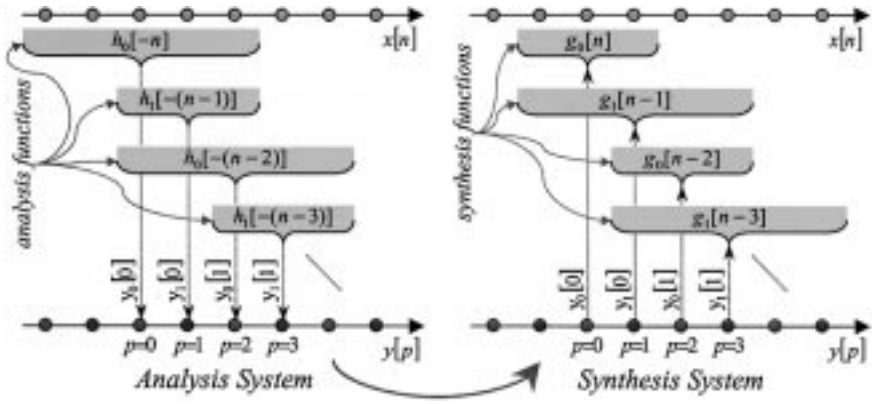

Fig. 7. Overlapping analysis and synthesis operators associated with a 1-D two-channel subband transform.

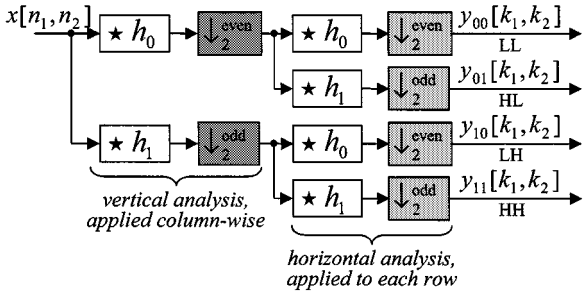

Fig. 8. Separable 2-D subband analysis. Synthesis is similar.

reconstructed images. At high compression ratios, block boundary artifacts dominate the appearance of JPEG compressed imagery, while the artifacts in JPEG2000 images are more distributed. Fig. 23 provides an illustration of this behavior. As we shall see in Section V, the overlapping synthesis functions also play an important role in shaping the subjective experience associated an interactive image retrieval application.

2) Image Transforms: Image transforms are almost invariably constructed by separable application of a 1-D transform in the vertical and horizontal directions. In the case of JPEG2000, the 1-D transform illustrated in Figs. 6 and 7 is applied first to each column of the image, producing vertically low- and high-pass subband images. The same 1-D transform is then applied to each row of the vertical subband images, decomposing each of them into horizontally low- and high-pass subbands. These operations are illustrated in Fig. 8. JPEG2000 Part 1 employs the well-known strategy of symmetric extension [13], [14], [2] at image boundaries to ensure that the total number of subband samples required to represent the image is the same as the original number of image samples.

The four subbands are denoted LL_1, HL_1 (horizontally high pass), LH_1 (vertically high pass), and HH_1. Each subband has half as many rows and half as many columns as the original image so that the total number of samples in the subbands is identical to the number of samples in the image. As shown in Fig. 3, the HL_1 subband contains information about vertically oriented image features, since these change most rapidly in the horizontal direction. Similarly, the LH_1 subband contains information about the horizontally oriented features and HH_1 responds most strongly to diagonally oriented image features. The LL_1 subband is a good low-resolution version of the image, since $h_0[n]$ acts as an antialiasing filter, applied in both directions prior to subsampling.

Applying this same separable subband transform to the LL_1 subband yields further subbands LL_2, HL_2, LH_2, and HH_2, each having one quarter the number of rows and columns as the original image. Continuing in this way, we obtain a tree-structured decomposition of the form illustrated in Fig. 2. A tree with D transform stages yields $3D+1$ subbands and offers $D+1$ image resolutions. The lowest resolution is obtained by discarding all but the LL_D subband at the base of the tree. The next resolution, LL_{D-1}, is obtained by separable synthesis of the LL_D, HL_D, LH_D, and HH_D subbands. In general, subbands LL_D, HL_D through HL_{d+1}, LH_D through LH_{d+1}, and HH_D through HH_{d+1} are sufficient to synthesize the image to resolution LL_d, where LL_0 denotes the original, full-resolution image.

In JPEG2000, each subband is coded independently. As a result, subsets of the compressed bitstream corresponding to each of the $D+1$ available resolutions may be obtained simply by discarding the unnecessary subbands. Unlike the Laplacian pyramid of Fig. 5, there is no redundancy in this multiresolution transform. The number of subband samples which must be quantized and coded is identical to the number of samples in the original image.

3) Wavelet Perspective: The terms wavelet transform and subband transform are often used interchangeably. More specifically, a DWT is generally understood to be a tree-structured subband transform with the structure shown in Fig. 2. There is, however, a subtle yet important distinction between wavelet and subband transforms. Subband transforms were first proposed in the mid-1970s for the coding of speech signals by Croisier *et al.* [15], [16]. Early work [17]–[21] focused on the properties of the analysis and synthesis filters and the conditions required for perfect reconstruction. Although tree-structured decompositions were of substantial interest, the basic building block of Fig. 6 was studied and optimized in isolation.

By contrast, wavelet transforms are concerned with the properties of the iterated transform. Since the transform is

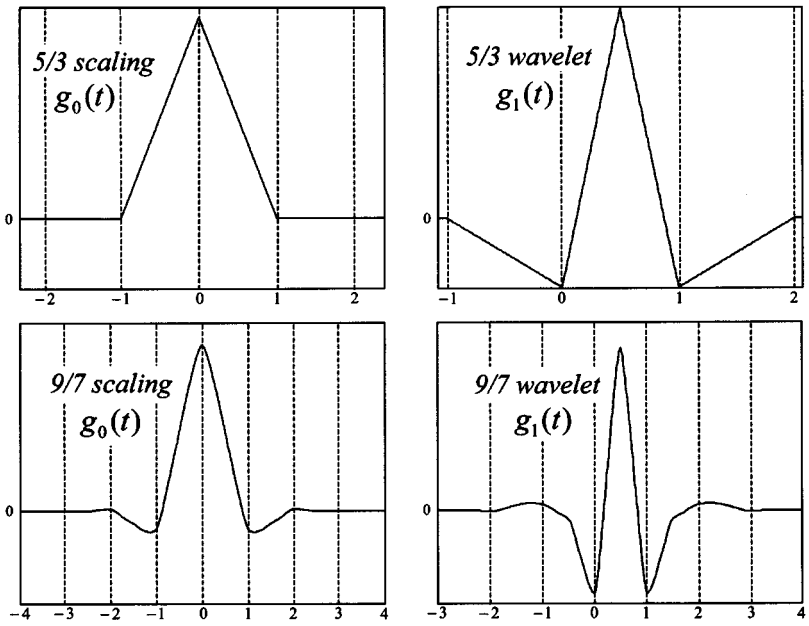

Fig. 9. Synthesis scaling and wavelet functions associated with the 5/3 and 9/7 subband filter sets.

linear, any given sample in any given subband from the tree makes an additive contribution to the reconstructed image. This contribution is of the form $y_{ij}^{(d)}[\mathbf{p}]g_{ij}^{(d)}[\mathbf{n}-2^d\mathbf{p}]$, where d is the level in the tree, $i, j \in \{0, 1\}$ identify the particular subband at that level, and $y_{ij}^{(d)}[\mathbf{p}] \equiv y_{ij}^{(d)}[p_1, p_2]$ is the subband sample value itself. The synthesis function $g_{ij}^{(d)}[\mathbf{n}]$ satisfies

$$g_{ij}^{(d)}[n_1, n_2] = g_i^{(d)}[n_1]g_j^{(d)}[n_2]$$

where $g_i^{(d)}[n]$ may be derived directly from the basic synthesis filters $g_0[n]$ and $g_1[n]$ by a process of iterative convolution and upsampling.

A wavelet transform is one in which $g_0^{(d)}[n]$ and $g_1^{(d)}[n]$ converge (as $d \to \infty$) to rate 2^d samplings of a pair of bounded continuous functions $g_0(t)$ and $g_1(t)$, known as the scaling function and wavelet function, respectively. The properties of the tree-structured transform are ultimately governed by these functions. The wavelet perspective, pioneered by Daubechies [22], [23] and Mallat [24], provides substantial intuition concerning the interpretation of multiresolution subband transforms as well as guidance concerning the selection of good filters for tree-structured transforms. Fig. 9 shows the wavelet and scaling functions associated with the 5/3 and 9/7 transforms defined by JPEG2000 Part 1. The interested reader is referred to [25] for a thorough development of these two filter sets. We note that the 5/3 transform was first proposed for image compression by Le Gall and Tabatai [26].

C. Lifting and Reversibility

In this section, we briefly introduce lifting realizations of the wavelet transform [27]. Lifting plays a central role in JPEG2000 where it enables efficient lossy and lossless compression to be achieved within a common framework [28], even within a single compressed data stream. The lifting framework also provides a vehicle for exploiting inherent structure in the analysis and synthesis filters to achieve computational and memory savings.

Consider again the basic 1-D two-channel subband transform of Fig. 6. A trivially invertible transform may be obtained by setting all filters to the identity operator. In this case, the subbands $y_0[k]$ and $y_1[k]$ are simply the even and odd subsequences of $x[n]$. This entirely useless transform is sometimes known as the "lazy wavelet." Starting from the lazy wavelet, more interesting transforms may be constructed through a sequence of so-called "lifting" steps, as illustrated in Fig. 10. The jth step converts an initial set of subbands $y_0^{(j-1)}[k]$ and $y_1^{(j-1)}[k]$ into a new set of subbands $y_0^{(j)}[k]$ and $y_1^{(j)}[k]$. The invertibility of each step is ensured by insisting that only one of the two subbands $y_{p_j}^{(j-1)}[k]$ be updated in step j and that the update consist in the addition of a filtered version of the other subband $y_{1-p_j}^{(j-1)}[k]$. In the example of Fig. 10, $p_j = j \bmod 2$ so that the steps $j = 1, 3, \ldots$ update the high-pass subbands, while steps $j = 2, 4, \ldots$ update the low-pass subbands. Specifically, we have

$$\begin{pmatrix} y_{p_j}^{(j)}[k] \\ y_{1-p_j}^{(j)}[k] \end{pmatrix} = \begin{pmatrix} y_{p_j}^{(j-1)}[k] + \sum_i \lambda_j[i] y_{1-p_j}^{(j-1)}[k-i] \\ y_{1-p_j}^{(j-1)}[k] \end{pmatrix}$$

(forward step)

$$\begin{pmatrix} y_{p_j}^{(j-1)}[k] \\ y_{1-p_j}^{(j-1)}[k] \end{pmatrix} = \begin{pmatrix} y_{p_j}^{(j)}[k] - \sum_i \lambda_j[i] y_{1-p_j}^{(j)}[k-i] \\ y_{1-p_j}^{(j)}[k] \end{pmatrix}$$

(inverse step).

Subband synthesis may thus be effected simply by reversing the order of the lifting steps and the signs of the update terms, as shown in Fig. 11.

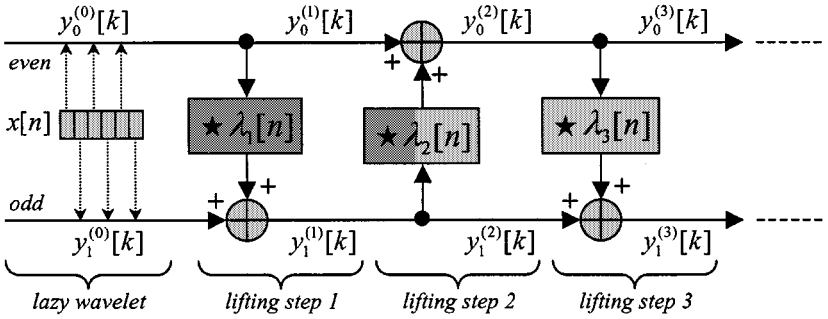

Fig. 10. Lifting steps for subband analysis.

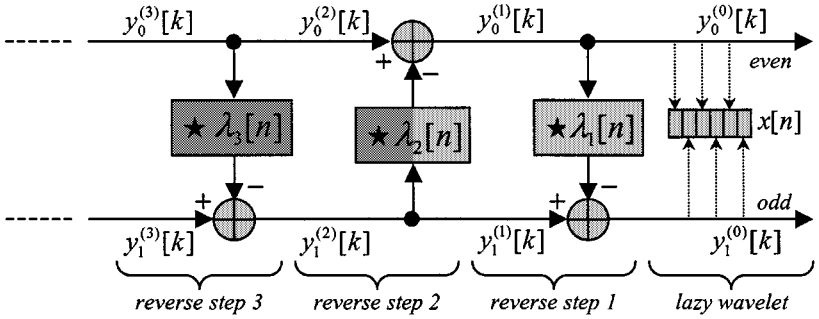

Fig. 11. Lifting steps for subband synthesis.

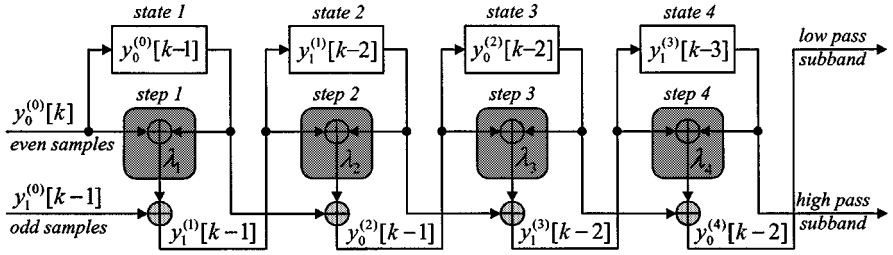

Fig. 12. Lifting analysis state machine for odd-length symmetric filters with lengths $2L+1$ and $2L-1$, shown here with $L=4$ lifting steps.

It turns out that any two-channel subband transform with finite support filters can be implemented through some sequence of lifting steps. The lifting steps have a particularly simple form when the subband filters are symmetric, with lengths $2L+1$ and $2L-1$, as is the case for the 5/3 and 9/7 filters defined by JPEG2000 Part 1. In this case, exactly L steps are required, each of which involves a two-tap filter with identical coefficients, λ_j [2, Sec. 6.4.4]. The specific structure is illustrated in Fig. 12. For the 5/3 transform, there are $L=2$ lifting steps with $\lambda_1=-(1/2)$ and $\lambda_2=1/4$. For the 9/7 filter set, there are $L=4$ lifting steps with

$$\lambda_1=-1.586\,134\,342, \qquad \lambda_2=-0.052\,980\,118$$
$$\lambda_3=0.882\,911\,075, \qquad \lambda_4=0.443\,506\,852.$$

Substituting the above coefficients, the reader may verify (by direct expansion) that the structure of Fig. 12 implements scaled versions of the 5/3 subband analysis filters of (2) or the 9/7 analysis filters of (3), as appropriate. Fig. 12 actually depicts a state machine, having L state variables. To produce each pair of subband samples, two new input samples must be pushed into the machine (from the left), and $2L$ additions and L multiplications must be performed. In the case of the 5/3 transform, the multiplication factors are trivial powers of 2. Even the 9/7 transform requires only two multiplications per sample. When the 1-D transform is applied separably to images, these meager complexity figures double.

If the application supplies image samples in a line-by-line fashion, the vertical transform may be implemented with as little as L line buffers, while the horizontal transform requires memory for only L samples. In the case of a full D-level DWT, we note that each successive stage operates on images whose width and height are reduced by a factor of 2 from the previous stage. Accounting for the need to buffer pairs of image lines between DWT stages, the total number of samples which must be buffered to implement a D level DWT is $(2+L)(W+2^{-1}W+2^{-2}W+\cdots+2^{1-D}W)<2(2+L)W$, where W is the full width of the image. Similar considerations apply for DWT synthesis. For a much more thorough analysis of the computational and memory requirements of the DWT, the reader is referred to [2].

Perhaps the most important property of the lifting structure is that the lifting filters may be modified in any desired manner without compromising invertibility. This prop-

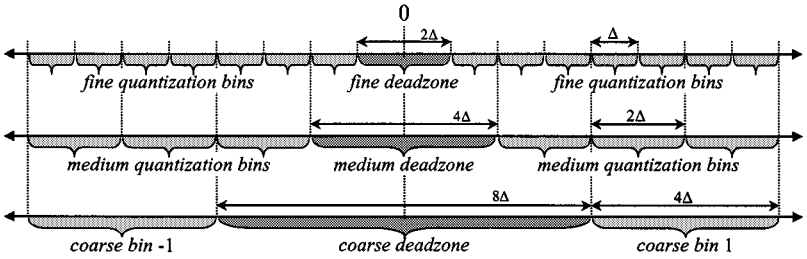

Fig. 13. Three embedded deadzone quantizers, each having a deadzone (or "fat zero") which is twice as large as the other quantization bins.

erty is exploited in JPEG2000 to provide efficient lossless and lossy compression from the same algorithm. Specifically, JPEG2000 defines "reversible" transforms in which the output of each lifting filter is rounded to an integer [28], [29]. In JPEG2000 Part 1, this rounding is available only in conjunction with the 5/3 transform, whose modified lifting steps are as follows:

$$y_1^{(1)}[k] = y_1^{(0)}[k] - \left\lfloor \frac{y_0^{(0)}[k] + y_0^{(0)}[k+1]}{2} \right\rfloor$$
$$= x[2k+1] - \left\lfloor \frac{x[2k] + x[2k+2]}{2} \right\rfloor$$
$$y_0^{(2)}[k] = y_0^{(1)}[k] + \left\lfloor \frac{y_1^{(1)}[k-1] + y_1^{(1)}[k]}{4} + \frac{1}{2} \right\rfloor.$$

Here $\lfloor x \rfloor$ is the floor function, rounding x down to the nearest integer. In this way, integer-valued image samples are transformed to integer-valued subband samples, all of which have a similar precision to that of the original image samples.

The term "reversible" is used to distinguish this type of transform from most conventional image transforms, from which the image cannot be exactly recovered using finite numerical precision. For example, the DCT used by JPEG has irrational coefficients, meaning that the transform samples cannot be represented exactly using any finite number of bits. Such a transform is clearly inappropriate for lossless compression. By contrast, JPEG2000 offers both state-of-the-art lossy compression and near state-of-the-art lossless compression with a single algorithm. In conjunction with the embedded quantization and coding strategies described next, JPEG2000 is able to produce an efficient losslessly compressed representation of the image, which embeds any number of efficient lossy compressed representations.

III. Embedded Block Coding

Just as multiresolution transforms are the key to resolution scalability, embedded quantization and coding are fundamental to distortion scalability. As mentioned in Section I-B (see Fig. 3), DWT subbands are partitioned into smaller "code-blocks," each of which is coded independently; typical code-block dimensions in JPEG2000 are 32×32 or 64×64. Associated with each block b is a finely embedded bitstream, which may be truncated to any of a large number of different lengths $L_b^{(z)}$, $z = 1, 2, \ldots$.

Most significantly, the majority of the available truncation points correspond to efficient compressed representations of the block's samples, in the rate-distortion sense. Specifically, the mean squared quantization error (distortion) incurred when reconstructing the samples from a length $L_b^{(z)}$ prefix of the full bitstream is comparable to that which could be obtained from a good nonembedded algorithm producing a bitstream with the same length $L_b^{(z)}$. The difference is that nonembedded coders produce bitstreams which cannot be truncated; quantization parameters must be adjusted at encode time, until the desired code length is achieved. The rich scalability offered by JPEG2000 is a direct consequence of embedded block coding, which allows the distortion associated with each individual code-block in each spatial frequency subband to be adjusted simply by truncating the block's bitstream.

In this section, we provide a brief overview of the embedded block coding algorithm used in JPEG2000. Since prefixes of an embedded bitstream must correspond to successively finer quantizations of the block's sample values, embedded coders are necessarily associated with a family of quantizers. In fact, these quantizers are inevitably embedded [30, Sec. 4B], in the sense that the quantization bins of a finer quantizer must be completely contained within those of a coarser quantizer.

JPEG2000 uses scalar "deadzone" quantizers having the structure depicted in Fig. 13. The central quantization bin, corresponding to those sample values which get quantized to 0, is known as the "deadzone." By making the deadzone twice as large as the nonzero quantization bins, a family of embedded quantizers arises when the step size Δ is halved between successive members of the family. The enlarged deadzone, sometimes called a "fat zero," is also helpful when coding high-frequency subband samples, which tend to be close to zero except in the neighborhood of appropriately oriented image edges and other important features. The numerous zero-valued quantization indices produced by samples falling within the deadzone can be efficiently coded using the adaptive arithmetic coding tools described shortly.

A. Bit-Plane Coding

The embedded quantization structure of Fig. 13 may be conveniently associated with the bit planes in a sign-magnitude representation of the subband samples. Let $y_b[\mathbf{j}] \equiv y_b[j_1, j_2]$ denote the 2-D sequence of subband samples associated with code-block b and let $q_b^{(0)}[\mathbf{j}]$ denote the quanti-

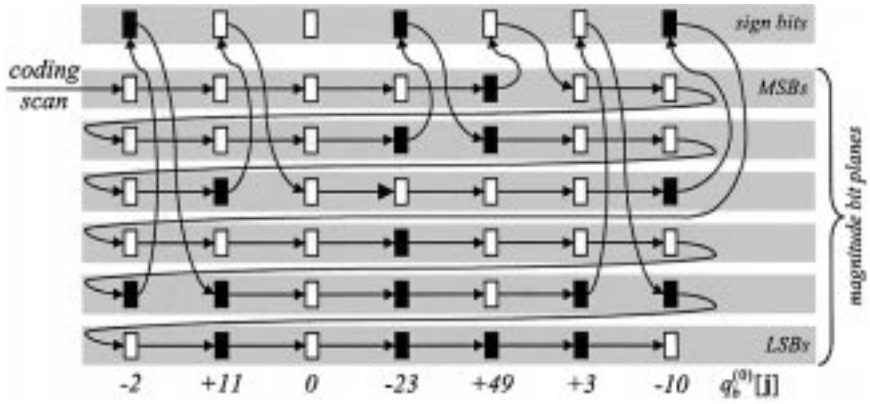

Fig. 14. Bit-plane coding procedure, shown for $K = 6$ magnitude bit planes. The MSBs correspond to magnitude bit-plane 5, while LSBs correspond to magnitude bit-plane 0. Black and white boxes correspond to values of "1" and "0," respectively.

zation indices (the bin labels) associated with the finest dead-zone quantizer for this block, having step size Δ_b. Then

$$q_b^{(0)}[\mathbf{j}] = \text{sign}(y_b[\mathbf{j}]) \cdot \left\lfloor \frac{|y_b[\mathbf{j}]|}{\Delta_b} \right\rfloor$$

where $\lfloor x \rfloor$ denotes the floor function, rounding x down to the nearest integer. Letting $q_b^{(p)}[\mathbf{j}]$ denote the indices of the coarser quantizer with step size $2^p\Delta_b$, we find that

$$q_b^{(p)}[\mathbf{j}] = \text{sign}(y_b[\mathbf{j}]) \cdot \left\lfloor \frac{|y_b[\mathbf{j}]|}{2^p\Delta_b} \right\rfloor = \text{sign}\left(q_b^{(0)}[\mathbf{j}]\right) \cdot \left\lfloor \frac{\left|q_b^{(0)}[\mathbf{j}]\right|}{2^p} \right\rfloor .$$

Thus, the coarser quantization indices $q_b^{(p)}[\mathbf{j}]$ are obtained simply by discarding the least significant p bits from the binary representation of the finer quantization indices' magnitudes, $|q_b^{(0)}[\mathbf{j}]|$.

Based on this observation, an embedded bitstream may be formed in the manner suggested by Fig. 14. Assuming a K-bit magnitude representation, the coarsest quantization indices, $|q_b^{(K-1)}[\mathbf{j}]|$, are represented by the most significant magnitude bit of each sample, together with the signs of those samples whose magnitudes are not quantized to zero. A bit-plane coder walks through each of the samples, coding these bits. If the bitstream is truncated at this point, the decoder receives the coarsest quantization indices, $q_b^{(K-1)}[\mathbf{j}]$. The bit-plane coder then moves to the next magnitude bit plane, coding the sign of any samples whose magnitudes first become nonzero in this bit plane. If the bitstream is truncated after this point, the decoder receives the finer quantization indices $q_b^{(K-2)}[\mathbf{j}]$, and so forth.

By exploiting the substantial redundancy which exists between successive bit planes, the embedded bitstream can have the same coding efficiency as a nonembedded bitstream in which each sample is coded completely before moving to the next sample. JPEG2000's bit-plane coder employs an adaptive arithmetic coding strategy known as the MQ coder, with 18 different adaptive probability models. The coder switches between these 18 models on the basis of previously coded bits from the same and previous bit planes in the same code-block. The relevant model adaptively estimates a probability distribution for the next bit to be coded and the arithmetic coder assigns a representation whose length closely matches the amount of information associated with the value of the bit, assessed in relation to the probability estimate. If the probability model predicts that the next magnitude bit should be zero with high probability and the value of the bit actually is zero (this happens very frequently), the code length increases by only a small fraction of a bit. For more information on bit-plane coding in JPEG2000, the reader is referred to [31] and [2]. For a tutorial treatment of arithmetic coding in general, the reader is also referred to [32].

B. Fractional Bit Planes

In the bit-plane coding procedure described above, the only natural truncation points for the embedded bitstream are the bit-plane end-points. These correspond to K different quantizations of the original code-block samples, whose quantization step sizes are related by powers of 2. JPEG2000's embedded block coding algorithm produces a much more finely embedded bitstream, with many more useful truncation points. This is achieved by coding the information in each new bit plane in a sequence of three "fractional bit-plane" coding passes. The first pass codes the next magnitude bit and any necessary sign bit, only for those samples which are likely to yield the largest reduction in distortion relative to their coding cost. Conversely, the last of the three coding passes in each bit plane processes those samples which are expected to be least effective in reducing distortion, relative to their cost in increased bit rate. Together, the three coding passes code exactly one new magnitude bit for every sample in the code-block.

Fig. 15 provides an illustration of these ideas, identifying the effect of different truncation lengths on the expected distortion in the reconstructed subband samples. As suggested by the figure, the operational distortion-length characteristic of a fractional bit-plane coder generally lies below (lower distortion) that of a regular bit-plane coder, except at the bit-plane end-points.

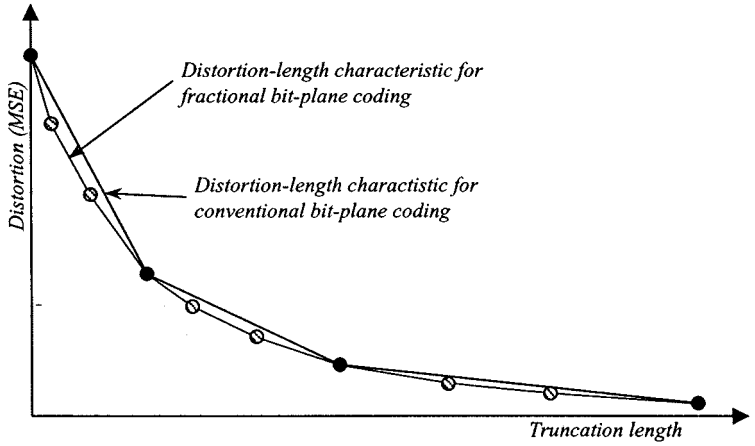

Fig. 15. Effect of fractional bit planes on the operational distortion-length characteristic of an embedded bitstream subject to truncation. Solid dots show the bit-plane end-points, while hatched dots correspond to the end-points of the fractional bit-plane coding passes.

In JPEG2000, the identity of the samples to be included in any given coding pass is assessed dynamically on the basis of the information which has already been coded. In particular, the samples in the first coding pass of any given bit plane are those which are currently zero (all coded magnitude bits so far have been zero), but have at least one nonzero neighbor. These tend to have the highest likelihood of "jumping out" of the deadzone in the next bit plane, yielding substantial reductions in distortion. Fractional bit-plane coding in JPEG2000 is the culmination of several research efforts [33]–[36]. We also note that similar principles lie at the heart of the well-known Set Partitioning in Hierarchical Trees (SPIHT) algorithm [37], while a more elaborate strategy was proposed in [38].

C. Quality Layers

JPEG2000 compressed images may contain a very large number of code-blocks, each with a large number of potential truncation points. To manage this complexity, JPEG2000 introduces the concept of "quality layers." Conceptually, each quality layer represents an increment in image quality. Unlike the embedded block coder, which is confined to process subband samples in a well-defined fashion which can be replicated by the decoder, quality layers may contain arbitrary (possible empty) incremental contributions from each of the code-blocks in the image, as illustrated in Fig. 4. The actual contribution made by each code-block to each layer is explicitly identified in the compressed data stream; in fact, this information is itself coded in a manner which exploits the redundancy between neighboring code-blocks.

Quality layers open the door for application developers to introduce novel or domain-specific interpretations of image quality. For example, medical images often contain local regions whose diagnostic value is much higher than others. The initial layers of a well-constructed JPEG2000 data stream should include much larger contributions from the code-blocks involved in reconstructing these sensitive regions. If the reversible transform described in Section II-C is employed, these important regions might even be reconstructed losslessly from relatively few quality layers, while distortion in the rest of the image might not dissipate until many more layers have been delivered to the decoder.

IV. Spatial Partitions in JPEG2000

One of the goals of JPEG2000 is to meet the emerging need for compression of and interaction with very large images. Uncompressed images from consumer digital cameras are already crossing the 10-MB barrier, while scanned documents and geospatial images can run to many gigabytes a piece. JPEG2000 provides two mechanisms for breaking the information in such large images into smaller spatial regions, which can be accessed on demand.

A. Tiles

JPEG2000 permits an image to be broken down into smaller subimages known as tiles, each of which is independently compressed. Each image component[6] of each tile (we call these "tile-components") has its own DWT, its own set of code-block bitstreams, and its own set of quality layers. Parameters controlling the number of DWT levels, quantization step sizes, the DWT kernels, and reversibility may all be adjusted on a tile-component basis by including appropriate markers in the code-stream. In fact, individual tiles or tile-components may be readily extracted and rewritten as valid JPEG2000 code-streams in their own right. Here and elsewhere, the term "code-stream" is used to identify the compressed data bits together with the marker codes and associated data segments which are used to signal coding parameters.

Although tiles are compressed entirely independently, there is value to including the tiles within a single JPEG2000 code-stream. All or most of the tiles will usually have a shared set of coding parameters. The tile dimensions are

[6]Recall that image components typically refer to the luminance and chrominance channels of a color image, but may play other roles in some applications.

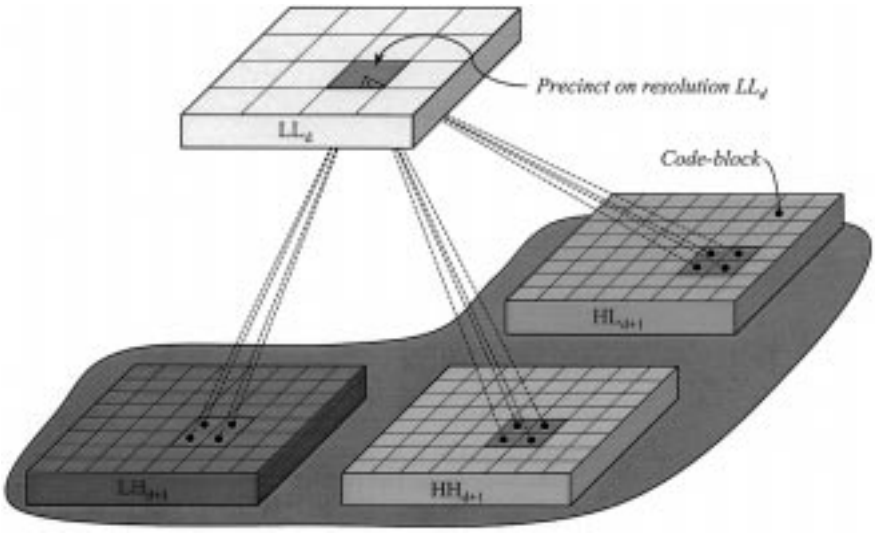

Fig. 16. Precinct partition on resolution LL_d, showing the induced partition on subbands HL_d, LH_d, and HH_d with their respective code-blocks.

deduced implicitly from those of the image and a set of four tiling configuration parameters. The tile partition is constructed in a manner which ensures that resolution scalability is properly preserved over the entire image, even if tiles have odd sizes or image components have odd subsampling factors. Most significantly, compressed data from the various tiles can be interleaved within the code-stream, allowing organizations in which image quality or resolution progressively improve over the entire image, as more of the code-stream is recovered. This is facilitated by the use of a common set of quality layers across all tiles with a common interpretation of image quality, although content providers are free to use quality layers in other ways.

Unfortunately, independent compression of each tile leads to boundary artifacts in images decompressed at lower bit rates. This is exactly the same phenomenon experienced with the JPEG algorithm, which applies a DCT transform independently to 8×8 image blocks. Typical tile sizes for JPEG2000, however, are much larger than those of JPEG blocks. For example, the lowest conformance profile for JPEG2000 decoders insists that images should either be untiled or use 128×128 tiles. JPEG blocking artifacts are revealed in Fig. 23, while Fig. 20 exposes tile boundary artifacts in a tiled JPEG2000 image.

It is important to realize that JPEG2000 does not mandate the use of tiles at all. Untiled images also possess spatial accessibility properties, as described below, while being free from boundary artifacts. All conformance profiles support untiled images, with arbitrarily large dimensions. In fact, for maximum interoperability, tiles are best avoided. Tiles are also not recommended for applications requiring substantial resolution scalability. If a large image is compressed using 128×128 tiles and subsequently decompressed at one eighth of the original resolution, the effective tile size is only 16×16, which is far too small to support an efficient compressed representation of the reduced resolution image, using the coding techniques defined by JPEG2000.

B. Precincts and Packets

As noted in Section I-B, the fact that "code-blocks" are coded independently is also a source of spatial accessibility in JPEG2000. Code-blocks form a partition of the image subbands, rather than the image itself. Each code-block has a limited region of influence within the reconstructed image, due to the finite support of the subband synthesis filters (see Fig. 7). Thus, given an image region of interest, it is possible to find the code-blocks whose influence intersects with that region and to selectively decompress only those code-blocks. This is discussed further in Section V. For the moment, however, we focus on the problem of accessing code-blocks within a JPEG2000 code-stream.

Unlike tiles, code-blocks are not explicitly delineated by code-stream markers. Direct access to code-blocks is further hampered by the fact that each code-block's bitstream may be distributed across many quality layers and the information describing these contributions is itself coded to exploit redundancy between neighboring code-blocks within the same subband. To overcome these obstacles, JPEG2000 defines spatial structuring elements known as "precincts." Each image resolution, LL_d (see Fig. 2) of each tile-component has its own precinct partition.

The purpose of precincts is to collect code-blocks into spatial and resolution groupings. As shown in Fig. 16, the precinct partition induces a partition of the subbands which are involved in synthesizing LL_d from the next lower resolution LL_{d+1}, if any. In a DWT with D levels, the precinct partition on LL_D is a partition of the code-blocks belonging to that subband. For $d < D$, the precinct partition on LL_d induces a partition of the subbands HL_{d+1}, LH_{d+1} and HH_{d+1}, through a convention which associates each sample in these subbands with a unique location on LL_d. By appropriately constraining the allowable dimensions for code-blocks and precincts, JPEG2000 ensures that the induced precincts on HL_{d+1}, LH_{d+1}, and HH_{d+1} each contain a whole number of

code-blocks. In this way, each code-block in the image is associated with exactly one precinct. In the example of Fig. 16, each precinct contains four code-blocks from each of its contributing subbands.

The value of precincts is that they may be explicitly identified as self-contained entities within a JPEG2000 code-stream. The inter-block coding of layer contributions mentioned in Section III-C extends only to blocks which belong to the same precinct, so the code-blocks of one precinct may be recovered without reference to any other. Apart from marker codes and their associated parameter data which are used to specify coding parameters, a JPEG2000 code-stream can be viewed as a concatenated list of "packets," where each packet consists of the contributions of all code-blocks from a single precinct to a single quality layer, together with the information required to identify the size of these contributions. In this way, a packet represents a single quality increment (one layer) of a single resolution increment (LL_{d+1} to LL_d), over a limited spatial region (precinct). Each image component of each tile has its own precincts and, hence, packets.

JPEG2000 provides a rich language for describing the order in which packets appear within the code-stream. Packets may be sequenced in resolution-major, component-major, or quality-major orders. Spatially progressive packet sequences are also defined to support low-memory streaming devices. In these sequences, the packets appear in an order which allows the image to be compressed or rendered progressively from top to bottom without buffering the entire compressed data stream.

The spatial extent of precincts may be separately adjusted in each resolution of each tile-component. Precincts may be so small that they include only one code-block from each of the contributing subbands or they may be as large as 2^{15} samples. Larger precinct sizes lead to somewhat higher compression efficiency at the expense of spatial accessibility. However, code-block data from an existing code-stream can be repacked into smaller or larger precincts with relatively little effort. The image server application described in Section V-A does exactly this.

JPEG2000 code-streams may also contain optional pointer information, which identifies the locations of particular tiles and/or particular packets. Such information allows regions of interest within the image to be selectively extracted from the code-stream and decompressed on demand.

V. Interactive Image Retrieval

The spatial accessibility features of JPEG2000 provide a firm foundation for interactive imaging applications. These features are extensively exploited by the Kakadu software tools, developed at the University of New South Wales. Applications based around Kakadu can choose to decompress any selected spatial region of any desired image components at any desired resolution. The high-level object interfaces offered by Kakadu allow applications to work with the compressed image from a geometric perspective which is rotated, flipped, windowed, or zoomed relative to the original. Image data are decompressed incrementally, in accordance with the requested geometry, in a manner which avoids any unnecessary computation or buffering.

These facilities allow enormous compressed images to be interactively decompressed and rendered with comparative ease. The "kdu_show" application[7] has been used to interactively view compressed images with uncompressed sizes of up to 6 GB. In fact, the size of the image need have very little impact on the responsiveness of the application or its memory and computational requirements. This is because Kakadu loads and parses only those portions of the code-stream which are required to reconstruct the current region and resolution of interest.[8] A caching strategy unloads previously recovered code-block bitstreams from memory as necessary.

A. Remote Browsing With JPIK

The Kakadu software framework also offers services to support interactive browsing of JPEG2000 compressed images over networks. Our recent experience with such services has been in the context of the "JPEG2000 Interactive with Kakadu" (JPIK)[9] protocol. JPIK is a connection-oriented network communication protocol using TCP, and optionally UDP, for the underlying network transport. As suggested by Fig. 17, the client communicates changes in its current region, resolution, and image components of interest, which the server uses to customize its ongoing transmission of compressed data. The server maintains a mirror image of the state of the client's cache, transmitting only new data which are not already available to the client.

The JPIK server essentially delivers a sequence of JPEG2000 packets to the client. The server recovers code-blocks of interest from the source code-stream on demand, taking advantage of whatever auxiliary pointer information that code-stream offers to minimize memory and disk accesses. It repackages the code-blocks on the fly into valid JPEG2000 packets which have the smallest precinct dimensions consistent with the code-block size. This means that each LL_D band precinct will have exactly one code-block and all other precincts will have three code-blocks, one from each of the HL_d, LH_d, and HH_d subbands at the relevant DWT level d. The original source code-stream's layering convention is used to build packets for these precincts, which are then delivered to the client. The first packet of every precinct whose code-blocks contribute to the current region of interest is sent first, followed by the second packet and so forth. An efficient variable-length signaling scheme is used to identify the particular packets which are being transmitted so that the client knows how to slot them into its cache.

[7]The application may be freely downloaded from http://www.kakadusoftware.com

[8]These statements are contingent on the use of modest precinct dimensions and the existence of suitable pointer information in the code-stream. In any event, Kakadu always loads the smallest possible amount of the code-stream at hand, which is required to satisfy the application's needs.

[9]A complete description of the JPIK protocol may be found by following the links at http://www.kakadusoftware.com

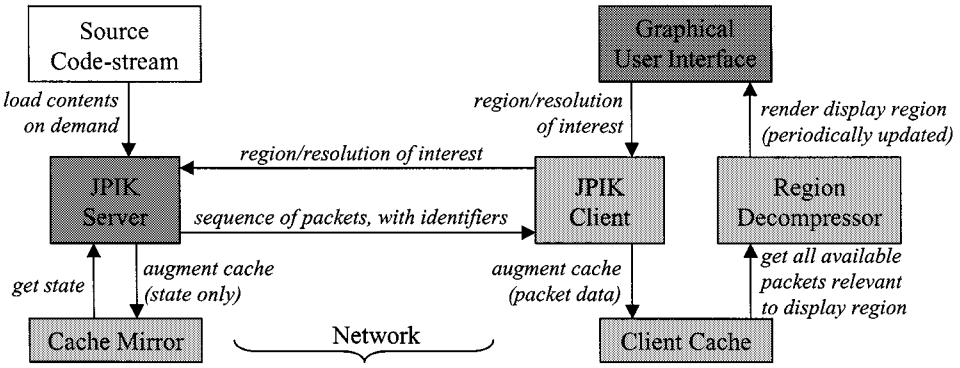

Fig. 17. Client–server interaction in a JPIK remote image browsing system. Note that the server maintains only the state of the client's cache and not its contents.

Fig. 18. Results of JPIK image browsing, showing a 972 × 883 fragment from the full 2944 × 1966 image. The reconstruction is based upon 59.4 kBytes of transmitted packet data, including all identifying headers.

Fig. 19. Expanded view of the client's region of interest in Fig. 18.

In this way, the client gradually accumulates information concerning the image, with higher image quality in those regions which the user has spent more time browsing. Fig. 18 shows the image quality obtained after a brief browsing session in which the user quickly zooms into an initial low-resolution version of the image, focusing attention on the region shown in Fig. 19. The original full color image has dimensions (width by height) of 2944 × 1966 for a total uncompressed size of 17.4 MB. The image is compressed to a maximum bit rate of 2 b/pixel to produce a 1.47-MB JPEG2000 file which is managed by the server. The server delivers a total of 59.4 kBytes to the client, which is enough to transfer all data relevant to the client's region of interest. The server will not resume transmission until the client's region or resolution of interest change.

Notice that the quality of the reconstructed image decreases progressively, with distance from the region of interest. In fact, Fig. 18 shows only a 972 × 883 region cropped from the complete image; quality continues to decline toward the borders of the full image (not shown here). This behavior provides convenient visual cues for interactive navigation within the image; it may be attributed to the properties of the DWT and the block-coding algorithm as follows. Recall from Section II-B that DWT synthesis involves the accumulation of overlapping contributions from each subband sample. The span of the overlapping synthesis functions grows as 2^d with the DWT level d. In the example at hand, the 5/3 transform has been used, with 32 × 32 code-blocks. The fraction of any given subband's samples which fall into one of its code-blocks also grows as 2^d. Since any code-block which intersects with the region of interest will eventually be delivered in full to the client, low-resolution information is bound to be available at a considerable distance from the region of interest, while high-resolution information is much more localized.

B. To Tile or Not to Tile?

Intuitively, one would expect that tiling the image should offer some advantages over the strategy described hitherto in which packets and code-blocks alone form the basis for spatial accessibility. This is because tiles are compressed entirely independently so that only a few tiles might be required to reconstruct a region of interest. Interestingly, our experience with the JPIK protocol suggests otherwise. The Kakadu server can deliver both tiled and untiled images to a remote client. In the case of tiled images, the DWT synthesis functions are confined to lie within the boundaries of their respective tiles and the code-blocks, precincts, and packets are also confined to tile boundaries.

Fig. 20 reveals the image quality obtained after browsing a tiled version of the same image shown in Fig. 18, with the same region of interest and tiles of size 128 × 128. The two images have been compressed in such a way as to yield reconstructed images with almost exactly the same mean squared error (MSE). In particular, the MSE of the region of interest recovered after browsing is exactly the same in both cases. To achieve this, the overall compressed bit rate of the tiled image is 2.25 b/pixel, which is 12% higher than

Fig. 20. Results of browsing a tiled image with 128 × 128 tiles, using the JPIK protocol. The region of interest and most other conditions are the same as those for the untiled result shown in Fig. 18.

in the untiled case. To satisfy the client's region of interest, the server transmits a total of 62.4 kBytes to the client, which is about 5% more than in the untiled case. Note that a small amount of information is supplied for tiles which do not intersect with the region of interest. This is because the Kakadu server starts transmitting data immediately, before it becomes aware of the client's region of interest. In this way, a few kBytes of low-resolution data are transmitted which span the entire image; this generally assists in interactive navigation.

Although it is difficult to precisely equalize the conditions associated with tiled and untiled image transmission, some useful conclusions may be drawn from Figs. 20 and 18. Even if tile boundary artifacts are not visible in the region of interest itself, their presence in surrounding regions is somewhat disturbing. Moreover, tiles do not appear to offer any advantage in transmission efficiency. In part, this is a consequence of the fact that the tiled image contains many more small code-blocks and packets than the untiled image. For example, at resolution LL_3, the 128 × 128 tiles measure only 16 × 16 so that the maximum packet size is 16 × 16 and the maximum code-block size for each of the associated subbands HL_4, LH_4, and HH_4 is 8 × 8—recall that code-blocks and precincts are all confined to tile boundaries. Small code-blocks do not compress efficiently and the numerous packets incur significant signaling overhead when transmitted to the client. While this overhead could be reduced by using fewer DWT levels (the present example has $D = 5$ levels), this would compromise resolution scalability.

The reader is reminded that the preceding discussion is concerned only with the value of tiles for scalable interactive image distribution in the context of JPEG2000 Part 1. Tiles do provide a simple tool for bounding memory requirements in both hardware and software implementations of the standard.[10] Tile boundaries can also be significantly reduced by exploiting some of the technologies embodied in Part 2 of the standard (see Section VII-C). Nevertheless, the JPEG2000 standard is not well adapted to coding small tiles or indeed small images. Each tile of each image component is decomposed into subbands and thence into code-blocks and packets, each of which incurs some signaling overhead. This signaling overhead grows with the number of DWT levels and the number of quality layers, both of which play key roles in imparting scalability to the compressed data stream. Interactive browsing of large images requires extensive resolution scalability, quality scalability for progressive display, and spatial accessibility for region of interest display. Our experience suggests that tiling has undesirable properties for such applications.

VI. JPEG Versus JPEG2000

In this section, we briefly discuss the relative merits of JPEG and JPEG2000. Additional information on this subject can be found in [39]. While JPEG2000 provides an advantage in compression efficiency over JPEG, its primary advantage lies in its rich feature set. Thus, we begin our discussion with this topic, deferring our discussion of compression performance until later in the section.

The JPEG standard specifies four modes: sequential, progressive, hierarchical, and lossless. In the sequential mode, imagery is compressed and decompressed in a block-based raster fashion from top to bottom. On the other hand, if the progressive mode of JPEG is employed, lower quality decompressions are possible and the code-stream is ordered so that the "most important" bits appear earliest in the code-stream. Hierarchical JPEG is philosophically similar. However, rather than improving quality, additional bytes are used to successively improve the "resolution" (or size) of the decoded imagery. When the lossless mode of JPEG is employed, only lossless decompression is available. High compression ratios are generally not possible with lossless compression.

Certain interactions between the modes are allowed according to the JPEG standard. For example, hierarchical and progressive modes can be mixed within the same code-stream. However, few if any implementations have exploited this ability. Also, quite different technologies are employed for the lossless and lossy modes. The lossless mode relies on predictive coding techniques, while lossy compression relies on the discrete cosine transform.

A JPEG code-stream must be decoded in the fashion intended by the compressor. For example, if reduced resolution is desired at the decompressor (when a progressive mode was employed at the compressor), the entire image must be decompressed and then downsampled. Conversion of a code-stream from one mode to another can be difficult.

[10]We must point out, however, that memory-efficient implementations are possible without resorting to tiling, since the DWT can be implemented efficiently without excessive working memory, as discussed in Section VII-C. Also, general purpose decompressors cannot assume that an image has been tiled.

Typically, such conversion must be accomplished via decompression/recompression, sometimes resulting in loss of image quality.

JPEG2000 tightly integrates the benefits of all four JPEG modes in a single compression architecture and a single code-stream syntax. The compressor can decide maximum image quality up to and including lossless. Also chosen by the compressor is the maximum resolution or size. Any image quality or size can be decompressed from the resulting code-stream, up to and including those selected at encode time.

Many types of progressive transmission are supported by JPEG2000. Progressive transmission is highly desirable when receiving imagery over slow communication links. As more data are received, the rendition of the displayed imagery improves in some fashion. JPEG2000 supports progression in four dimensions: quality, resolution, spatial location, and component.

The first dimension of progressivity in JPEG2000 is quality. As more data are received, image quality is improved. A JPEG2000 code-stream ordered for quality progression corresponds roughly to a JPEG progressive mode code-stream. We remark here that any quality up to and including lossless may be contained within a single compressed code-stream.

The second dimension of progressivity in JPEG2000 is resolution. In this type of progression, the first few bytes are used to represent a small "thumbnail" of the image. As more bytes are received, the resolution (or size) of the image increases by factors of 2 on each side. Eventually, the full-size image is obtained. A JPEG2000 code-stream ordered for resolution progression corresponds roughly to a JPEG hierarchical mode code-stream.

The third dimension of progressivity in JPEG2000 is spatial location. With this type of progression, imagery can be decompressed in approximately raster fashion, from top to bottom. This type of progression is particularly useful for memory-constrained applications such as printers. It is also useful for encoding. Low-memory scanners can create spatially progressive code-streams "on the fly" without buffering either the image or the compressed code-stream. A JPEG2000 code-stream ordered for progression by spatial location corresponds roughly to a JPEG sequential mode code-stream.

The fourth and final dimension of progressivity is the component. JPEG2000 supports images with up to 16 384 components. Most images with more than four components are from scientific instruments (e.g., LANDSAT). More typically, images are one component (gray-scale), three components (e.g., RGB and YCbCr), or four components (CMYK). Overlay components containing text or graphics are also common. With progression by component, a gray-scale version of an image might become available first, followed by color information, followed by overlaid annotations, and text, etc. This type of progression, in concert with the other progression types, can be used to effect various component interleaving strategies.

The four dimensions of progressivity are very powerful and can be "mixed and matched" within a single code-stream. That is, the progression type can be changed within a single code-stream. For example, the first few bytes might contain the information for a low-quality, gray-scale, thumbnail image. The next few bytes might add quality, followed by color. The resolution of the thumbnail might then be increased several times so that the size is appropriate for display on a monitor. The quality could then be improved until visually lossless display is achieved. At this point, the viewer might desire to print the image. The resolution could then be increased to that appropriate for the particular printer. If the printer is black and white, the color components can be omitted from the remainder of the code-stream.

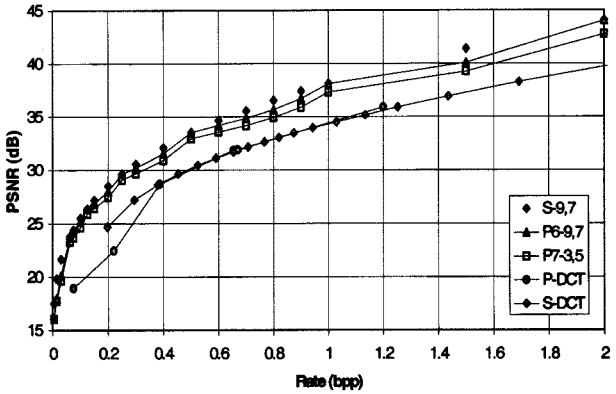

Fig. 21. Performance comparison of JPEG and JPEG2000.

The main points to be understood from this discussion are that: 1) the imagery can be improved in many dimensions as more data are received and 2) only the data required by the viewer need to be transmitted or decoded. This can dramatically improve the latency experienced by an image browsing application. Thus, the "effective compression ratio" experienced by the client can be many times greater than the actual compression ratio as measured by file size at the server.

Although stored files can only have a single order, an existing JPEG2000 code-stream can always be parsed and rewritten with a different progression order without actually decompressing the image. A smart server can even construct the most appropriate progression order on the fly, in response to user requests. The potential for intelligent and bandwidth-efficient client–server applications based on JPEG2000 has already been demonstrated in Section V-A. By contrast, the different progressions offered by JPEG involve various modifications to the underlying coding algorithm, which hampers efficient or dynamic reordering of the information by image servers and other applications.

A. Compression Performance

1) Lossy Compression: Fig. 21 [3] provides a performance comparison for two different JPEG code-streams and three different JPEG2000 code-streams representing the "bike" image (gray-scale, 2048 $\times$ 2560) from the JPEG2000 test set. A portion of this image is shown in Fig. 3. The two JPEG code-streams were generated using the Independent JPEG group software available online.[11] These JPEG code-streams are progressive (P-DCT) and sequential (S-DCT), each with optimized Huffman tables.

[11]Available. [Online.] http://www.ijg.org

The three JPEG2000 code-streams were generated using the JPEG2000 Verification Model software. They are single layer[12] with the 9/7 wavelet transform (S-9,7), six-layer quality progressive with the 9/7 wavelet transform (P6-9,7), and seven-layer quality-progressive with the reversible 5/3 wavelet transform (P7-3,5). In the last two cases, a single code-stream is generated, having quality layers optimized for the bit rates 0.0625, 0.125, 0.25, 0.5, 1.0, and 2.0 b/pixel. The single code-stream is simply truncated to obtain compressed representations at each test bit rate. For the 5/3 reversible transform, the seventh quality layer yields truly lossless decompression.

With this image,[13] the JPEG2000 results are significantly better than those produced using JPEG, for all modes and at all bit rates. With JPEG2000, the progressive performance is almost identical to the single-layer performance at bit rates for which the layers are optimized. The slight difference is due solely to the increased signaling cost associated with the additional layers. As mentioned in Section I-B, it is possible to decode at rates between those corresponding to layers. This is accomplished by discarding partial layers, with some resulting suboptimality. This is the cause of the "scallops" in the upper curves, where the progressive performance drops somewhat below the single-layer performance. This effect can be mitigated by adding more layers. The scallops will be effectively removed, but at the expense of a minor decrease in overall PSNR due to additional signaling overhead.

Although JPEG2000 provides significantly lower distortion for the same bit rate, the computational complexity is higher. The fastest JPEG2000 software implementations currently run roughly a factor of three times slower than optimized JPEG implementations.

2) Lossless Compression: The lossless compression performance of JPEG2000, for natural photographic imagery, is typically within about 2%–5% of JPEG-LS [40].[14] For compound documents, containing text and half tones, the comparison is not always as favorable. For such documents, JPEG-LS generally outperforms JPEG2000, producing compressed files 40%–45% smaller. On the other hand, computer-generated graphic imagery containing a limited number of colors can be efficiently compressed with JPEG2000. A particularly effective approach treats the image as a single palettized component. The sample values of this component then specify image sample colors via a look-up-table (LUT). Careful construction of the LUT often results in compression performance close to that of JPEG-LS [41].

3) Binary Imagery: Binary valued components (or binary valued tiles of components) can be compressed using JPEG2000. Lossless compression of such binary data can be accomplished by setting the bit depth to 1 and setting 0 levels of wavelet transform. The result of these settings is that no wavelet transform is performed, and the binary image is treated as a single bit plane at a single resolution. This bit plane is divided into code-blocks and subjected to context-dependent arithmetic coding. The performance of JPEG2000 is very similar to that of the CCITT facsimile compression standard G4 [42]. On the other hand, JBIG [43] outperforms JPEG2000, producing files which are about 30%–35% smaller.

Unfortunately, when the processing described in the previous paragraph is employed, scalability in quality and resolution are lost. On the other hand, spatial random access is preserved. If the wavelet transform is employed for compressing binary imagery, scalability and progressivity are preserved, but with some loss in compression efficiency over the "zero-level" case. The quality of lossy decompressed binary imagery can vary widely, depending on the image content and bit rate chosen.

B. Subjective Comparison of JPEG and JPEG2000

Visual comparisons of JPEG versus JPEG2000 have been conducted by Fujifilm Software California and Eastman Kodak [44].[15] The JPEG2000 imagery for these tests was generated using the 9/7 wavelet transform and contrast sensitivity function (CSF) weighting as described in [45]. Such weighting is used to modify the embedding of the code-stream so that (visually) more important data appear earlier. The JPEG imagery was generated using the Independent JPEG Group implementation. The default mode was employed (i.e., baseline sequential JPEG) and Huffman tables were optimized. All visual comparisons were made using 24-b color prints at 300 dpi.

The tests were conducted using six 24-b color images with natural photographic content. Ten "reference" JPEG images were created for each of the six original images. These reference images were created by compressing and decompressing to precise bit rates of 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.2, and 1.4 bit per color pixel. For example, a bit rate of 0.5 b/pixel corresponds to a compression ratio of 48:1. Four JPEG2000 "test" images were also created for each original image. These test images were compressed/decompressed to rates of 0.25, 0.50, 0.75, and 1.00 b/pixel using the JPEG2000 Verification Model software.

Visual quality testing was then carried out by six observers.[16] For a given original image, the JPEG reference prints were placed on a table in order of lowest to highest rate. Each observer was given a JPEG2000 test image and asked to find the JPEG reference image of comparable quality. This process was repeated for each JPEG2000 rate and each original image. In this way, the rate required for JPEG to achieve the same visual quality as JPEG2000 was determined.

The average of these results over the six observers is shown in Fig. 22. Two of the curves in this figure show results for individual images. The third curve shows the results averaged over all images. Each of these three curves represents the rate required by JPEG to achieve comparable perceptual quality to that of JPEG2000. For ease of comparison, the figure includes a fourth curve, indicating the rate required by JPEG2000 (to

[12]Code-streams with only one quality layer are not distortion scalable. A separate code-stream must be generated for each bit rate under test.

[13]There are images where the difference can be substantially less, especially in the range from 1.0 to 1.5 b/pixel.

[14]Software available at http://www.hpl.hp.com/loco/software.htm

[15]Also available from http://www.jpeg.org/public/wg1n1583.ppt

[16]Subsequent testing at Fujifilm Software California using more observers yielded similar results [58].

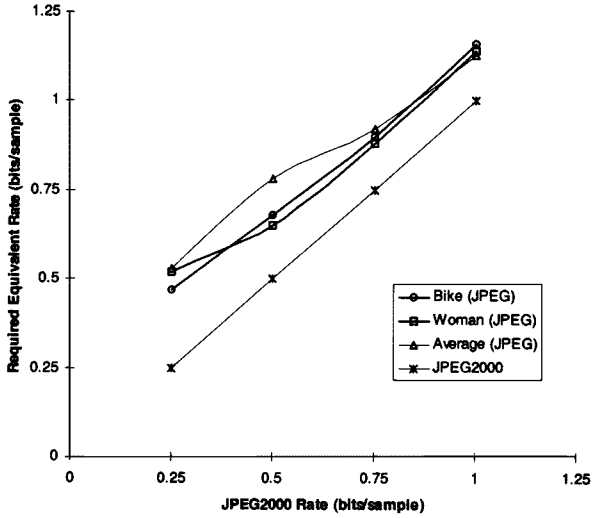

Fig. 22. Rate required to achieve visual quality equivalent to that of JPEG2000.

achieve the quality of JPEG2000). Of course, this latter curve is a straight line of slope 1.

From the figure, we see that, in the case of the bike image, JPEG2000 provides a decrease in compressed bit rate ranging from 14% to 47% over that of JPEG. Equivalently, JPEG requires an increase in bit rate ranging from 16% to 88% in order to achieve equivalent visual quality to that of JPEG2000. The results for the woman image are similar. On average, JPEG2000 provides a bit-rate reduction of between 11% and 53% relative to JPEG.

In each case, the largest improvements occur at the lower rates. This is not surprising since, at low rates, the "blocking" artifacts of JPEG tend to be significantly more annoying than the "smoothing" of JPEG2000, as demonstrated in Fig. 23. In fact, a general observation reported in [44] was that JPEG imagery tends to be "sharper" than JPEG2000 imagery at all encoding rates. Furthermore, the sharpness of JPEG increases more quickly than that of JPEG2000 as encoding rate is increased. At very high rates (>1.0 b/pixel), artifacts are nearly imperceptible for both JPEG and JPEG2000 compression. Careful observation, however, reveals that JPEG quality is comparable to that of JPEG2000. In fact, JPEG imagery can be slightly superior to JPEG2000 imagery at rates on the order of 2.0 b/pixel. On the other hand, the JPEG sharpness advantage is more than overcome at lower rates by the absence of blocking artifacts in JPEG2000.

VII. Other Features and Enhancements

A. Region of Interest Coding

In previous sections, we mentioned the possibility of varying quality by spatial region. Such variation can be effected at encode time or in subsequent parsing or decode operations. This capability derives from the independence of code-blocks, and so the code-block dimensions govern the granularity of the spatial regions that can be targeted.

JPEG2000 also allows the encoder to select entirely arbitrary regions of interest for preferential treatment [46], [47]. In this case, the region of interest (ROI) must be chosen at encode time and is not easily altered via parsing or decoding. For this form of ROI coding, wavelet coefficients that affect image samples within the ROI are preemphasized prior to bit-plane coding. This preemphasis amounts to a binary upshift of coefficient magnitudes. The amount of upshifting must be such that all bit planes of all wavelet samples in the ROI are encoded prior to the encoding of the most significant bit plane of the background (non-ROI) wavelet samples. For this reason, this method of ROI coding is referred to as the "max-shift" method. An example of this form of ROI coding is shown in Fig. 24. The overall encoding rate for this image is 0.125 b/pixel, large enough that some of the background data has been decoded.

B. Additional Parts to JPEG2000

The JPEG2000 standard is currently comprised of six parts, with additional parts under discussion. For the purpose of interchange, it is important to have a standard with a limited number of options, so that decoders in browsers, printers, cameras, or palm-top computers can be counted on to implement all options. In this way, an encoded image will be displayable by all devices.[17] For this reason, Part 1 describes the minimal decoder and code-stream syntax required for JPEG2000, which should be used to provide maximum interchange.

Part 1 [1] also describes an optional minimal file format known as JP2. The ".jp2" suffix should be applied only to files which conform to this file format, not to files which contain only JPEG2000 code -streams. JP2 is structured as a sequence of "boxes." Each box consists of a box-type identifier, a box length, and box contents. In addition to identification boxes and a box containing the compressed code-stream, every JP2 file must contain a description of the color space and associated information required to correctly render the image to a display device. JP2 defines a variety of optional boxes for describing image resolution, copyright ownership, and so forth [48].

There are many applications for image compression where interchange is less important than other requirements. For these applications, Part 2 [49] of the standard describes optional "value-added" extensions to enhance compression performance or enable efficient compression of less common data types, at the expense of interoperability. These extensions are not required of all implementations so that images encoded using Part 2 technologies may not be decodable by Part 1 decoders. Part 2 also describes an enhanced file format, known as JPX.

Part 3 [50], known as "Motion JPEG2000"[18] or "MJ2," provides a file format for representing sequences of images, each coded using the techniques described by JPEG2000 Part 1. The Part 3 file format is derived from Apple's "Quick Time" file format and is designed for compatibility with MPEG-4. In fact, the file formats described by Parts 1, 2, 3 and 6 all share the box concept and other attributes borrowed

[17]It is worth noting that the standard specifies only the decoder and code-stream syntax. Although informative descriptions of some encoding functions are provided in the text of the standard, there are no requirements that the encoder perform compression in any prescribed manner. This leaves room for future innovations in encoder implementations.

[18]We note that "Motion JPEG" has been a commonly used format for the purpose of video editing (e.g., in production studios), even though it was never officially standardized.

Fig. 23. Lena image (512 × 512) decompressed at 0.25 b/pixel using JPEG2000 (left) and JPEG (right). A cropped and zoomed portion of the fully decompressed image is displayed here.

from "Quick Time." These file formats may be said to belong to the "JP2 family," and it is possible to construct files conforming to all four formats simultaneously. Part 6 [51] describes a file format known as "JPM," which is tailored to the needs of compound document compression.

Part 4 [52] provides compliance/conformance definitions and testing procedures for implementations of Part 1, while Part 5 [53] includes reference software for Part 1. There are two software implementations included in Part 5. The JJ2000 implementation[19] is written in Java, while the JasPer implementation[20] [54] is written in C. Parts 4 and 5 work together to assist developers in producing compliant implementations.

At the time of this writing, several additional parts to the standard have been created as work items. These new parts deal with three-dimensional (3-D) coding extensions (Part 8), client–server protocols for JPEG2000 (Part 9), security (Part 10), and wireless transmission of JPEG2000 content (Part 11). Part 7 was reserved for a possible description of reference hardware implementations for Part 1, but is not currently being progressed.

C. Part 2 Features

Up to now, this paper has discussed only the technology in JPEG2000 Part 1. The purpose of this subsection is to provide a high-level overview of the additional features found in Part 2. Part 2 contains extensions to allow variable level offsets and point nonlinearities both as pre/postprocessing steps. The variable-level offset capability is useful for adjusting the "dc" level of imagery, while point nonlinearities provide for contrast adjustments such as gamma corrections.

[19]Available. [Online.] http://jj2000.epfl.ch

[20]Available. [Online.] http://www.ece.ubc.ca/~mdadams/jasper

Fig. 24. Example of ROI coding at 0.125 b/pixel. The rectangular ROI in the facial region is well preserved at the expense of the background.

As discussed previously, JPEG2000 Part 1 uses scalar quantization with a zero bin (deadzone) twice as wide as the other bins. JPEG2000 Part 2 allows for adjusting the deadzone sizes in scalar quantization as well as the ability to employ trellis-coded quantization [55]. Nonlinear compensations for visual masking can also be applied, with either scalar quantization or trellis-coded quantization, to obtain substantial improvements in visual quality [56], [57].

Several extensions are supported with respect to the wavelet transform. Substantial flexibility is available to select custom wavelet kernels. A rich language is also

provided for modifying the wavelet decomposition tree structure. Finally, the wavelet transform may be applied to overlapping "cells" and/or tiles. This latter feature allows block-based processing to be performed without the introduction of severe block artifacts.

Extended decorrelating transforms for multiple component imagery are also included in JPEG2000 Part 2. Whereas Part 1 defines only a single reversible color transform and a single irreversible transform (RGB to YCbCr), Part 2 supports general linear transforms, predictive transforms, and wavelet transforms for the decorrelation of components.

Enhanced support is also provided for ROI encoding. In addition to the max-shift method discussed in Section VII-A above, Part 2 provides for arbitrary up-shifts, with explicit signaling of the regions whose coefficient magnitudes are to be shifted. Explicit region signaling is confined to rectangular and elliptical regions of interest.

Finally, JPEG2000 Part 2 specifies the extended file format known as JPX. JPX is backward compatible with the JP2 file format of Part 1 but contains many enhancements. Such enhancements provide more flexibility in the specification of color spaces, opacity information, and metadata. Also included is the ability to combine multiple code-streams to obtain compositing or animation from a single JPX file.

VIII. Summary

JPEG2000 is much more than just a new way to compress digital imagery. Central to this new standard is the concept of scalability, which enables image components to be accessed at the resolution, quality, and spatial region of interest. The technology on which JPEG2000 is based departs radically from that used in the JPEG standard as an unavoidable consequence of the features required of the new standard.

As demonstrated in this paper, JPEG2000 improves on the compression performance offered by JPEG while simultaneously allowing interactive access to the image content. The information in a JPEG2000 code-stream may be reordered at will to suit a wide range of applications from memory-constrained hardware platforms such as printers to fully interactive client–server systems. It is possible to embed enormous images in a JPEG2000 code-stream, with qualities all the way up to lossless, while permitting access at much lower resolutions and/or qualities over networks with only modest capabilities.

Part 1 of the standard provides an excellent platform for efficient, interoperable interaction with rich image content while Part 2 provides extensions to serve the needs of special purpose applications.

References

[1] *Information Technology—JPEG 2000—Image Coding System—Part 1: Core Coding System*, ISO/IEC 15 444-1, 2000.

[2] D. Taubman and M. Marcellin, *JPEG 2000: Image Compression Fundamentals, Standards and Practice*. Norwell, MA: Kluwer, 2002.

[3] M. Marcellin, M. Gormish, A. Bilgin, and M. Boliek, "An overview of JPEG2000," in *Proc. IEEE Data Compression Conf.*, Snowbird, UT, 2000.

[4] M. Rabbani and R. Joshi, "An overview of the JPEG2000 still image compression standard," *Signal Process.: Image Commun.*, vol. 17, pp. 3–48, Jan. 2002.

[5] M. D. Adams, "The JPEG-2000 still image compression standard—Tech. Rep. distributed with the JasPer JPEG-2000 software,", [Online]. Available: http://www.ece.ubc.ca/~mdadams/jasper, Tech. Rep. N2412, ISO/IEC JTC1/SC29/WG1, Sept. 2001.

[6] C. Christopoulos, A. Skodras, and T. Ebrahimi, "The JPEG2000 still image coding system: An overview," *IEEE Trans. Consumer Electron.*, vol. 46, pp. 1103–1127, Nov. 2000.

[7] *Information Technology—JPEG—Digital Compression and Coding of Continuous-Tone Still Image—Part 1: Requirements and Guidelines*, ISO/IEC 10 918-1 and ITU-T Recommendation T.81, 1994.

[8] G. K. Wallace, "The JPEG still picture compression standard," *Commun. ACM*, vol. 34, pp. 30–44, Apr. 1991.

[9] W. Pennebaker and J. Mitchell, *Still Image Data Compression Standard*. New York: Van Nostrand, 1992.

[10] M. Gormish, E. Schwartz, A. Keith, M. Boliek, and A. Zandi, "Lossless and nearly lossless compression of high-quality images," *IEEE Trans. Signal Processing*, vol. 45, pp. 62–70, Mar. 1997.

[11] J. Lim, *Two-Dimensional Signal and Image Processing*. Englewood, NJ: Prentice-Hall, 1990.

[12] M. Vetterli and J. Kovačević, *Wavelets and Subband Coding*, NJ: Prentice-Hall, 1995.

[13] M. Smith and S. Eddins, "Analysis–synthesis techniques for subband image coding," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 48, pp. 1446–1456, Aug. 1990.

[14] C. Brislawn, "Classification of nonexpansive symmetric extension transforms for multirate filter banks," *Appl. Comput. Harmon. Anal.*, vol. 3, pp. 337–357, 1996.

[15] A. Croisier, D. Esteban, and C. Galand, "Perfect channel splitting by use of interpolation/decimation/tree decomposition techniques," in *Int. Conf. Information Sciences and Systems*, Aug. 1976, pp. 443–446.

[16] ——, "Application of quadrature mirror filters to split band voice coding systems," in *Proc. Int. Conf. Acoust. Speech and Signal Processing*, 1977, pp. 191–195.

[17] J. Johnston, "A filter family designed for use in quadrature mirror filter banks," in *Proc. Int. Conf. Acoust. Speech and Signal Processing*, 1980, pp. 291–294.

[18] F. Mintzer, "Filters for distortion-free two-band multirate filter banks," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-33, pp. 626–630, June 1985.

[19] M. Smith and T. I. Barnwell, "Exact reconstruction techniques for tree structured subband coders," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-34, pp. 434–441, June 1986.

[20] M. Vetterli, "Filter banks allowing perfect reconstruction," *Signal Process.*, vol. 10, pp. 219–244, Apr. 1986.

[21] P. Vaidyanathan, "Theory and design of m-channel maximally desimated quadrature mirror filters with arbitrary m, having the perfect reconstruction property," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-35, pp. 476–492, Apr. 1987.

[22] I. Daubechies, "Orthonormal bases of compactly supported wavelets," *Commun. Pure Appl. Math.*, vol. 41, pp. 909–996, Nov. 1998.

[23] I. Daubechies, *Ten Lectures on Wavelets*. Philadelphia, PA: SIAM, 1992.

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

[25] A. Cohen, I. Daubechies, and J.-C. Feauveau, "Biorthogonal bases of compactly supported wavelets," *Commun. Pure Appl. Math.*, vol. 45, pp. 485–560, June 1992.

[26] D. Le Gall and A. Tabatabai, "Sub-band coding of digital images using symmetric short kernel filters and arithmetic coding techniques," in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing*, Apr. 1988, pp. 761–764.

[27] W. Sweldens, "The lifting scheme: A custom-design construction of biorthogonal wavelets," *Appl. Comput. Harmon. Anal.*, vol. 3, pp. 186–200, Apr. 1996.

[28] R. Calderbank, I. Daubechies, W. Sweldens, and B. Yeo, "Wavelet transforms that map integers to integers," *Appl. Comput. Harmon. Anal.*, vol. 5, pp. 332–369, July 1998.

[29] M. Adams and F. Kossentini, "Reversible integer-to-integer wavelet transforms for image compression: Performance evaluation and anaylsis," *IEEE Trans. Image Processing*, vol. 9, pp. 1010–1024, June 2000.

[30] D. Taubman and A. Zakhor, "A common framework for rate and distortion based scaling of highly scalable compressed video," *IEEE Trans. Circuits Syst. Video Technol.*, pp. 329–354, Aug. 1996.
[31] D. Taubman, "High performance scalable image compression with EBCOT," *IEEE Trans. Image Processing*, vol. 9, pp. 1158–1170, 2000.
[32] I. Witten, R. Neal, and J. Cleary, "Arithmetic coding for data compression," *Commun. ACM*, vol. 30, pp. 520–540, June 1987.
[33] E. Ordentlich, M. Weinberger, and G. Seroussi, "A low-complexity modeling approach for embedded coding of wavelet coefficients," in *Proc. IEEE Data Compression Conf.*, Snowbird, UT, Mar. 1998, pp. 408–417.
[34] F. Sheng, A. Bilgin, P. Sementilli, and M. Marcellin, "Lossy and lossless image compression using reversible integer wavelet transforms," in *Proc. IEEE Int. Conf. Image*, Oct. 1998, pp. 876–880.
[35] D. Taubman, "Embedded, independent block-based coding of subband data,", Tech. Rep. N871R, ISO/IEC JTC1/SC29/WG1, July 1998.
[36] M. Marcellin, T. Flohr, A. Bilgin, D. Taubman, E. Ordentlich, M. Weinberger, G. Seroussi, C. Chrysafis, T. Fisher, B. Banister, M. Rabbani, and R. Joshi, "Reduced complexity entropy coding,", Tech. Rep. N1312, ISO/IEC JTC1/SC29/WG1, June 1999.
[37] A. Said and W. Pearlman, "A new, fast and efficient image codec based on set partitioning in hierarchical trees," *IEEE Trans. Circuits Syst. Video Technol.*, pp. 243–250, June 1996.
[38] J. Li and S. Lei, "Rate-distortion optimized embedding," in *Proc. Picture Coding Symp.*, Berlin, Germany, Sept. 1997, pp. 201–206.
[39] D. Santa Cruz, R. Grosbois, and T. Ebrahimi, "JPEG 2000 performance evaluation and assessment," *Signal Process.: Image Commun.*, vol. 17, pp. 113–130, Jan. 2002.
[40] *Information Technology—JPEG-LS—Lossless and Near-Lossless Compression of Continuous-Tone Still Images*, ISO/IEC 14495-1 and ITU-T Recommendation T.87, 1999.
[41] W. Zeng, J. Li, and S. Lei, " An efficient color re-indexing scheme for palette-based compression," in *Proc. IEEE Int. Conf. Image Processing*, vol. 3, Sept. 2000, pp. 476–479.
[42] CCITT Recommendation T.6, "Facsimile coding schemes and coding control functions for group 4 facsimile apparatus,", Recommendation T.6., 1984.
[43] *JBIG Bi-Level Image Compression Standard*, ISO/IEC 11 544 and ITU-T Recommendation T.82, 1993.
[44] T. Chinen and A. Chien, "Visual evaluation of JPEG2000 color image compression performance,", Tech. Rep. N1583, ISO/IEC JTC1/SC29/WG1, Mar. 2000.
[45] M. Nadenau and J. Reichel, "Opponent color, human vision and wavelets for image compression," in *Proc. 7th Color Imaging Conf.*, 1999, pp. 237–242.
[46] J. Askelof, M. Larsson Carlander, and C. Christopoulos, "Region of interest coding in JPEG 2000," *Signal Process.g: Image Compression*, vol. 17, pp. 105–111, 2002.
[47] C. Christopoulos, J. Askelof, and M. Larsson, "Efficient methods for encoding regions of interest in the upcoming JPEG 2000 still image coding standard," *IEEE Signal Processing Lett.*, vol. 17, pp. 247–249, Sept. 2000.
[48] J. Houchin and D. Singer, "File format technology in JPEG 2000 enables flexible use of still and motion sequences," *Signal Process.: Image Compression*, vol. 17, pp. 131–144, Jan. 2002.
[49] *Information Technology—JPEG 2000 Image Coding System—Part 2: Extensions*, ISO/IEC 15 444-2, 2002.
[50] *Information Technology—JPEG 2000 Image Coding System—Part 3: Motion JPEG 2000*, ISO/IEC 15 444-3, 2002.
[51] *Information Technology—JPEG 2000 Image Coding System—Part 6: Compound Image File Format*, ISO/IEC 15 444-6, 2002.
[52] *Information Technology—JPEG 2000 Image Coding System—Part 4: Compliance Testing*, ISO/IEC 15 444-4, 2002.
[53] *Information Technology—JPEG 2000 Image Coding System—Part 5: Reference Software*, ISO/IEC 15 444-5, 2002.
[54] M. Adams and F. Kossentini, "JasPer: A software-based JPEG-2000 codec implementation," in *Proc. IEEE Int. Conf. Image Processing*, vol. 2, Oct. 2000, pp. 53–56.
[55] J. Kasner, M. Marcellin, and B. Hunt, "Universal trellis coded quantization," *IEEE Trans. Image Processing*, vol. 8, pp. 1677–1687, Dec. 1999.
[56] D. S. Zeng and W.and S. Lei, "Point-wise extended visual masking for JPEG-2000 image compression," in *Proc. IEEE Int. Conf. Image Processing*, vol. 1, Sept. 2000, pp. 657–660.
[57] W. Zeng, S. Daly, and S. Lei, "An overview of the visual optimization tools in JPEG 2000," *Signal Process.: Image Commun.*, vol. 17, pp. 85–104, Jan. 2002.
[58] T. Chinen, private communication.

David S. Taubman received the B.S. and B.E. (electrical) degrees from the University of Sydney, Sydney, Australia, in 1996 and 1998, respectively, and the M.S. and Ph.D. degrees from the University of California at Berkeley in 1992 and 1994, respectively.

From 1994 to 1998, he worked at Hewlett-Packard's Research Laboratories, Palo Alto, CA, joining the University of New South Wales, NSW, Sydney, Australia, in 1998 as Senior Lecturer in the School of Electrical Engineering. Since 1998, he has been involved with ISO/IEC JTC1/SC29/WG1, contributing substantially to the core JPEG2000 technology. He wrote the JPEG2000 Verification Model Software (VM3A) and the popular Kakadu implementation of the standard. He is also the coauthor, with M. Marcellin, of the book *JPEG2000: Image Compression Fundamentals, Standards and Practice* (Norwell, MA: Kluwer, 2002). His research interests include highly scalable image and video compression, inverse problems in imaging, perceptual modeling, and multimedia distribution.

Dr. Taubman was awarded the University Medal from the University of Sydney, the Institute of Engineers, Australia, Prize, and the Texas Instruments Prize for Digital Signal Processing, all in 1998. He has received two Best Paper awards from the IEEE Circuits and Systems Society in 1997 (for the 1996 paper "A Common Framework for Rate and Distortion Based Scaling of Highly Scalable Compressed Video") and from the IEEE Signal Processing Society in 2002 (for the 2000 paper "High Performance Scalable Image Compression with EBCOT").

Michael W. Marcellin (Fellow, IEEE) was born in Bishop, CA, on July 1, 1959. He received the B.S. degree (*summa cum laude*) from San Diego State University, San Diego, CA, in 1983 and the M.S. and Ph.D. degrees from Texas A&M University, College Station, in 1985 and 1987, respectively, all in electrical engineering.

Since 1988, he has been with the University of Arizona, Tucson, where he is a Professor of electrical and computer engineering. His research interests include digital communication and data storage systems, data compression, and signal processing. He has authored or coauthored more than 100 papers in these areas. He is a major contributor of technology to JPEG2000, the emerging second-generation standard for image compression. Throughout the standardization process, he chaired the JPEG2000 Verification Model Ad Hoc Group, which was responsible for the software implementation and documentation of the JPEG2000 algorithm. He is coauthor, with D. S. Taubman, of the book *JPEG2000: Image Compression Fundamentals, Standards and Practice* (Norwell, MA: Kluwer, 2002). This book is intended to serve as a textbook on image compression fundamentals as well as the definitive reference on JPEG2000.

Prof. Marcellin has served as Associate Editor for the IEEE TRANSACTIONS ON IMAGE PROCESSING and has served on the organizing committees and as session chair for numerous conferences. Recently, he served as Vice Chair of the program committee for IEEE Globecom 1997 and as Technical Area Chair of the program committee for the 1997 Asilomar Conference on Signals, Systems, and Computers. He has served as a member of the program committees for the 2000, 2001, and 2002 SPIE Conferences on Hybrid Image and Signal Processing, the 2000 IEEE Symposium on Geoscience and Remote Sensing, as well as the 1999, 2000, 2001, and 2002 Data Compression Conferences. He is a member of Tau Beta Pi, Eta Kappa Nu, and Phi Kappa Phi. He is a 1992 recipient of the National Science Foundation Young Investigator Award, and a corecipient of the 1993 IEEE Signal Processing Society Senior (Best Paper) Award. He has received teaching awards from NTU (1990, 2001), IEEE/Eta Kappa Nu student sections (1997), and the University of Arizona College of Engineering (2000). In 2001, he was named the Litton Industries John M. Leonis Distinguished Professor of Engineering at the University of Arizona. While at San Diego State University, he was named the most outstanding student in the College of Engineering.