Feature-Preserving Image Coding for Very Low Bit Rates

Dirck Schilling and Pamela Cosman

Department of Electrical and Computer Engineering
University of California, San Diego, La Jolla CA, 92093-0407
{dschilli, pcosman}@code.ucsd.edu (858) 822-1250

Abstract: Many progressive wavelet-based image coders are designed for good performance on natural images. They attempt to achieve the greatest reduction in mean squared error (MSE) with each bit sent, an approach that is most effective when the image is composed chiefly of low-frequency content. Many images, however, include sharp-edged objects, text characters or graphics that are not well handled by standard wavelet-based methods. These features, which may contain information important for recognition, become distorted and obscured when highly compressed by standard wavelet-based methods. In this paper, we present a new progressive image coder that treats an image as being composed of three types of information: edges, texture, and edge-associated detail. The locations of important edges are encoded using line graphic techniques. Texture is encoded using a wavelet-based zerotree approach. Detail near edges – that cannot be efficiently encoded as texture – is encoded separately with a bitplane coding technique. With this approach, features in the image that may be important for recognition are well preserved, even at low bit rates.

1 Introduction

Many images contain features that do not lend themselves well to wavelet-based coding methods. Such features may include sharp-edged objects, text characters and graphics. During a wavelet transform, the high-frequency content at the edges of these features tends to result in significant energy being concentrated in the transform's higher bands. Yet many progressive wavelet-based approaches send information about low-frequency coefficients first, e.g., [1,2]. When the image is transmitted at low bit rates, the high-frequency coefficients are coarsely quantized or even zeroed out, causing noticeable distortion in the reconstructed image. In applications such as fast browsing, where early recognition is important, or in low-bandwidth situations such as wireless Internet access, distortion of sharp-edged features may slow recognition of the image content by the user.

It may be advantageous to code these sharp-edged features using other methods, while retaining a wavelet-coding approach for the remaining regions. Several techniques have been proposed for compressing mixed-mode images – images containing more than one type of image data. Some methods segment the image into rectangular regions, and then treat each region effectively as a separate image, for which separate coders are employed. These methods do not address situations where the different regions are oddly shaped or overlap each other. Other approaches [3,4] segment the image into foreground and background regions. The segmentation is encoded as a binary bitmap, and wavelet techniques are employed to separately encode the foreground and background images.

With this approach, every foreground region is surrounded by a sharply defined closed contour, since it is defined as a connected component of pixels. Objects that are sharply defined on one side, but less clearly on the other, are not well represented by this approach, since it creates a sharp edge around the entire object. A further class of methods [5,6] addresses both the issue of non-rectangular, overlapping image regions, and that of image features not surrounded by a closed sharp contour. These methods transmit the locations of sharp-edged features separately, and combine this information with texture information to yield a reconstructed image with well-preserved sharp features, even at low bit rates.

In this paper, we present a new progressive image coder that 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. This paper is organized as follows. In Section 2 we provide a general overview of the proposed algorithm and its components. In Section 3 we describe the feature-preserving wavelet transform employed, and discuss the handling of lossy edges and edge-associated detail. We present some compression results and conclusions in Section 4.

2 Overview of the Proposed Image Coder

Our primary goal in the design of a feature-preserving image codec (FPIC) is to identify features in an image that may be important for human recognition and understanding, and to preserve the clarity of these features at low bit rates, at the expense of somewhat reduced fidelity in other regions of the image. *Edges* often represent information that is valuable for understanding, for example in the case of text characters or graphics. After "removing" the edges, the remaining information can be described as background *texture*. An edge itself actually consist of two parts: the location of the edge, and the image intensities on each side of it. The method we describe in Section 3 for removing the edges, a feature-preserving wavelet transform, allows the intensity information for each side of an edge to be stored in most cases in the texture component. In some situations, though, it is inefficient to retain edge intensity information in the texture component. Such situations occur where edges include sharp corners, are close together, or surround small regions, for example small text characters. In these cases, it can be more efficient to encode the intensity information separately, hence the third information component, *edge-associated detail.*

A block diagram of FPIC is shown in Figure 1. The input image is first passed through an edge detection step [7] to extract edges which are deemed to be important for recognition. The resulting edge pixels are approximated with connected line segments [8]. While our implementation approximates curved edges by sets of linked straight line segments, higher-order representations such as arcs, ellipses and polynomials could also be used. 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 procedure therefore includes a step to remove short edges.

Following edge extraction, the edge locations are encoded with a multiring chain coder [9], and are transmitted to the decoder. Our implementation is similar to that

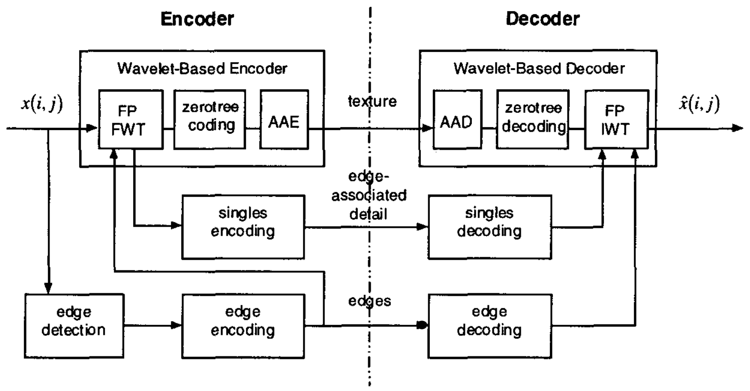

Figure 1: Block diagram of the feature-preserving image coder (FPIC).

employed for EEIC in [5]. This method allows edges to be encoded nearly losslessly, in that no encoded endpoint is further than a given tolerance ℓ from the input curve. If ℓ is one pixel, for example, the encoded edges are virtually indistinguishable from the input edges. The cost of encoding the edges can be adjusted by appropriately selecting this parameter: as ℓ is increased, greater error is allowed, and the bit cost is reduced.

When the edges have been encoded, 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. The standard wavelet transform employed by SPIHT is replaced here by a feature-preserving transform that removes the edges as it transforms the image. The transform outputs the coefficients, representing the image texture, and a set of isolated or "single" coefficients, which represent the edge-associated detail. Finally, the quantized, entropy-encoded texture coefficients are transmitted, and the single coefficients are separately encoded and transmitted.

3 Feature-Preserving Wavelet Transform

In this section we describe the wavelet transform employed in FPIC. Like most image transforms, it compacts the energy of the image into a relatively few low-frequency coefficients. As it does this, it also uses the encoded edge locations to "remove" the edges from the image's texture information. In so doing, it 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. Finally, it extracts the edge-associated detail information, contained in isolated "single" coefficients near certain edges, and passes this out for separate encoding.

3.1 Forward Transform

A standard one-level, one-dimensional forward wavelet transform is illustrated in Figure 2. 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.

Figure 3 illustrates the operation of the forward feature-preserving wavelet transform. In this case, both the forward and inverse transforms will have knowledge of the location of the ideal step, or edge. In this paper we define edges as *crack edges*, so that the location of an edge falls between two image pixels. 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.

3.2 Inverse Transform

The step removal procedure described above is a linear one, and so can be described fully by a system of equations in 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 $N \times N$ step-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

$$\begin{bmatrix} f^T & & & & \\ & g^T & & & \\ & & f^T & & \\ & & & \ddots & \\ & & & & g^T \end{bmatrix} \begin{bmatrix} x_o \\ x_1 \\ x_2 \\ \vdots \\ x_{N-1} \end{bmatrix} = \begin{bmatrix} c_o \\ c_1 \\ c_2 \\ \vdots \\ c_{N-1} \end{bmatrix}$$

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 $f = [f_0 \quad f_1 \quad f_2 \quad f_3 \quad f_4]^T$ and $g = [g_0 \quad g_1 \quad g_2]^T$. In the step-removal procedure applied during the forward transform, the transformed coefficients are computed as follows:

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

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:

$$F \bullet x + \left[\begin{array}{ccc:ccc}
0 & \cdots & 0 & 0 & \cdots & 0 \\
 & & \vdots & \vdots & & \\
 & & f_4 & -f_4 & & \\
 & & g_2 & -g_2 & & \\
\hdashline
 & & -(f_0+f_1) & f_0+f_1 & & \\
 & & 0 & 0 & & \\
 & & \vdots & \vdots & & \\
0 & \cdots & 0 & 0 & \cdots & 0
\end{array}\right]
\left[\begin{array}{c}
x_0 \\ \vdots \\ x_{s-1} \\ \hdashline x_s \\ \vdots \\ x_{N-1}
\end{array}\right] = c$$

Denoting the edge removal matrix as E, the transform becomes $(F+E)x = c$. The one-level inverse transform is therefore given by $\hat{x} = (F+E)^{-1}c$. The effect of additional edges is additive, so that for L edges, $(F + E_0 + E_1 + \ldots + E_{L-1})x = c$, and $\hat{x} = (F + E_0 + E_1 + \ldots + E_{L-1})^{-1}c$. The combined step removal matrix is $E = E_0 + E_1 + \ldots + E_{L-1}$, 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 [10], 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.

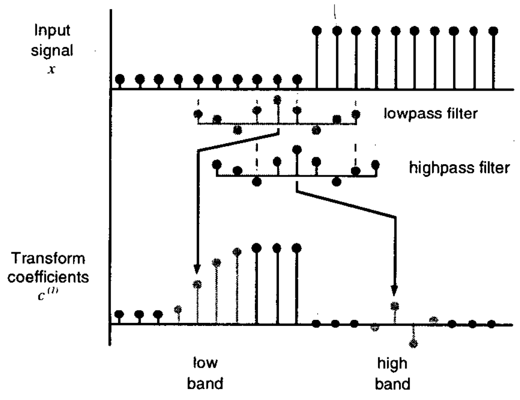

Figure 2: Schematic of a standard one-level wavelet transform on a one-dimensional input signal containing an ideal step function.

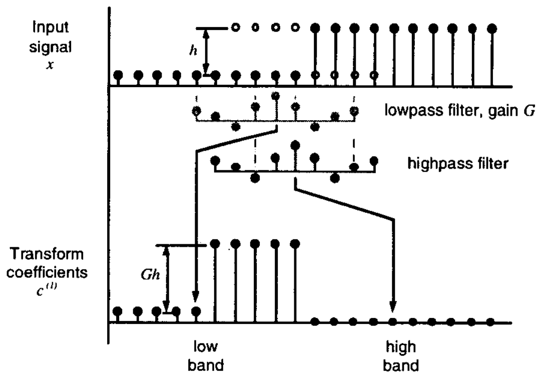

Figure 3: Schematic of a one-level feature-preserving wavelet transform on a one-dimensional input signal containing an ideal step function.

3.3 Edge Preprocessing

It is clear from the preceding discussion that the wavelet transform is 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. When edges are encoded lossily, though, their reported locations differ frequently from the actual edge positions. 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, starting at a radius R (typically 2-4 pixels) from the edge and progressing successively nearer. It has the effect of moving the image data slightly, to match the lossy edges as encoded. In this way the benefit of the step-removal process for texture coding can be achieved, with little loss in overall image fidelity.

3.4 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 [6]. 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 placed into the high band. Its statistics do not match those of the other high-band coefficients, though. These isolated, "single" coefficients contain the information about image color near closely spaced edges, which we call edge-associated detail.

Since the statistical characteristics of singles differ from the other coefficients in the high bands, it is desirable to employ a separate coding method for them. 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.

4 Results and Conclusions

FPIC is similar to an edge-based compression algorithm presented by Mertins in [6], however there are several significant differences. First, FPIC allows edges to be coded lossily, by adjusting the image to 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 [6], and assuming the same edge cost of 0.052 bpp. 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. The low overall bit rate for the following comparisons reflects this focus.

Figure 4a shows the input "cameraman" image. The image compressed by SPIHT is shown in Figure 4b, Mertins' approach in Figure 4c and FPIC in Figure 4d, all at 0.10 bpp. Both edge-based methods provide significantly improved sharpness over the SPIHT image, though at lower PSNR. In the FPIC image, substantial improvement over the Mertins image can be seen in the details. 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 [11] for texture coding, and the separate coding of edge-associated detail (singles).

Figure 5 shows an example of the full FPIC algorithm's output, including lossy edge coding. In each case the image was compressed to 0.10 bpp. The central object is significantly sharper in the FPIC images (c and d) than in the SPIHT image (b). In (c), the edges were encoded nearly losslessly with $\ell = 1$ pixel, consuming 20.6% of the bit budget. In (d), ℓ was set to 4 pixels, so that the edges consumed 12.6% of the bit budget.

In conclusion, we have presented a progressive feature-preserving image coder that treats an image as being composed of three types of information: edges, texture, and edge-associated detail. Each information component is encoded using a method tailored to its specific characteristics. FPIC allows the user a degree of flexibility in deciding the relative importance of the three information components. For images containing features such as sharp-edged objects, text characters or graphics, FPIC can substantially reduce ringing and distortion near these features. In applications such as fast browsing, where early recognition is important, or for wireless access to the Internet, where bandwidth is strictly limited, the technique may prove useful in allowing recognition of image content at very low bit rates.

Acknowledgements: The authors wish to thank Dr. Alfred Mertins for providing images, edge files and helpful information about his coding methods, and Dr. Bhaskar Rao for many helpful discussions.

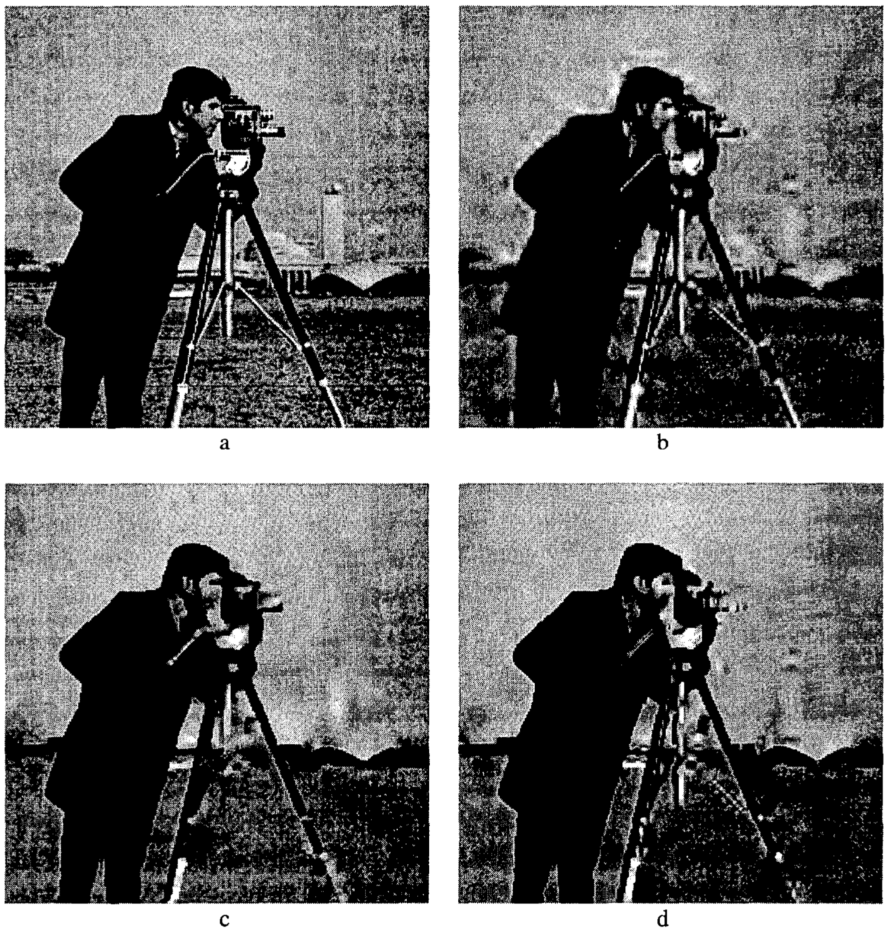

Figure 4: a) The original 256×256 image; b) compressed by SPIHT to 0.10 bpp, PSNR = 24.37; c) as reported in [6], 0.10 bpp, PSNR = 22.39 dB; d) compressed by FPIC, using edges identical to those in (c) and assuming the same edge bit rate, 0.052 bpp, for those edges. Overall bit rate is 0.10 bpp, PSNR = 22.39 dB.

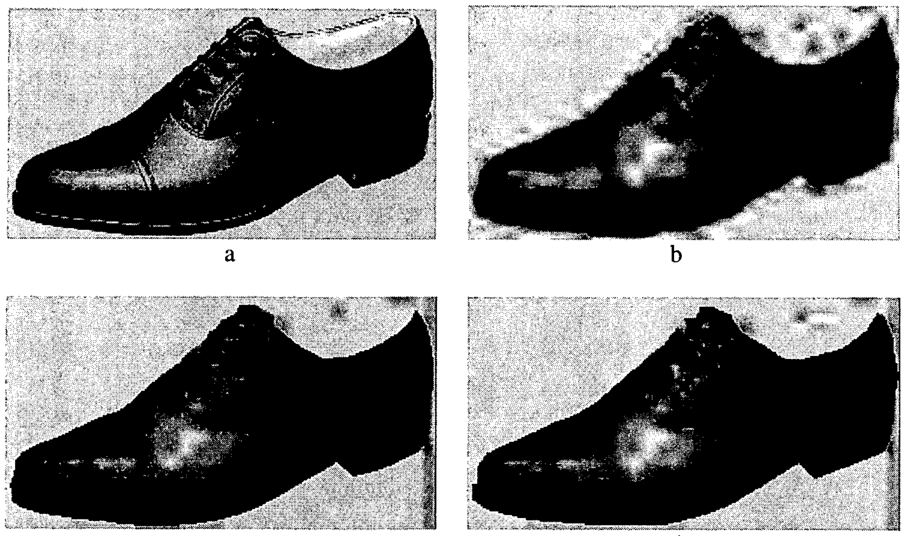

Figure 5: A shoe. a) Original 250 × 135 image, b) SPIHT at 0.10 bpp, PSNR = 21.62 dB; c) FPIC at 0.10 bpp, of which edges consumed 20.6%, PSNR = 20.34 dB; d) FPIC at 0.10 bpp, of which edges consumed 12.6%, PSNR = 19.43 dB.

References

[1] J. Shapiro, Embedded Image Coding Using Zerotrees of Wavelet Coefficients, *IEEE Trans. on Signal Processing*, 41(12):3445–62, 1993.

[2] A. Said, W. Pearlman, A New, Fast, and Efficient Image Codec Based on Set Partitioning in Hierarchical Trees, *IEEE Trans. Circ. and Sys. for Video Tech.*, 6(3):243–50, 1996.

[3] P. Haffner, L. Bottou, P. Howard, P. Simard, Y. Bengio, Y. LeCun, High Quality Document Image Compression with DjVu. *Electronic Imaging*, 7(3):410–25, July 1998.

[4] D. Huttenlocher, P. Felzenszwalb, W. Rucklidge, Digipaper: a Versatile Color Document Image Representation, *1999 IEEE Intl. Conf. on Image Proc.*, vol. 1, p. 219–23, 1999.

[5] D. Schilling, P. Cosman, Edge-Enhanced Image Coding for Low Bit Rates. *1999 IEEE Intl. Conf. on Image Proc.*, vol. 3, p.747-51, 1999.

[6] A. Mertins, Image Compression via Edge-Based Wavelet Transform. *Optical Engineering*, 38(6):991-1000. SPIE, June 1999.

[7] S. M. Smith, J. M. Brady, SUSAN – A New Approach to Low Level Image Processing, *Intl. Journal of Computer Vision*, 23(1):45–78, 1997.

[8] P. Rosin, G. West, Nonparametric Segmentation of Curves into Various Representations, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 17(12):1140–53, 1995.

[9] T. Berger, D. Miller, Method and an Apparatus for Electronically Compressing a Transaction with a Human Signature, U.S. patent 5091975, 1992.

[10] William H. Press, *Numerical Recipes in C: the Art of Scientific Computing*, 2nd Edition, Cambridge University Press, New York, 1992.

[11] R. W. Buccigrossi, E. P. Simoncelli, Progressive Wavelet Image Coding Based on a Conditional Probability Model, *1997 IEEE ICASSP*, vol. 4, p. 2957–60, 1997.