Preserving Step Edges in Low Bit Rate Progressive Image Compression

Dirck Schilling and Pamela C. Cosman, *Senior Member, IEEE*

***Abstract*—With the growing importance of low-bandwidth applications such as wireless access to the Internet, images are often sent or received at low bit rates. At these bit rates they suffer from significant distortion and artifacts, making it difficult for those viewing the images to understand them. In this paper we present two progressive compression algorithms that focus on preserving the clarity of important image features, such as edges, at compression ratios of 80:1 and more. Both algorithms capture and encode the locations of important edges in the images. The first algorithm then transmits a standard SPIHT bit stream, and at the decoder applies a nonlinear edge-enhancement procedure to improve the clarity of the encoded edges. The second approach uses a modified wavelet transform to "remove" the edges, and encodes the remaining texture information using SPIHT. With both approaches, features in the images that may be important for recognition are well preserved, even at low bit rates.**

***Index Terms*—Edge preservation, edge-based wavelet transform, progressive compression.**

I. Introduction

DESIGNERS of image coders face a more complex task when designing progressive algorithms than nonprogressive ones. While a nonprogressive algorithm seeks to achieve the highest image quality at a single target bit rate, a progressive algorithm attempts to do the same across an entire range of bit rates. There are tradeoffs at each bit rate along the way. Image quality may be improved at a given bit rate, yet this may require a reduction in quality, relative to other algorithms, at another bit rate.

Image quality is often measured with peak signal-to-noise ratio (PSNR), and many progressive algorithms are designed with this measure in mind. Often low frequency information is sent first, delaying higher frequency information until later. For many images this approach works very well, particularly those consisting largely of low-frequency, smooth information such as natural scenes. For other image classes, such as those containing text or graphics, this prioritization performs poorly at low bit rates. The sharp edges and details of important features are often obscured by blotches and ringing artifacts caused by coarse quantization of the transform coefficients.

Compression algorithms may employ alternative tradeoff strategies to the strict minimization of mean squared error (see e.g., [1]). For example, the coder may spend more of its early bit rate to improve the appearance of specific image features deemed to be important, while representing other features at lower quality until later in the bit stream. Edges are one type of image feature whose importance for recognition has long been noted [2]–[4]. For certain image classes, such as those containing simple, sharply defined objects, text or graphics, enhancing the visual clarity of some edges early in the bit stream may allow the user to understand the basic content of the images, even at very low bit rates.

In this paper, we present two progressive image coders designed with the goal of improving the visual clarity of specific image features early in the progressive bit stream. Both algorithms capture the locations of important edges with an edge detection step, then encode these edges and transmit them to the decoder as part of the image header. The first, an *edge-enhancing* image coder (EEIC), follows the header with a standard SPIHT bit stream [5]. Its decoder uses the edge information to enhance (sharpen) the edges in the blurred, low-bit-rate image decoded from the SPIHT stream. The second coder improves upon the first by making use of the information contained in the edge packet when performing the wavelet transform. The edges are effectively removed from the image during the forward transform, and reinserted by the decoder during the inverse transform. We refer to this algorithm as a *feature-preserving* image coder (FPIC). We restrict our approach in this paper to handling step edges. However, both algorithms could be extended to more general edge models, such as that described in [6].

This paper is organized as follows. In Section II we describe the edge detection and coding procedures used in both algorithms, and the edge enhancement technique specific to EEIC. Section III describes FPIC, particularly its feature-preserving transform. In Section IV we provide compression results and a comparison of the two algorithms with existing approaches. We present our conclusions in Section V.

II. Edge-Enhancing Image Coder

In [7], we introduced a coder that combines SPIHT (or any other progressive wavelet coder) and edge enhancement, with the goal of allowing faster human recognition of the progressively decoded images. A block diagram for this edge-enhancing image coder (EEIC) is shown in Fig. 1. The source image passes through an edge detector, which identifies

Manuscript received March 19, 2001; revised June 25, 2003. This work was supported by the NSF under Grants MIP-9617366 and MIP-9624729 (CAREER), by the Center for Wireless Communications at UCSD, and by the CoRe program of the State of California. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. David S. Taubman.

D. Schilling was with the Department of Electrical and Computer Engineering, University of California, San Diego, La Jolla, CA 92093-0407 USA. He is now at ViaSat, Inc., Carlsbad, CA 92009 USA (e-mail: dirck.schilling@viasat.com).

P. Cosman is with the Department of Electrical and Computer Engineering, University of California, San Diego, La Jolla, CA 92093-0407 USA (e-mail: pcosman@code.ucsd.edu; http://www.code.ucsd.edu/cosman/).

Digital Object Identifier 10.1109/TIP.2003.819308

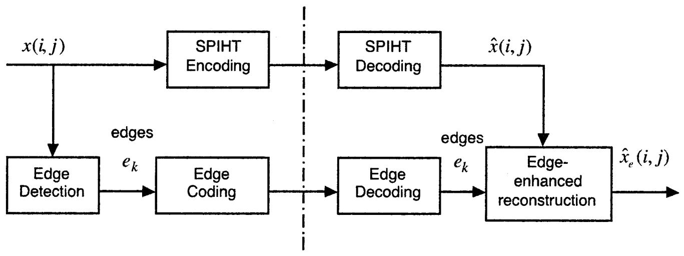


Fig. 1. Block diagram of edge-enhancing image coder.

prominent edges likely to aid in human recognition. Lines are extracted from the identified edge pixels, and the line segment endpoints can be encoded using standard techniques for line graphics, such as modified multiring chain coding [8]. The encoded edge information is sent as part of the image header. Encoding then proceeds in the usual SPIHT fashion. The decoder enhances edges in the output image by combining the edge location information with pixel intensity information available from the progressive SPIHT bit stream.

A. Edge Extraction

The edge enhancement step is independent of the algorithms used for edge detection and line extraction. Edge detection is performed by a robust and effective method known as SUSAN (Smallest Univalue Segment Assimilating Nucleus) [9]. This method is based on a nonlinear local region-finding procedure which has several advantages over derivative-based methods such as Sobel or Canny edge detection. It is insensitive to noise, provides good edge localization independently of mask size, and reports sharp corners with good connectivity. Line segment extraction is carried out using methods described by Rosin and West in [10]. The current implementation approximates curved edges by sets of linked straight line segments, however the enhancement technique is applicable to higher-order representations such as arcs, ellipses and polynomials.

The primary goal of our approach is to improve recognition by enhancing only the most "important" edges in the image. There is no widely accepted measure for the importance of an edge, however in most situations a longer edge will convey more information than a shorter one. The edge detection methods described above intrinsically filter out much noise, but still report short, sharply defined edges. Such short edges often correspond to fine details or cluttered areas in the image. Coding these edges is costly, and enhancing them is not likely to improve recognition significantly. The edge detection procedure therefore includes a step to remove short edges.

It would be possible to incorporate other definitions of what constitutes an important edge. Rohaly *et al.* discuss metrics that predict object detection [11], while Giraudet *et al.* studied the effect of different spatial frequencies on object recognition [12]. One could consider important edges to be those that keep their position in a multiscale representation. Alternatively, in some cases one may be searching for images with specified contents. In this case important edges might be ones which are similar to the desired contents (see, e.g., [13]).

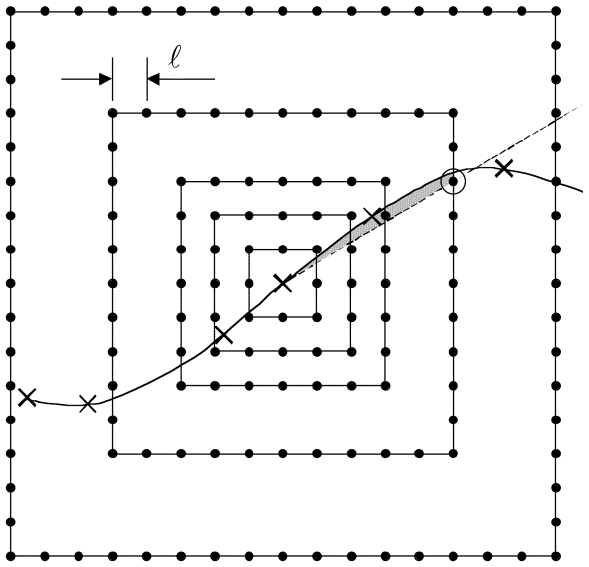


Fig. 2. Multiring chain coding grid structure, with the radius 13 ring omitted for clarity. Dots indicate indexed grid points; X's are points on the input curve. The circled grid point indicates the next output point, and the shaded area is the error between the input and output curves for the current step.

B. Edge Coding

One efficient method of encoding edge information is the modified multiring chain code with Fibonacci spacing described in [8]. This method was originally proposed for coding human signatures, but can be applied to any line-based graphics. Its advantages include its relatively simple implementation, and the fact that it allows a tradeoff between cost and distortion. Fig. 2 summarizes the method. Given a base unit of length l, a structure of concentric square rings is defined whose radii are the product of l and the Fibonacci sequence 1, 2, 3, 5, 8, and 13. Grid points are located along each ring with a spacing of l, yielding a total of 256 grid points. The input curve consists of an ordered sequence of points $X = \{x_0, x_1, \dots, x_N\}$, which is approximated by the output sequence $Y = \{y_0, y_1, \dots, y_M\}$. At each step in the procedure, the multiring structure is positioned with its origin on the previously coded output point y_{j-1}. The intersection of the input curve with each ring is examined in succession from the outer ring inward. The nearest grid point to this intersection is chosen as the next candidate output point y_{cand}. If no point along X between y_{j-1} and y_{cand} is further than l from the line segment $(y_{j-1}, y_{\text{cand}})$, then y_j becomes y_{cand} and the encoder outputs its multiring index.

This approach encodes the input curve nearly losslessly, in that no point on the input curve is further than l from the approximating output curve. Points on the output curve are a maximum distance of $13\sqrt{2}l$ apart, so the choice of l represents a tradeoff between the accuracy of the output curve and the number of output points (number of ring indices) required to transmit it. For our implementation, we used $l = 1.5$ pixels.

The ring indices are entropy coded to take advantage of the predictability in continuous curves. In our implementation, after each ring index is sent, the grid points are renumbered such that the indices corresponding to "straight ahead"—0, 8, 24, 48, 88, and 152—are positioned along the line extending in the direction of the previously encoded line segment. The remaining in-

TABLE I
EDGE PACKET CODING COSTS FOR SEVERAL IMAGES. FIGURES IN PARENTHESES ARE BITS PER ORIGINAL EDGE POINT

Image	Orig. Edge Points	Huffman bits (bpop)	Untrained Arithmetic bits (bpop)	Trained Arithmetic bits (bpop)
airplane	243	1593 (6.6)	1683 (6.9)	1534 (6.3)
angels	296	2058 (7.0)	2129 (7.2)	1976 (6.7)
apples	1018	5866 (5.8)	5979 (5.9)	5860 (5.8)
aranda	665	3643 (5.5)	3709 (5.6)	3498 (5.3)
bora	386	2335 (6.0)	2486 (6.4)	2388 (6.2)
mtilt	65	963 (14.8)	962 (14.8)	1011 (15.6)
orange	402	2337 (5.8)	2451 (6.1)	2328 (5.8)
picnic	241	1903 (7.9)	1994 (8.3)	1988 (8.3)
wbird	374	2390 (6.4)	2441 (6.5)	2348 (6.3)
wgrass	45	837 (18.6)	790 (17.6)	830 (18.4)
Mean	**373.5**	**2393 (8.43)**	**2462 (8.53)**	**2376 (8.45)**

dices are shifted by a corresponding amount around their respective rings. The result is that each index reflects an angular offset relative to the previously coded line segment, rather than an absolute offset. The distribution of index occurrences is thereby skewed toward smaller angular changes, improving compression of the indices. The 256 ring indices, plus a 257th index to indicate "end of edge," are encoded using integer arithmetic encoding similar to the technique in [14].

We obtain a small additional improvement in coding efficiency by representing each ring index as a combination of three values: the ring on which the index lies, the number of gridpoints away from "straight ahead" that it lies, and whether it lies to the left or right of straight ahead. Each component is arithmetically encoded with a context of one or two previous values. The improvement can be attributed to the reduced number of contexts that the adaptive encoder must learn for each value.

1) Coding of Start Points and Single Segments: The multiring chain coder must be initialized with the starting point of each edge. For this purpose we employ arithmetic differential offset coding of the edge start points. The encoder computes the means of the horizontal and vertical offsets between successive edge start points, and transmits these along with the total number of edges in an edge packet header. The encoder then arithmetically encodes the differences between the means and the horizontal and vertical offsets to each successive edge from the previous one.

The multiring chain code is particularly effective for long, smooth curves. It performs less well for edges consisting of only one line segment, since no information is available from which to predict the angle of the line segment. For this reason we use differential offset coding to transmit both the start and end points of these edges.

2) Edge Coding Performance: Table I shows the cost of coding the edge information for several test images. Costs are shown both in total bits, and in *bits per original point* (bpop).

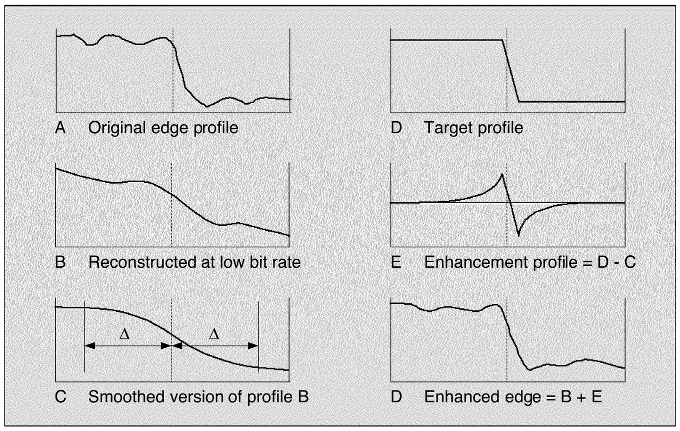

A Original edge profile
D Target profile
B Reconstructed at low bit rate
E Enhancement profile = D - C
Δ
Δ
C Smoothed version of profile B
D Enhanced edge = B + E

Fig. 3. Edge enhancement procedure used in EEIC.

These images ranged in size between 256 and 512 pixels in each dimension, so transmitting each original edge point uncompressed would require $2\log_2(512) = 18\ \text{bits}$. Three methods for coding the ring indices are compared: Huffman coding, arithmetic coding with no initial data, and arithmetic coding with initial index probabilities computed from several training images. In both the Huffman and trained arithmetic cases, the training sets used to generate index probabilities did not include the test images. In the arithmetic cases, the coder state included context from prior indices. Arithmetic coding with initial probabilities performed best in most cases except those with the smallest edge packets. Untrained arithmetic coding performed worse than Huffman coding, due primarily to the small amount of data to be coded: the arithmetic coder has too little time to adapt. Based on these results, Huffman coding is a good choice because of its simplicity and performance.

C. Edge Enhancement

We now describe the approach used by the decoder to enhance the edges transmitted in the edge packet. For the current implementation we assume that the edges of interest are predominantly of the step or single-sided ramp type. Extensions are possible, however, which allow other types of edges such as narrow ridges to be handled. The basic principle of the edge enhancement procedure is illustrated in Fig. 3.

The original edge, shown in profile in part A of Fig. 3, is reconstructed by the wavelet coder at a low bit rate as B. This is the version of the edge that is obtainable from the SPIHT (or other wavelet coder) bit stream alone, without making use of the edge information received. The decoder smooths B to obtain C (we used a 20-point Gaussian filter); alternatively it could retain an earlier version of B in memory. The decoder obtains the location of the edge from the image header. The intensity values of profile C at a distance Δ from the edge on either side, I_1 and I_2, are used to define a target profile D. In the current implementation, this profile is a nearly ideal step, where I_1 and I_2 are the step's two values, with a 1-pixel-wide ramp between them. The difference between D and C yields E, the enhancement profile. Finally, E is added to the reconstructed profile B, resulting in the enhanced edge profile F.

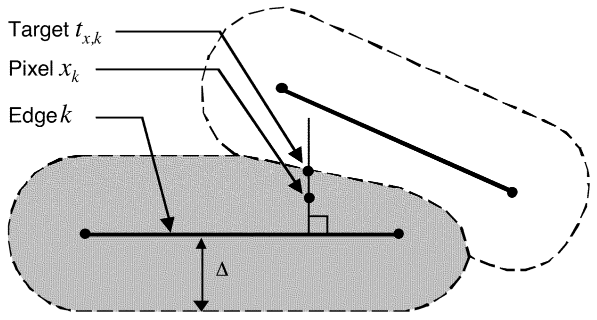


Fig. 4. Pixel enhancement for edge k. Pixels in shaded area belong to k.

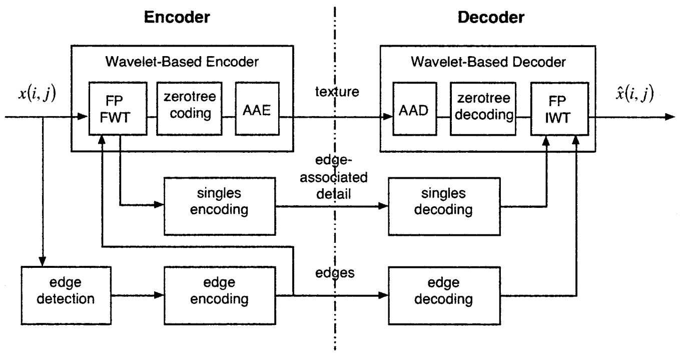


Fig. 5. Block diagram of FPIC.

In practice, several complications to the above procedure arise, for instance when two or more edges fall within a distance 2Δ of one another. The intent of the target profile D is to determine valid pixel intensities to use for the "high" and "low" sides of the edge. Thus, the desired target pixel intensities should not be drawn from the opposite side of a neighboring edge. To handle this, the decoder computes a distance transform (such as the 5–7–11 Chamfer algorithm [15]), labeling each image pixel with its distance from the nearest edge and the identity of that edge. For each edge, the enhancement procedure is performed on all pixels within Δ of the edge and not closer to another edge. Fig. 4 illustrates the regions of influence of two neighboring edges. For each pixel x_k lying within edge k's region of influence, an enhancement target value is determined, i.e., the value from profile D corresponding to x_k's location. This target value is taken as the intensity of the target pixel $t_{x,k}$, where $t_{x,k}$ is the pixel in k's region of influence that is farthest from k along the normal line from x_k to k.

EEIC is successful in improving the clarity of edges in highly compressed images; in Section IV we provide examples of its output. However, EEIC and related edge-based coders such as [16] function only as an enhancement procedure at the decoder: the edge information extracted at the encoder is not used at all in encoding the transform coefficients. Because of this, there is redundancy between the edge packet and the coefficients. A further shortcoming is EEIC's reliance on the fixed parameter Δ for determining the width of an edge. In Section III, we discuss a coder designed to address these issues.

III. Feature-Preserving Image Coder

Ideally, we wish to avoid any redundancy in encoding the various types of information contained in an image. The feature-preserving image coder (FPIC) [17] described in this section treats an image as being composed of three types of information: *edges*, *texture*, and *edge-associated detail*. Each component is encoded progressively by a method tailored to its specific characteristics. This approach allows a great deal of flexibility at the encoder in balancing the bits budgeted for, and thus the resulting quality of, each component.

A block diagram of FPIC is shown in Fig. 5. As with EEIC, the encoder determines the locations of important edges and transmits them to the decoder. Edge detection, line segment extraction and edge encoding are performed in almost the same manner as for EEIC, with one small improvement. Occasionally, input edges may contain exceptionally long line segments (e.g., >50 pixels). The multiring chain coder handles these by repeating the same ring index numerous times, which can be inefficient, and may allow small but visible deviations in the straightness of these edges. Instead, we send an escape code when encountering such a segment, and transmit the x and y offsets for the segment. Ring coding then resumes as before. This modification improves the appearance of long, straight edges, and it often reduces their cost in bits.

In contrast to EEIC, FPIC makes use of the edge locations when encoding the remaining image information. The input image is passed through a wavelet-based encoder which also receives the encoded (lossy) edge locations as input. This encoder, like SPIHT, consists of a forward wavelet transform, a quantization (zerotree coding) step, and arithmetic encoding of the quantized wavelet coefficients. In place of the standard wavelet transform employed by SPIHT, however, FPIC uses a new feature-preserving transform. This transform separates the image into its three components: it removes information about the edge locations (which has already been transmitted), and divides the remaining information into texture and edge-associated detail.

A. Feature-Preserving Wavelet Transform

The wavelet transform employed in FPIC improves the efficiency with which the texture can be encoded, by reducing the energy caused by the edges in the high-frequency bands of the transform. It also extracts the edge-associated detail information, contained in isolated coefficients near certain edges, and passes this out for separate encoding.

1) Forward Transform: A standard one-level, one-dimensional (1-D) forward wavelet transform is illustrated in Fig. 6. The input signal x contains an ideal step function. As the lowpass filter passes over the step, its averaging properties cause it to smear the step across several coefficients in the transform's low band. Similarly, as the highpass filter encounters the step, it results in several spikes in the transform's high frequency band. If these spikes are coarsely quantized or zeroed during low bit rate transmission, the step becomes distorted.

Fig. 7 illustrates the operation of the forward feature-preserving wavelet transform. In this case, both the forward and inverse transforms know the location of the ideal step, or edge. In this paper, we specify the location of an edge by using pixel cracks, also called crack edges. An edge of any width can have its location specified as being centered between two

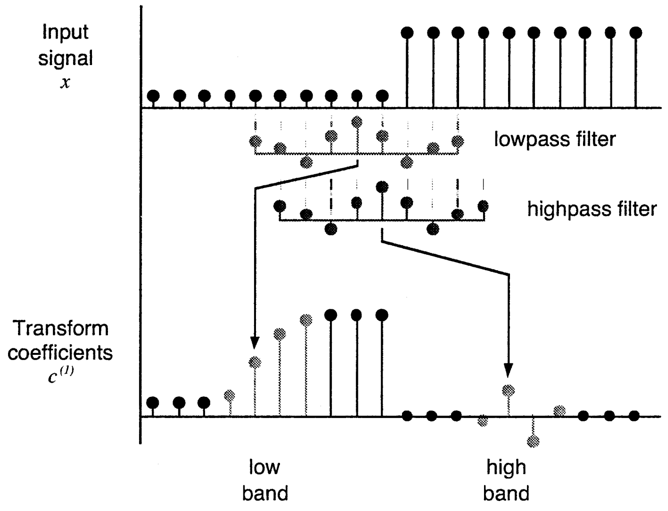


Fig. 6. Schematic of a standard one-level wavelet transform on a 1-D input signal containing an ideal step function.

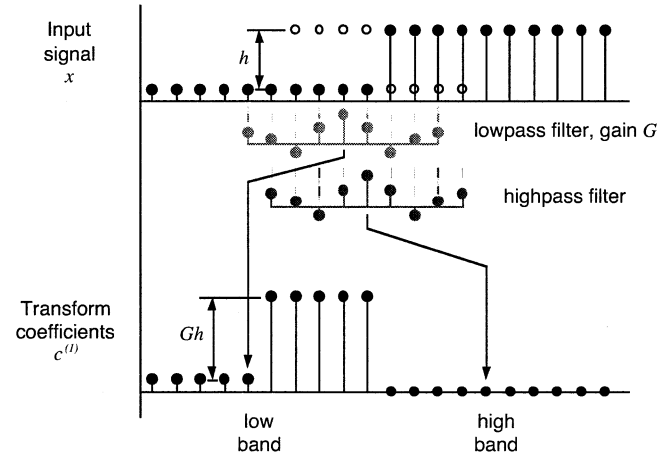


Fig. 7. Schematic of a one-level feature-preserving wavelet transform on a 1-D input signal containing an ideal step function.

pixels, rather than being centered on a pixel. This method of specifying edge location can be used for step edges as well as ramp edges and other edge models. In the forward transform, as the lowpass filter crosses the edge, the height h of the step is subtracted from each input value beyond the edge before that value is fed to the filter. Then, after the center tap of the filter passes the edge, h is added to each input value prior to the edge. In this way, all values seen by the filter at each position have had the step removed from them. For the lowpass filter, this means that the ideal step is reproduced without smearing in the transform's low band. Likewise, since the highpass filter never sees the step, no spikes due to that step occur in the high band. Even if the coefficients in either band are coarsely quantized, the sharpness of the step remains unaffected. The step's height has been preserved in the low band (scaled by the gain of the lowpass filter), but, because of the downsampling, its precise location is now contained only in the separate edge information, not in the transform coefficients. The edge has been removed from the high band coefficients.

2) Inverse Transform: The edge removal procedure described above is a linear one, and so can be described fully by a system of equations of the form $F_e \cdot x = c$, where c is the vector of transformed coefficients prior to reordering into high and low bands. F_e is an $N\times N$ matrix consisting of the base wavelet transform matrix F plus an $N\times N$ edge-removal matrix E. It is therefore possible to reconstruct the original input vector x from the coefficients by matrix inversion with $\hat{x} = {F_e}^{-1} \cdot c$.

Given an input signal x of length N, with lowpass filter f and highpass filter g, the standard one-level octave-band wavelet transform is computed by

$$\begin{pmatrix} f^T & & & & \\ & g^T & & & \\ & & f^T & & \\ & & & \ddots & \\ & & & & g^T \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_{N-1} \end{pmatrix} = \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_{N-1} \end{pmatrix}$$

or $F \cdot x = c$. Note that the filter taps in f and g must be properly reflected back on themselves at the signal boundaries. For the following discussion we assume odd-length filters, with $f = [f_0\, f_1\, f_2\, f_3\, f_4]^T$ and $g = [g_0\, g_1\, g_2]^T$. In the edge-removal procedure applied during the forward transform, the transformed coefficients are computed as follows:

$$\begin{aligned} c_0 &= f_2x_0 + (f_1+f_3)x_1 + (f_0+f_4)x_2 \\ c_1 &= g_0x_0 + g_1x_1 + g_2x_2 \\ c_2 &= f_0x_0 + f_1x_1 + f_2x_2 + f_3x_3 + f_4x_4 \\ &\vdots \\ c_{s-2} &= f_0x_{s-4} + f_1x_{s-3} + f_2x_{s-2} + f_3x_{s-1} + f_4x_s \\ &\quad - hf_4 \\ c_{s-1} &= g_0x_{s-2} + g_1x_{s-1} + g_2x_s - hg_2 \\ c_s &= f_0x_{s-2} + f_1x_{s-1} + f_2x_s + f_3x_{s+1} + f_4x_{s+2} \\ &\quad + h(f_0+f_1) \\ c_{s+1} &= g_0x_s + g_1x_{s+1} + g_2x_{s+2} \\ &\vdots \\ c_{N-1} &= (g_0+g_2)x_{N-2} + g_1x_{N-1} \end{aligned}$$

where the step occurs between the input values x_{s-1} and x_s, and has height $h = x_s - x_{s-1}$. In this example, s is even. Stated in matrix notation, and substituting for h, this is

$$F \cdot x + \left(\begin{array}{cc|cc} 0 \;\cdots & 0 & 0 & \cdots\; 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & f_4 & -f_4 & 0 \\ 0 & g_2 & -g_2 & 0 \\ \hline 0 & -(f_0+f_1) & (f_0+f_1) & 0 \\ 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 \;\cdots & 0 & 0 & \cdots\; 0 \end{array}\right) \times \begin{pmatrix} x_0 \\ \vdots \\ x_{s-1} \\ \hline x_s \\ \vdots \\ x_{N-1} \end{pmatrix} = c.$$

Denoting the edge removal matrix as E, the transform becomes $(F+E)\cdot x = c$. The one-level inverse transform is therefore given by $\hat{x} = (F + E)^{-1} \cdot c$. The effect of additional edges is additive, so that for L edges, $(F+E_0+E_1+\cdots+E_{L-1})\cdot x = c$, and $\hat{x} = (F + E_0 + E_1 + \cdots + E_{L-1})^{-1} \cdot c$. The combined edge removal matrix is $E = E_0 + E_1 + \cdots + E_{L-1}$, and the overall transform matrix is then $F_e = F + E$. Note that, as with the base transform matrix F, care must be taken when constructing E to properly reflect the filter taps near the matrix boundaries.

Computational Complexity of the Inverse Transform: As described above, the inverse transform requires inverting an $N \times N$ matrix for each N-length signal, i.e., each row and column of an image. Although matrix inversion in general is approximately an $O(N^3)$ process, inversion of the overall transform matrix F_e can be accomplished by methods costing substantially less than $O(N^3)$, since F is constant, and each E_i contains only two nonzero columns. The Sherman-Morrison formula can be applied [18], reducing the cost when the number of edges in a given input line is small in relation to N. Another possibility is to use matrix inversion only on the portions of the input signal affected by an edge. That is, only submatrices of F_e containing the edges need be inverted.

Computational complexity can further be reduced by precomputation. The matrix F_e contains only information about the edge locations, not their heights, so that its inverse is valid for any signal having the same edge positions. The inverses of submatrices of F_e can be precomputed for common edge combinations. A precomputed submatrix for a lone edge, and submatrices for two edges within one to K pixels of one another, given filter length K, would be applicable to many commonly occurring cases. The overall inverse matrix F_e can then be assembled at run-time by reading these submatrices from lookup tables.

B. Edge Preprocessing

The wavelet transform is clearly dependent on an accurate initial knowledge of the edge locations. If an edge is reported to lie several pixels away from the actual location of the sharpest gradient in image intensity, the value of h, taken as the difference between the image intensities on both sides of the reported edge, will not be equal to the true height of the edge. As a result, the benefit of subtracting and adding h during the transform will be lost or reduced. Recall that the lossy edge coding method employed by FPIC allows the encoder to trade off cost and accuracy of the edges. To allow for lossy edges, FPIC incorporates an edge adjustment step prior to the wavelet transform. This step successively interpolates pixel values near each reported edge using values farther from the edge. The edge adjustment step functions as follows. A list of all image pixels that are within a distance n pixels of any edge is created, and sorted in order of decreasing distance from the nearest edge. In most of our tests, n was chosen to be 3. For each pixel p in the list, an adjusted intensity value I_p is computed, where I_p is the average of p's eight-neighbors (weighted by distance from p) that are both farther away from any edge than p, and are not separated from p by an edge. If I_p differs from the original pixel intensity by more than a threshold (which we set at 15), the original value is replaced by I_p. The effect of this adjustment is to shift the locations of sharp intensity gradients spatially by a few pixels, to match the lossy edge locations as encoded. In this way the benefit of the edge removal process for texture coding can be achieved, with little loss in overall image quality.

The edge adjustment step is useful at the lowest bit rates (for example, 0.05 to 0.2 bpp) which are the recognition bit rates for which FPIC is intended. If the edge adjustment is not done, FPIC still provides some advantage at these low rates. At higher rates (for example, greater than 0.3 bpp), the adjustment step does not provide an advantage, and can ultimately prevent the algorithm from converging on lossless or perceptually lossless quality as the bit rate increases. Thus, FPIC should be used with the edge adjustment step if recognition at low rates is the main goal, and progression all the way to lossless quality is not important. FPIC can be used without edge adjustment if recognition at low rates is desirable and ultimately convergence on lossless quality is also desired. This would somewhat reduce the effectiveness of FPIC at low rates. Alternatively, FPIC can be used with edge adjustment, and a residual coding step can be used at high rates, to keep the low-rate benefit of FPIC while converging on lossless quality. This would require a greater lossless file size than with the previous option.

C. Edge-Associated Detail

At each level of the forward transform, the current map of edge locations is subsampled to correspond with the current low-low (LL) band of the transform, using a procedure like that employed in [19]. When edges in the original image are close together, a situation may occur after one or more transform levels where an LL-band pixel is surrounded on each side by an edge. If such a pixel lies at an even-numbered position, it is encountered by the lowpass filter, and so is placed in the low band after filtering. If it lies at an odd-numbered position, however, it is encountered by the highpass filter, and would have to be represented as an isolated, or *single*, coefficient in the high band. Its statistics do not match those of the high-band coefficients, though. These single coefficients contain the information about image pixel intensity near closely spaced edges, which we call edge-associated detail.

It is desirable to employ a separate coding method for singles. This is simplified by the fact that both the encoder and decoder possess the same information about edge locations, and therefore know the location of each single, and when it arises during the transform. The only information that need be transmitted is the magnitude of the single. FPIC encodes this information progressively. Singles from each transform level are scaled to the same dynamic range, and bitplane encoded using adaptive arithmetic coding. For progressive coding, individual bitplanes of the edge-associated detail can be interspersed between bitplanes of the texture information and information about edges sorted by length.

IV. Results and Discussion

In this work we are primarily interested in the low end of the bit rate progression, where recognition is likely to take place, rather than the high-rate, high-quality end. Our work in [20], [21] showed that human observers can recognize and respond

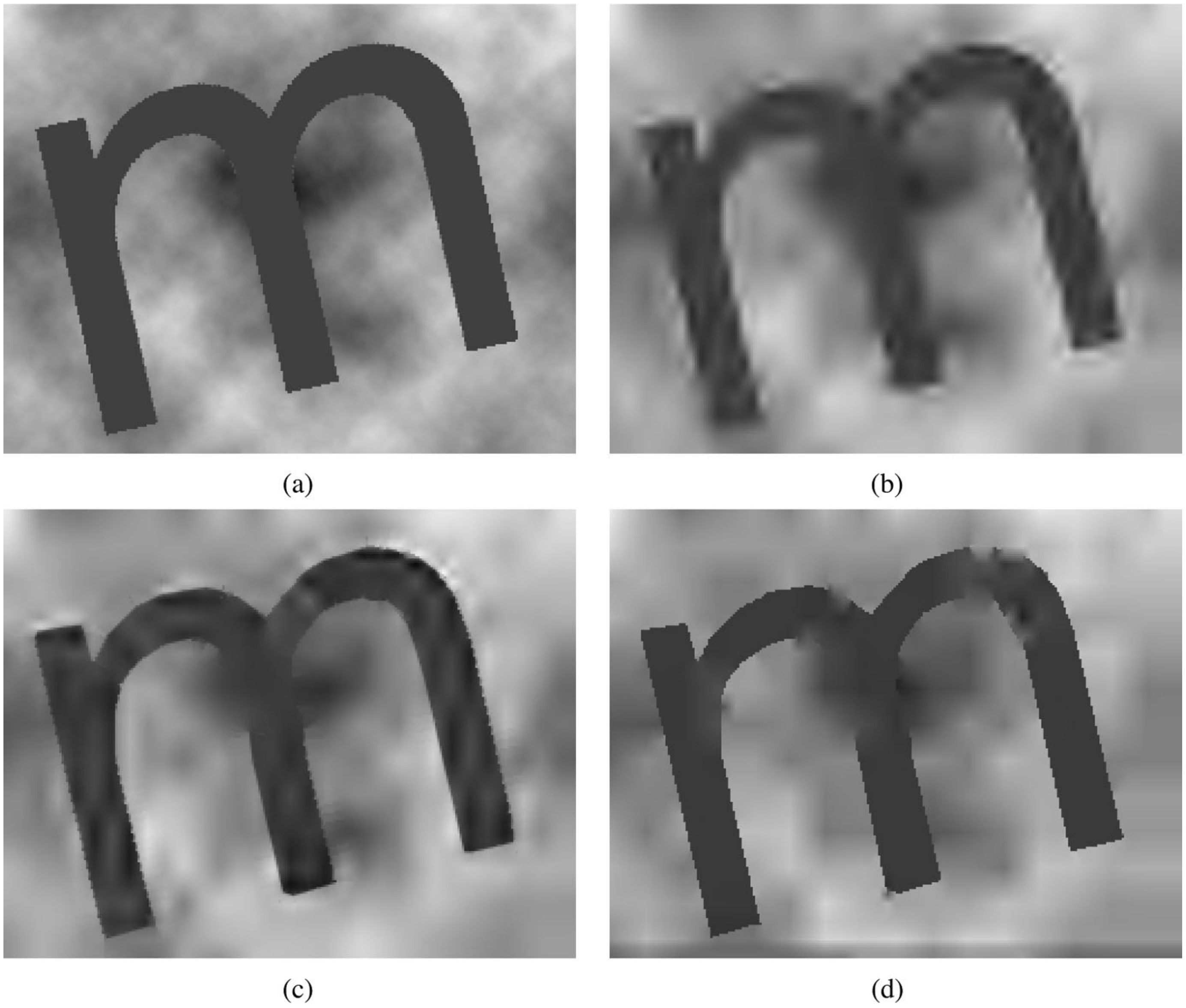


Fig. 8. Comparison of SPIHT, EEIC and FPIC. (a) original 256×201 image, (b) SPIHT-coded image at 0.05 bpp, (c) EEIC-coded image at 0.05 bpp. There were 67 original edge points. The wavelet coding uses 0.034 bpp and the arithmetically encoded edge data requires 0.016 bpp. (d) FPIC-coded image at 0.05 bpp. Edge data uses 0.019 bpp, singles 0.001 bpp and texture 0.03 bpp.

to image content at bit rates of 0.05 to 0.1 bpp, using the SPIHT algorithm, for many natural image classes. These results suggest that it is important for algorithm designers to consider very low bit rates when designing compression algorithms for fast browsing tasks and wireless Internet access. The bit rates chosen for the following comparisons reflect this focus.

Examples of the output of EEIC and FPIC are shown in Figs. 8–11. The 256×201 input image in Fig. 8(a) is characterized by a simple, sharply defined object superimposed on a smoothly varying background. This situation is found in many images containing symbols and graphics. Fig. 8(b) shows the image compressed by SPIHT to 0.05 bpp, or 160:1. The EEIC-coded image at the same bit rate is shown in Fig. 8(c). The edge information transmitted by EEIC contained 67 edge points costing 840 bits, or 33% of the total bit budget at this compression ratio. The object's edges in the SPIHT image suffer from the artifacts typical of standard wavelet coders, while the edges in the EEIC-coded image are noticeably improved. For the FPIC-compressed image shown in Fig. 8(d), 37% of the bit budget was spent on edges, 3% on singles, and the remainder on texture. This image provides the closest visual similarity to the original image.

Fig. 9 illustrates the comparison for an image containing both natural objects and text. The original 512×512 image (Fig. 9(a)) was composed of text superimposed on the "peppers" image. Edge detection was restricted to a 512×110 pixel region surrounding the text, under the assumption that a region classifier such as [22] could provide the text's location. The compressed images in this example are all shown at 0.04 bpp, or 200:1. The SPIHT-compressed image is shown in Fig. 9(b). Fig. 9(c) shows the EEIC-coded image, for which the 290 edge points consumed 3035 bits, or 29% of the total bit rate. For the FPIC-compressed image (Fig. 9(d)), 30% of the bit budget was spent on edges, 5% on singles, and the remainder on texture.

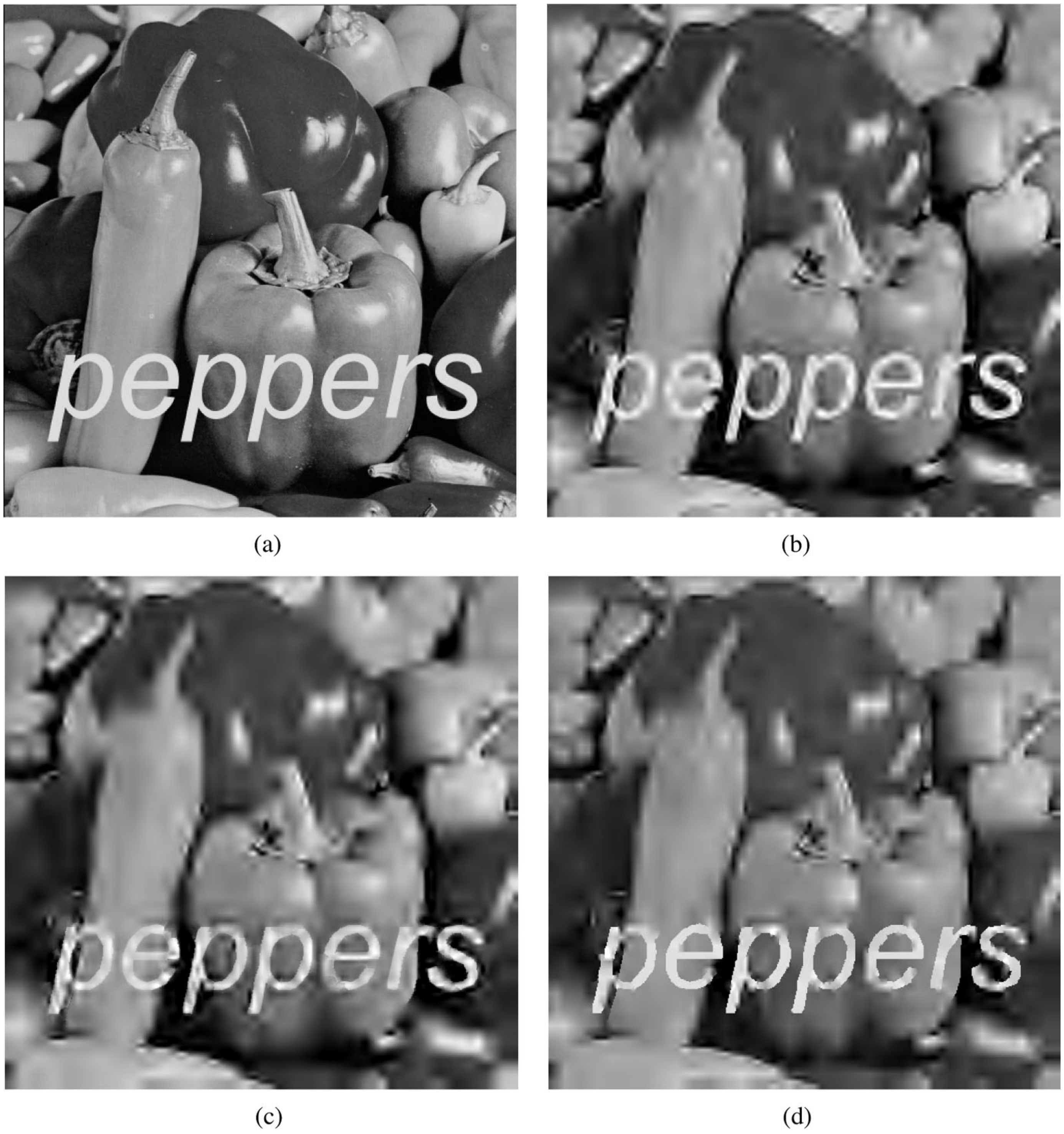


Fig. 9. Comparison using Peppers with added text. (a) original 512 × 512 image, (b) SPIHT-coded image at 0.04 bpp, (c) EEIC-coded image at 0.04 bpp. There were 290 edge points. The wavelet coding uses 0.028 bpp and the edge data requires 0.012 bpp. (d) FPIC-coded image at 0.04 bpp. Edge data uses 0.012 bpp, singles 0.002 bpp and texture 0.026 bpp.

Enlargements of the text region, and of the lossily encoded edges, are shown in Fig. 10. Again, EEIC provides improved clarity over SPIHT, and FPIC yields further improvement. Note that, since EEIC and FPIC are progressive, and the edge information packet is fixed in size, the percentage of the bit budget consumed by the edge information decreases as the progression continues. At high bit rates—corresponding to high image fidelity—the size of the edge packet becomes negligible in comparison to the overall file size.

Both the accuracy of the edge map, and the efficiency with which it can be encoded, are strongly dependent on the edge detection and line extraction steps. The output of these steps varies significantly with the choice of several thresholds and parameters. In some applications, however, *a priori* knowledge

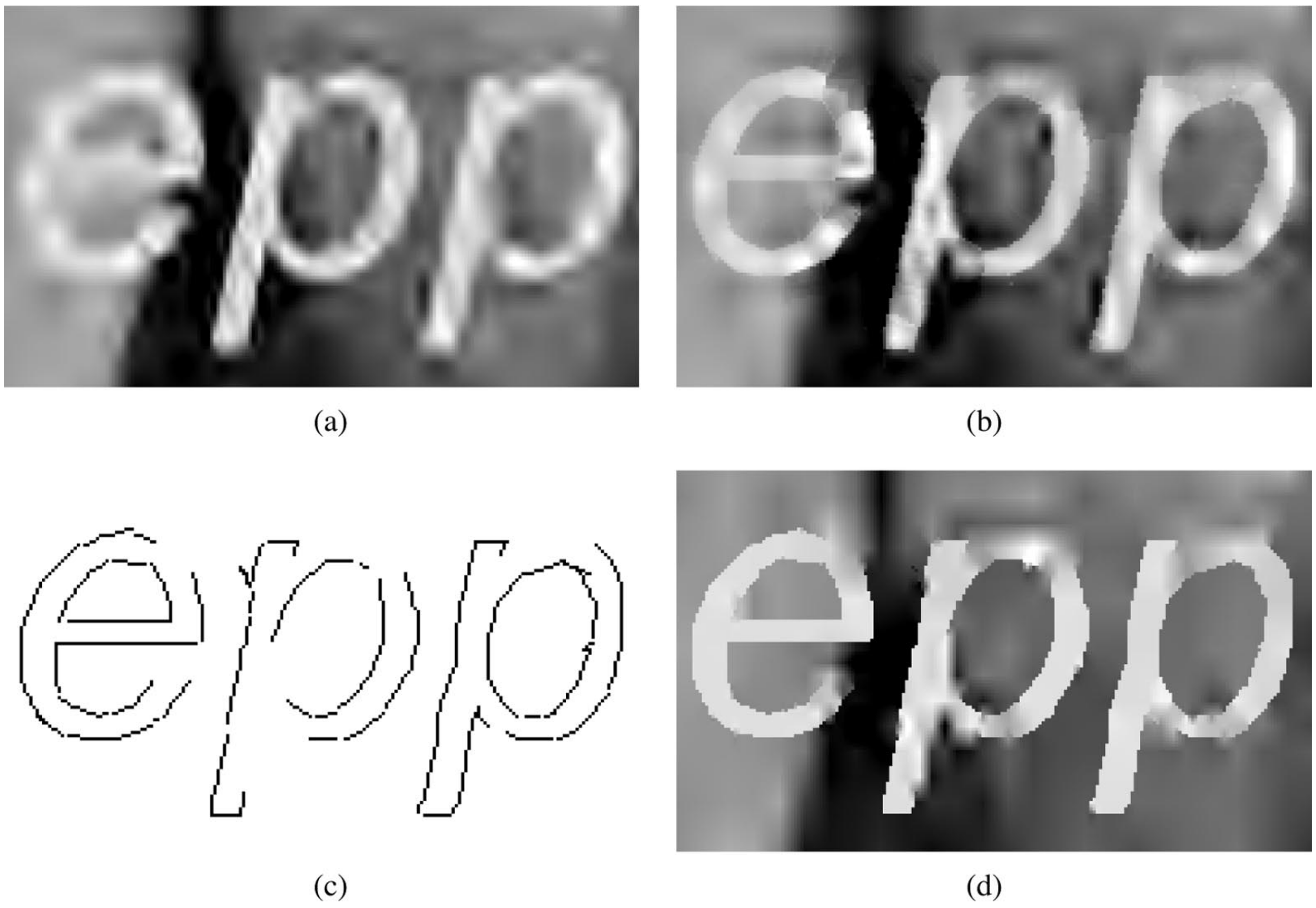


Fig. 10. Enlargement of the Peppers comparison at 0.04 bpp. (a) SPIHT-coded image, (b) EEIC-coded image, (c) Lossily encoded edges used by both EEIC and FPIC. (d) FPIC-coded image.

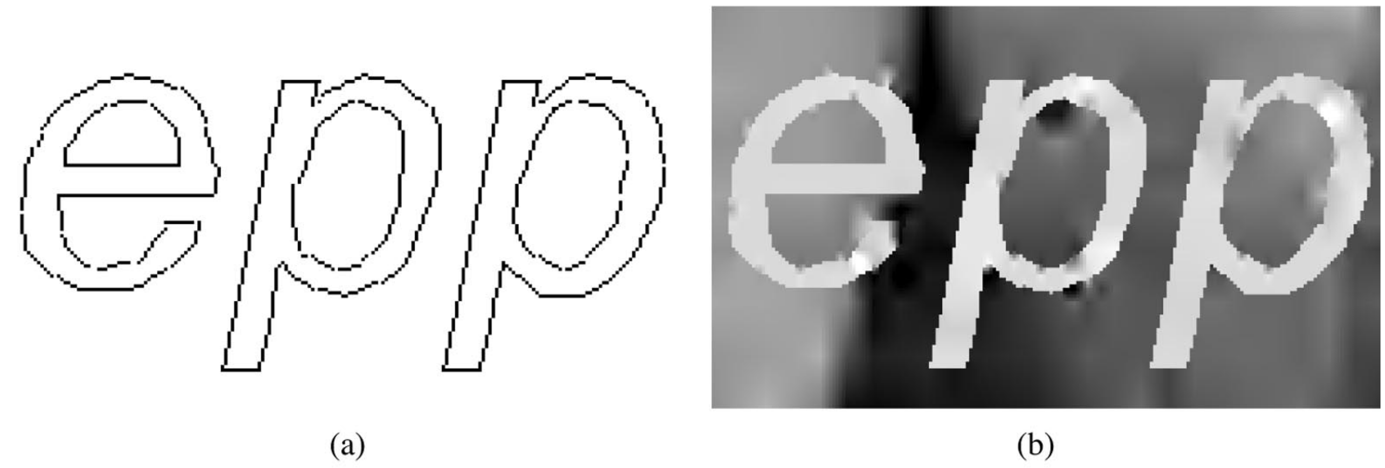


Fig. 11. Enlargement of compressed image with manually defined edges. (a) Edges after lossy encoding, requiring 0.0095 bpp. (b) FPIC-coded image at 0.04 bpp.

may be available of the text characters' locations and fonts. By manually defining the edges input to FPIC, we can approximate the best result obtainable by the algorithm, given that the text edge locations are known. Fig. 11 shows the manually-defined input edges for the Peppers image (after lossy encoding), and the FPIC-compressed result at 0.04 bpp. The edges in this case required only 0.0095 bpp (23.7%) versus 0.0122 bpp (30.4%) for the automatically-detected case. The visual appearance of the text is improved due to the more accurately defined edges, and the bit rate savings can be deployed toward coding the image texture.

FPIC is similar to an edge-based compression algorithm presented by Mertins in [19], however there are several significant differences. First, FPIC allows edges to be coded lossily,

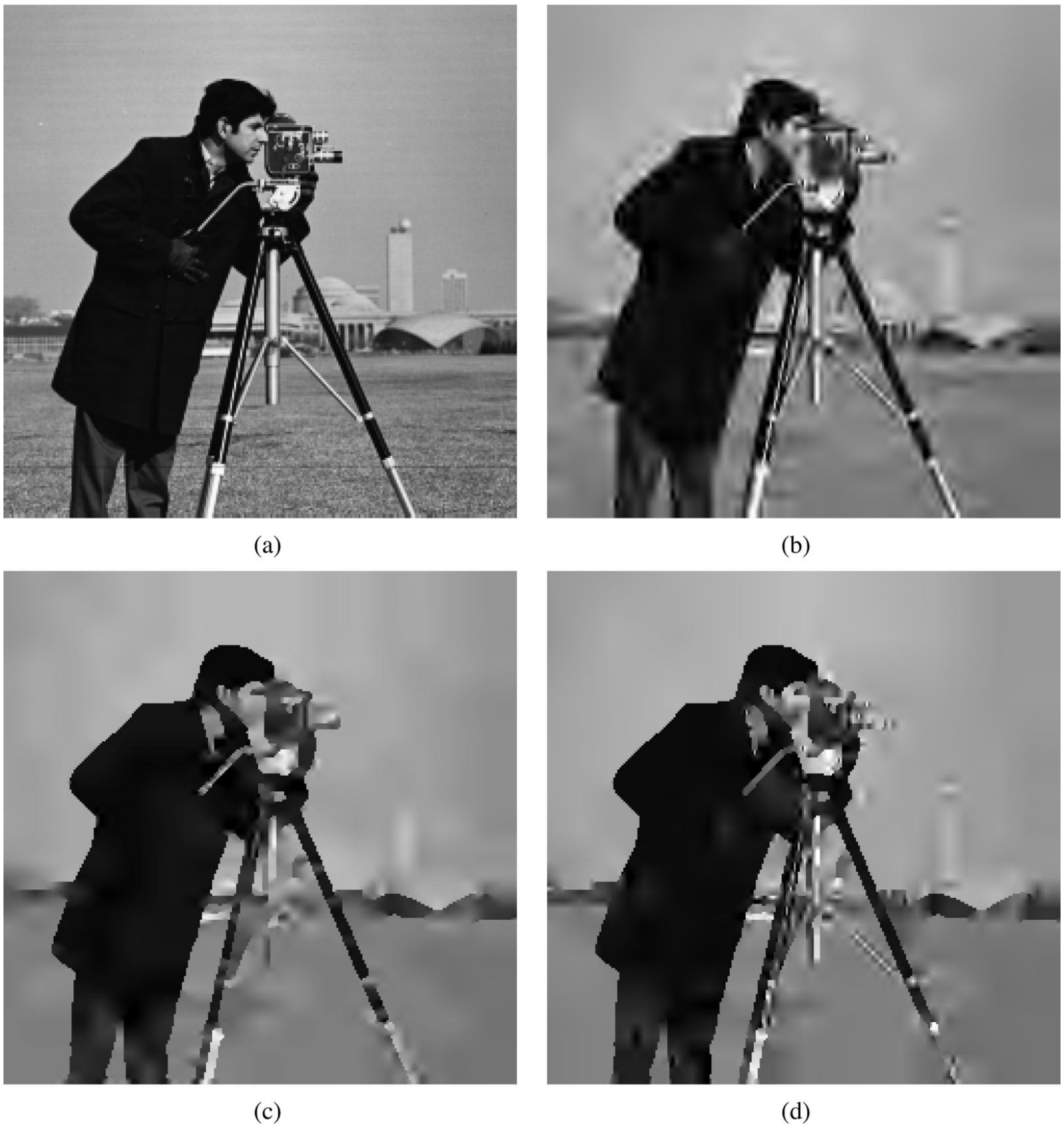


Fig. 12. (a) Original 256 × 256 "cameraman" image, (b) compressed by SPIHT to 0.10 bpp, $\mathrm{PSNR} = 24.37$ dB; (c) as reported in [19], 0.10 bpp, $\mathrm{PSNR} = 22.39$ dB; (d) compressed by FPIC, using edges identical to those in (c) and assuming the same edge bit rate, 0.052 bpp, for those edges. Overall bit rate is 0.10 bpp, $\mathrm{PSNR} = 22.39$ dB.

by adjusting the image texture to match the lossily encoded edges prior to performing the wavelet transform. Second, the transform filters are not boundary filters, as in Mertins' approach, but pass across the edges, incorporating texture information from both sides of an edge in the resulting transform coefficients—while ignoring the edge itself. Third, single coefficients are not placed into the transform's high bands, as they are with Mertins' method. Single coefficients possess the statistical characteristics of low-band coefficients, so placing them in the high bands is inefficient. Further, by the nature of SPIHT's zerotree coding, the locations of significant coefficients are encoded together with their magnitudes. If singles are encoded this way, however, their locations are effectively being transmitted twice—both in the edge information and in the coefficient information. Instead, FPIC separates the single coefficients from the others, and then transmits only their magnitudes. The savings achieved by this technique is dependent on the detail in the input image: an image with only a few important edges will also have few singles, while an image with many edges may have hundreds of singles, making the separate coding of these singles worthwhile.

Note that each component of FPIC could be swapped in a modular way with that employed by Mertins: i.e., lossy segment-based vs. lossless pixel-based edge coding, the two edge-based wavelet transforms, and the differing methods of encoding singles.

An objective comparison of FPIC with Mertins' entire algorithm, as well as with its individual components, is difficult, since PSNR is not a good measure of the effectiveness of edge-based coders. In order to provide a reasonable visual comparison while excluding the effect of the differing edge-coding approaches, a test was performed with FPIC using edges identical to those reported in [19], and assuming the same edge cost of 0.052 bpp.

Fig. 12(a) shows the input "cameraman" image. The image compressed by SPIHT is shown in Fig. 12(b), Mertins' approach in Fig. 12(c) and FPIC in Fig. 12(d), all at 0.10 bpp. Both edge-based methods provide significantly improved sharpness over the SPIHT image, though at lower PSNR. The FPIC image provides more detail than the Mertins image. Since the edge coding is fixed to be the same, the improvement can be attributed to a combination of the different filtering method employed by FPIC in its wavelet transform, the use of SPIHT instead of EPWIC [23] for texture coding, and the separate coding of edge-associated detail (singles).

V. Conclusions

We have presented two progressive image coders intended for use with applications such as fast browsing, or low-bandwidth Internet access, in which users benefit from being able to understand the images early in the progressive bit stream. In fast browsing, for example, if a user is receiving an image, but determines it is not the one she wants, she can terminate the transmission and move on to the next image, saving time. In wireless applications, the transmitter may have knowledge of the available bandwidth, and decide to transmit only the early, low-quality portion of the image. In this case the user may not receive the higher quality image at all, and therefore relies on being able to understand the lower quality version. Both EEIC and FPIC yield images in which, at low bit rates, important edges are clearer than with traditional wavelet-based coders. The improvement is most pronounced for images containing simple, sharply defined objects, large text characters or graphics. For images of these types, the proposed coders may allow improved understanding of image content at low bit rates.

Acknowledgment

The authors thank Dr. A. Mertins for providing images, edge files and helpful information about his coding methods, and Dr. B. Rao for many helpful discussions. We thank the anonymous reviewers for their helpful suggestions.

References

[1] T. Frajka, P. G. Sherwood, and K. Zeger, "Progressive image coding with spatially variable resolution," in *Proc. ICIP-97*, vol. 1, Oct. 1997, pp. 53–56.

[2] G. Sperling, M. Landy, Y. Cohen, and M. Pavel, "Intelligible encoding of ASL image sequences at extremely low information rates," *Comput. Vis., Graph., Image Process.*, vol. 31, pp. 335–391, 1985.

[3] P. Schyns and A. Oliva, "From blobs to boundary edges: Evidence for time- and spatial-scale-dependent scene recognition," *Psychol. Sci.*, vol. 5, no. 4, pp. 195–200, July 1994.

[4] A. Oliva and P. Schyns, "Coarse blobs or fine edges? Evidence that information diagnosticity changes the perception of complex visual stimuli," *Cogn. Psychol.*, vol. 34, no. 1, pp. 72–107, Oct. 1997.

[5] A. Said and W. A. Pearlman, "A new, fast, and efficient image codec based on set partitioning in hierarchical trees," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, pp. 243–250, June 1996.

[6] G. Fan and W.-K. Cham, "Model-based edge reconstruction for low bit-rate wavelet-compressed images," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 10, no. 1, pp. 120–132, Feb. 2000.

[7] D. Schilling and P. Cosman, "Edge-enhanced image coding for low bit rates," in *Proc. Int. Conf. Image Processing (ICIP-99)*, vol. 3, Kobe, Japan, 1999, pp. 747–751.

[8] T. Berger and D. Miller, "Method and an Apparatus for Electronically Compressing a Transaction With a Human Signature," U.S. patent 5 091 975, 1992.

[9] S. M. Smith and J. M. Brady, "Susan—A new approach to low level image processing," *Int. J. Comput. Vis.*, vol. 23, no. 1, pp. 45–78, Jan. 1997.

[10] P. Rosin and G. West, "Nonparametric segmentation of curves into various representations," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 17, pp. 1140–1153, Dec. 1995.

[11] A. M. Rohaly, A. J. Ahumada Jr., and A. B. Watson, "A comparison of image-quality models and metrics predicting object detection," in *SID '95 Dig.*, 1995, pp. 45–48.

[12] G. Giraudet, C. Roumes, and J. Plantier, "Relative influence of spatial frequencies in object recognition," in *Proc. 4th Int. Conf. Cognitive and Neural Systems*, Boston, MA, May 2000.

[13] A. D. Sloan, "Retrieving database contents by image recognition: New fractal power," *Adv. Imag.*, vol. 5, no. 9, pp. 26–30, 1994.

[14] *JBIG2 Committee Draft*, ISO/IEC JTC1/SC29/WG1 N1093., Nov. 1998.

[15] G. Borgefors, "Distance transformations in digital images," *Comput. Vis., Graph., Image Process.*, vol. 34, no. 3, pp. 344–371, June 1986.

[16] S.-W. Hong and P. Bao, "Hybrid image compression model based on subband coding and edge-preserving regularization," *Proc. Inst. Elect. Eng.*, vol. 147, no. 1, pp. 16–22, Feb. 2000.

[17] D. Schilling and P. Cosman, "Feature-preserving image coding for very low bit rates," in *Proc. 2001 IEEE Data Compression Conf. (DCC)*, J. A Storer and M. Cohn, Eds., Snowbird, UT, Mar. 2001, pp. 103–112.

[18] W. H. Press, *Numerical Recipes in C: The Art of Scientific Computing*, 2nd ed. New York: Cambridge Univ. Press, 1992.

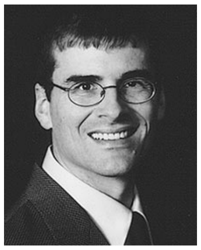
[19] A. Mertins, "Image compression via edge-based wavelet transform," *Opt. Eng.*, vol. 38, no. 6, pp. 991–1000, June 1999.

[20] S. Cen, H. Persson, D. Schilling, P. Cosman, and C. Berry, "Human observer responses to progressively compressed images," in *Proc. 31st Asilomar Conf. Signals, Systems, Computers*, vol. 1, Pacific Grove, CA, Nov. 1997, pp. 657–661.

[21] D. Schilling, P. Cosman, and C. Berry, "Image recognition in single-scale and multiscale decoders," in *Proc. 32nd Asilomar Conf. Signals, Systems, Computers*, Pacific Grove, CA, Nov. 1998, pp. 477–481.

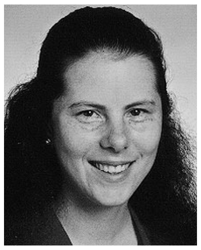
[22] J. Li and R. M. Gray, "Context-based multiscale classification of document images using wavelet coefficient distributions," *IEEE Trans. Image Processing*, vol. 9, pp. 1604–1616, Sept. 2000.

[23] R. W. Buccigrossi and E. P. Simoncelli, "Progressive wavelet image coding based on a conditional probability model," in *Proc. ICASSP*, vol. 4, Munich, Germany, Apr. 1997, pp. 2957–2960.



Dirck Schilling received the Sc.B. degree in mechanical engineering from Brown University, Providence, RI, in 1983, and the M.S. and Ph.D. in electrical engineering from the University of California, San Diego, in 1997 and 2001, respectively. His research interests include digital communications, signal processing, and image and video compression.

He is currently with ViaSat, Inc., Carlsbad, CA.



Pamela Cosman (S'88–M'93–SM'00) received the B.S. degree with honors in electrical engineering from the California Institute of Technology, Pasadena, in 1987, and the M.S. and Ph.D. degrees in electrical engineering from Stanford University, Stanford, CA, in 1989 and 1993, respectively.

She was an NSF Postdoctoral Fellow at Stanford University and a Visiting Professor at the University of Minnesota, Minneapolis, from 1993 to 1995. Since July of 1995, she has been on the faculty of the Department of Electrical and Computer Engineering at the University of California, San Diego, where she is currently an Associate Professor. Her research interests are in the areas of data compression and image processing.

Dr. Cosman is the recipient of the ECE Departmental Graduate Teaching Award (1996), a Career Award from the National Science Foundation (1996–1999), and a Powell Faculty Fellowship (1997–1998). She was an Associate Editor of the IEEE COMMUNICATIONS LETTERS (1998–2001), and was a Guest Editor of the June 2000 special issue on "error-resilient image and video coding" of the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS. She is currently Associate Editor of the IEEE SIGNAL PROCESSING LETTERS and a Senior Editor of the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATION. She was the Technical Program Chair of the 1998 Information Theory Workshop in San Diego and is a member of Tau Beta Pi and Sigma Xi.