Memory Constrained Wavelet-Based Image Coding

Pamela Cosman and Kenneth Zeger
Dept. Electrical and Comp. Engr., University of California at San Diego,
La Jolla, CA 92093-0407. E-mail: {cosman,zeger}@code.ucsd.edu

Abstract

Transform coding methods such as JPEG which operate on small blocks tend to have poorer performance than full-frame transform methods based on wavelet decompositions, but the memory requirements are much lower. We present a method for ordering the wavelet coefficient information in a compressed bit stream that allows the image to be sequentially decoded, with lower memory requirements than conventional wavelet decompression schemes. In addition, we introduce a hybrid filtering scheme that uses different horizontal and vertical filters, each with different depths of wavelet decomposition. This reduces the decoder memory requirements by reducing the instantaneous number of wavelet coefficients needed to perform inverse filtering.

1 Introduction

We consider applications in which an image or video is to be transmitted in compressed format to an inexpensive output device, such as a low-end color printer or a wireless hand-held videophone. For such devices, the amount of on-board memory is a significant cost factor and affects the competitive positioning of the product. An image or video compression algorithm which provides excellent image reproduction quality with low memory usage would be attractive in this situation. Some existing compression algorithms already have low memory requirements, but with inferior reproduction quality. For example, in baseline sequential JPEG, the image is processed in blocks of 8×8 pixels. The blocks are processed in raster scan order, so a decoder needs to buffer only 8 contiguous rows of an image at any one time. After reconstructing a set of 8 lines, a decoder can flush them out of memory and work on the next strip of 8 lines. However, many wavelet-based algorithms have much better distortion vs. rate performance than does JPEG. One of the most effective known image compression algorithms is wavelet zerotree coding, introduced by Shapiro [1] and later refined by Said and Pearlman [2]. It is a progressive transmission technique in which a low-resolution description of the image can be reconstructed at the decoder using only a small fraction of the bit stream, and the quality of the image continually improves as more of the compressed bit stream arrives at the decoder. Shapiro's technique, known as Embedded Zerotree Wavelet (EZW) coding, and the refinement due to Said and Pearlman, known as Set Partitioning in Hierarchical Trees (SPIHT), are among the many wavelet algorithms which provide excellent compression performance, but which require a complete image (at each transmission rate) to be reconstructed before any particular part of the image can be printed or displayed. The same difficulty also occurs with algorithms based on full-frame Discrete Cosine Transforms.

In this paper, we introduce two new techniques which work together to reduce memory requirements. First, we alter the order of the transmitted information, so that the encoder only sends to the decoder the minimal set of wavelet coefficients needed to compute a given segment of an inverse wavelet transform and to produce one output line of the image. The number of coefficients involved in the given partial inverse transform operation can be minimized by judicious selection of different filter lengths in the two spatial directions, and at different levels of hierarchical decomposition. We describe our memory-efficient coding scheme in terms of an improvement to embedded zerotree algorithms such as EZW and SPIHT. We use the SPIHT zerotree approach purely as a quantization method - we use a different wavelet transform and a different bit stream ordering, and we omit their entropy coding step to reduce complexity.

2 Bit-stream re-ordering

One approach to reducing memory requirements is to partition the image into horizontal strips, each containing a small number of horizontal rows, and then to compress each strip independently using an EZW-type algorithm on the rectangular sub-images. In practice, this technique yields much poorer compression performance, due primarily to the fact that EZW and SPIHT advantageously exploit the existence of multiple levels of wavelet decomposition in the hierarchy, and thus save many bits when large zerotrees of coefficients occur. In contrast, strip-based EZW or SPIHT algorithms used on narrow strips (e.g., 32 rows) have only a limited number of decomposition levels available. Additionally, partitioning the image into strips tends to cause undesirable visible "blocking" artifacts at the boundaries of the strips. With a strip-based decomposition, the overall PSNR performance can be improved by optimally allocating bits among the strips. We tested such an optimal allocation, using standard Lagrangian methods and found that the image quality is still poor and the blocking artifacts between the strips remain visually displeasing.

The boundaries of the strips can be made less visually noticeable by using overlapping strips, at the expense of increased rate. Figure 1 shows the distortion vs. rate performance on the 512×512 color Lena image when fairly large strips (64 rows) are compressed, and bits are allocated optimally among them. The performance for the original SPIHT coder is shown for comparison. The two lower curves display the results for the cases where the strips are not overlapped, and where they are overlapped by 4 rows on top and bottom. As can be seen, compared to the original SPIHT coder, the performance loss due to the more shallow zerotrees is substantial [3].

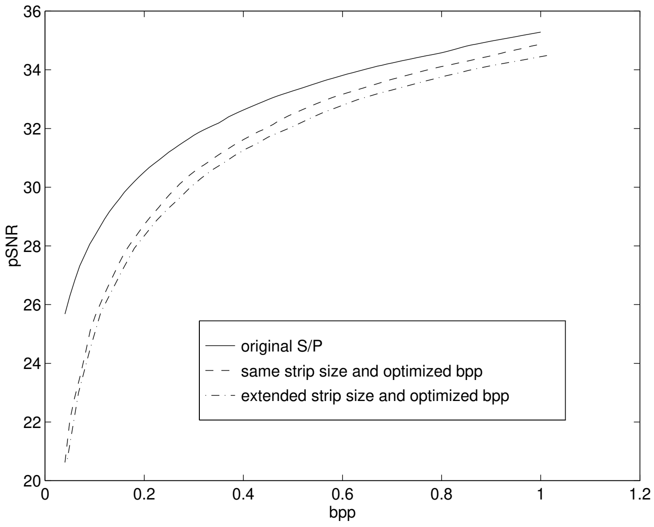

Figure 1: Comparison of the distortion-rate performance for the full-image SPIHT algorithm, the strip-based SPIHT algorithm on strips of 64 rows, and strip-based SPIHT on overlapping strips.

A better approach is to maintain the full-frame wavelet transform, but to rearrange the order of the transmitted bit stream from EZW or SPIHT, so that a reduced amount of memory is needed at any given time. Even though a full-frame forward wavelet transform is used to obtain the wavelet coefficients array, only a small subset of the wavelet coefficients is required in the inverse transform to obtain any given output row, without further quantization loss. This process can be described in terms of line-by-line reconstruction in the spatial domain of the image. We seek the *minimum set* of wavelet coefficients that must be received by the decoder in order to reconstruct a given single horizontal row in the output image. After the decoder has received that minimal set, and used it to reconstruct one particular row in the image, we then seek the *minimum set of additional wavelet coefficients* that must be transmitted by the encoder so that the following row can be reconstructed. We also determine how much of the currently stored set of coefficients can be purged from memory. Coefficients that can be expunged are those which are not needed in the reconstruction of any future rows. This process is iterated for each row until the entire image is reconstructed.

As an example, suppose the subband decomposition operation consists of only a single level of decomposition using a 2-tap Haar wavelet filter. The coefficients required in order to reconstruct the top row of the output image are shown in Figure 2. If more levels of decomposition are used, then the buffering requirements at the decoder are increased. As shown in Figure 3, if a two-level decomposition is performed, the decoder's inverse transform operation can be analyzed in terms of one level of inverse wavelet transform (IWT) at a time. A single row of coefficients is required from each of the 4 smaller subbands; these allow reconstruction of the top two rows of the LL_1 band. Using only the top row from the LL_1 band, together with the top row from each of the other subbands, allows a second round of inverse wavelet filtering to reconstruct the top row (and second row) of image pixels to be reconstructed.

Now consider the zerotree dependencies. With zerotree structures defined as in EZW, each coefficient in the lowest band (LL_2 band) has three children, one in each of the three directional bands at that level of decomposition. And each of those three children has a 2×2 block of children in the next lower directional subband. This additional quantization information is not needed in the inverse transform to reconstruct the top row of pixels. However, with an EZW-style quantiza-

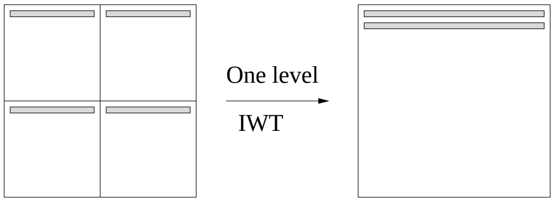

Figure 2: The decoder performs an inverse wavelet transform (IWT) operation to reconstruct the top row (and the second row) of pixels, using only a single line of coefficients from each of the four bands.

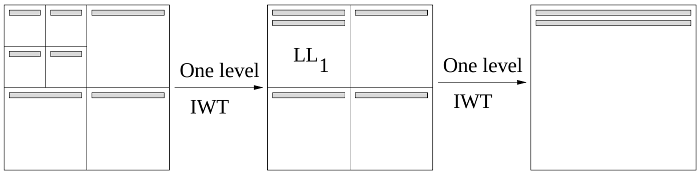

Figure 3: The decoder performs one level of IWT to reconstruct the top two rows of the LL_1 subband, using only a single line of coefficients from each of the four smallest subbands. Another round of IWT allows reconstruction of two rows of pixels.

tion, one cannot avoid the transmission and storage of this additional information when the top row is reconstructed. For this quantization method, the minimal set of quantized coefficients received by the decoder (for this decomposition level and choice of filter) is shown in Figure 4. This minimal set of coefficients allows reconstruction of the top line of the image and is in fact sufficient to enable the decoder to reconstruct the top four lines of the image.

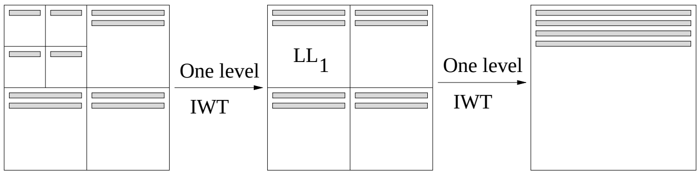

Figure 4: Because of the zerotree dependencies, the decoder receives extra coefficients which are not required in the inverse transform operation that yields the top row. The extra coefficients allow four rows to be reconstructed.

If the image is of size 512×512, the top four lines contain $512 \times 4 = 2048$ pixels. With the Haar filter, 2048 coefficients are needed to reconstruct those 2048 pixels. With a four-level decomposition, the decoder needs 1 line from each of the four smallest bands, and 2, 4, and 8 lines at the subsequent levels of the hierarchy, and then can reconstruct 16 rows. Again the number of wavelet coefficients received equals the number of pixels reconstructed. The entire group can then be purged from the decoder's memory, followed by a similar-sized group being received and decoded in order to reconstruct the next set of 16 rows.

The situation is less favorable when longer filters are used. For the 4-tap Daubechies filter, Figures 5 and 6 are analogous to Figures 2 and 4. These figures show the minimal sets of coefficients needed by the decoder to reconstruct a single row in the image, using one and two levels of decomposition and zerotree quantization. In Figure 5, two rows of coefficients are required from each of the 4 subbands (a total of $4 \times 512 = 2048$ coefficients). The inverse wavelet transform operation on those coefficients alone allows reconstruction of the top row of the output image (512 pixels). To reconstruct the second row, no additional coefficients are needed, nor can we purge any of the coefficients from the first row. To reconstruct the third row, one line can be purged from each of the 4 subbands, and we need to receive a new line from the encoder from each of the 4 subbands. The memory usage thus remains constant. This forms a "sliding window" in which wavelet coefficients enter the window, are used for a few rounds of inverse filtering to reconstruct some rows, and then exit the window. This sliding window for wavelet coefficients is the main tool that we use in line-by-line wavelet coding. With a two-level decomposition, as depicted in Figure 6, we need 2 rows from each of the 4 small subbands, and therefore, because of the zerotree structure, 4 rows from each of the outer subbands. With these coefficients, inverse transforming yields the first row of the image (actually, we would obtain the first two rows, but we are only concerned at this point with what is required to reconstruct one row, and the requirements are the same.)

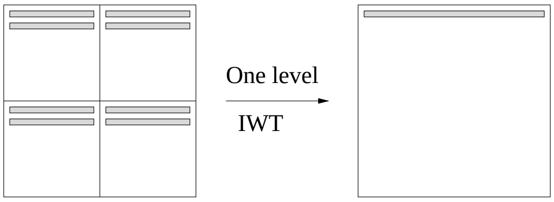

Figure 5: Wavelet coefficients required to reconstruct top row, with a 4-tap filter.

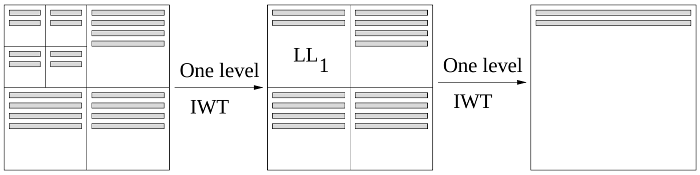

Figure 6: Wavelet coefficients from a 2-level decomposition required to reconstruct top row, with a 4-tap filter, when the coefficients are transmitted in zerotree structures.

2.1 Implementation issues in the bit stream ordering

This process can be viewed as a re-ordering of the SPIHT or EZW output bit stream. An ordinary SPIHT encoder (without entropy coding) is used initially to encode the input image to the desired transmission rate. The compressed bit stream is stored at the encoder. Each bit of the stored image is associated with one of the trees of coefficients. The encoder re-arranges the progressively ordered bits so that the bits corresponding to the top line of trees are transmitted first, followed by the next line of trees, etc. Entropy coding can be applied to this re-ordered bit stream.

A very small amount of header information is required. The bit streams corresponding to the succession of "minimal sets" of coefficients for the successive rows arrive at the decoder in one continuous stream. The decoder needs to know the beginning and end of each set. This can be achieved by transmitting, at the start of each set, a single number which informs the decoder how many bits will be transmitted for that set (row). The overhead is small, but it turns out that it is not needed, as there is a more compact way for the decoder to learn this information. This is accomplished as follows. The encoder transmits one single header at the beginning of the entire image, which describes the threshold T_n and the coordinates (x,y) of the coefficient at which the encoding of the original algorithm (not re-ordered) terminates. The decoder then evaluates, as it decodes each row in the re-ordered bit stream, whether the received bits correspond to a threshold of T_n, and, if they do, whether the coefficient coordinates correspond to a position that exceeds (x,y). If they do, then the decoder deduces that the data corresponds to the next row.

3 Hybrid Filtering

The examples given above illustrate that more wavelet coefficients are required to produce a single output row when either the filter length increases, or when the number of decomposition levels increases. For example, with a 512×512 image undergoing a single level of filtering with a 2-tap filter, only $2 \times 512 = 1024$ wavelet coefficients are required to produce an output row (512 pixels). This constitutes 0.4% of the wavelet coefficient array. With a full 6 levels of decomposition using the same short filter, 64 rows or 12.5% of the array is involved in producing a single output row. The decrease in SPIHT performance is very significant when Haar filters are substituted for the 9-7 biorthogonal filters. With the filtering and zerotree structure used in the SPIHT encoder, 100% of the coefficient array is involved in producing a single output row. For the color Lena image, Figure 7 shows the performance reduction from decreasing the number of decomposition levels while using the 9-7 biorthogonal filters. Figure 8 similarly shows the performance reduction from substituting a Haar filter for the 9-7 biorthogonal filters, while maintaining the number of decomposition levels at six.

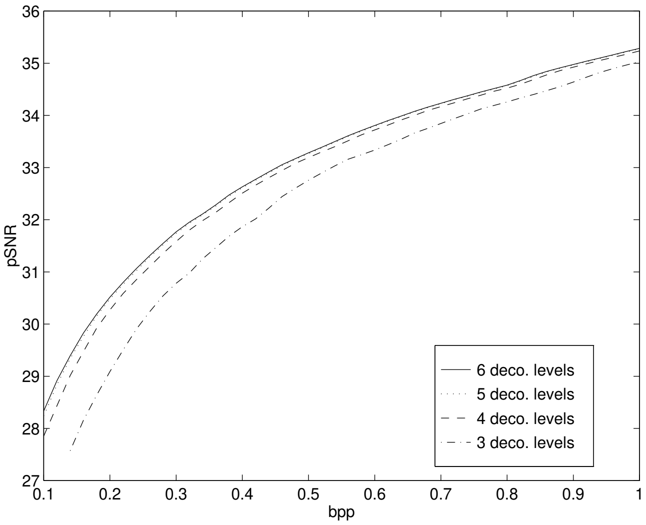

Figure 7: Comparison of the distortion-rate performance for SPIHT with different numbers of decomposition levels, ranging from 3 to 6.

The goal of this work has been to provide good distortion vs. rate performance with limited decoder buffering. The performance of EZW and SPIHT with Haar filters and 3 or 4 levels of decomposition is very poor however. The traditional wavelet/subband coding approach applies the same filtering in the horizontal and vertical directions. For example, the SPIHT algorithm uses 6 levels of decomposition with 9-7 biorthogonal filters in both the horizontal and vertical directions.

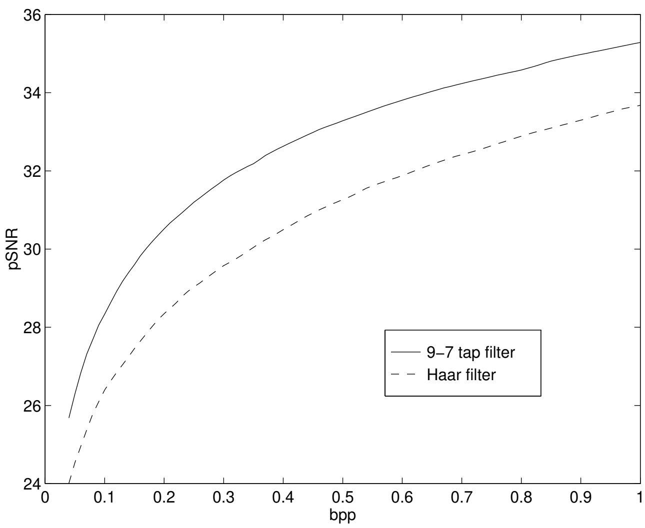

Figure 8: Comparison of the distortion-rate performance for SPIHT (with six levels of decomposition), with the 9-7 biorthogonal filters vs. the Haar filters.

In our scheme, large numbers of levels of decomposition *in the vertical direction* and long filter lengths *in the vertical direction*, cause increased memory requirements. The filtering in the horizontal direction has no effect on the memory requirements, since one horizontal row always is reconstructed at one time. For printing applications, this is useful because the order of reconstructing lines will match the direction in which the paper exits the printer. To satisfy (and exploit) these constraints, we introduce a hybrid filter– a wavelet transform that uses either different numbers of decomposition levels or different filter lengths, or both, in the different spatial directions.

As an example, we could perform the full 6 levels of decomposition using the 9-7 filters in the horizontal direction. In the vertical direction we could decompose 3 levels with the 9-7 filters, and an additional 3 levels with the 2-tap Haar filters. The total number of levels of decomposition in each direction would be 6 in this example, but the buffering would be much less than if the 9-7 filters were used fully on both directions. The total number of levels of decomposition does not have to be the same in the two directions. For example, one could perform the horizontal filtering as before, whereas the vertical direction might consist of 2 levels of decomposition with the 9-7 filters, and 3 levels of filtering with the Haar filters of length 2, for a total of only 5. There exist many different possibilities, each presenting its own trade-off between distortion, rate, and buffering requirements. A further generalization is to allow the transforms in the two directions to be of completely different types. For example, the horizontal transform might be a wavelet transform, whereas the vertical transform might be a block DCT. This would present still other possible trade-offs between distortion-rate performance and buffering requirements, as well as issues concerning visual artifacts due to "blocking".

3.1 Example with block diagram

Figure 9 shows a block diagram of the decoding mechanism used with a hybrid filtering structure in the vertical direction. The figure shows 6 levels of wavelet reconstruction. The first 3 levels are performed using Haar filters and the latter 3 levels use 9-7 biorthogonal filters. It is assumed that all 6 levels of decomposition in the horizontal direction use the 9-7 biorthogonal filters (which are not shown). In addition to the filters, delay elements are shown (labelled "D") preceding the high pass filters (and the first low pass filter).

Once the filters' memories are filled, to produce 64 new outputs (i.e., vertical lines) at the end of the filter bank, one must provide 1 scalar input at $L1$ and $H1$, 2 inputs at $H2$, 4 at $H3$, 8 at $H4$, 16 at $H5$, and 32 at $H6$. For some quantization schemes, only the memory of these filter banks is necessary to completely decode the image from the wavelet domain. For other types of quantization, namely zero-tree encoding, we must make use of the delay elements shown in Figure 9.

In EZW coding, information from different bit planes must be temporarily stored until entire wavelet coefficients can be deduced. Once wavelet coefficients are known, they can (in the proper order) by sent into the filters and propagated through the filter bank to produce output image values. The "D" boxes represent the process of accumulating bit plane information and delaying the input until wavelet coefficients are fully known (one by one). For EZW coding, for example, the total memory requirements are the sum of the filter memories and the sum of the delay memories. A substantial savings in memory over the usual full-image EZW decoding can still be achieved.

In general, the memory requirements can be divided into a portion required by the inverse filtering operations (to keep the filters' memories filled), and an additional portion required by the quantization scheme (depicted by the boxes labeled D). Memoryless scalar quantization (SQ) requires no extra storage, whereas predictive SQ and a context-based adaptive entropy coded SQ might require some extra coefficients to be stored and used for prediction or context during reconstruction. Zerotree-based schemes also require extra storage beyond the amount required for the filters.

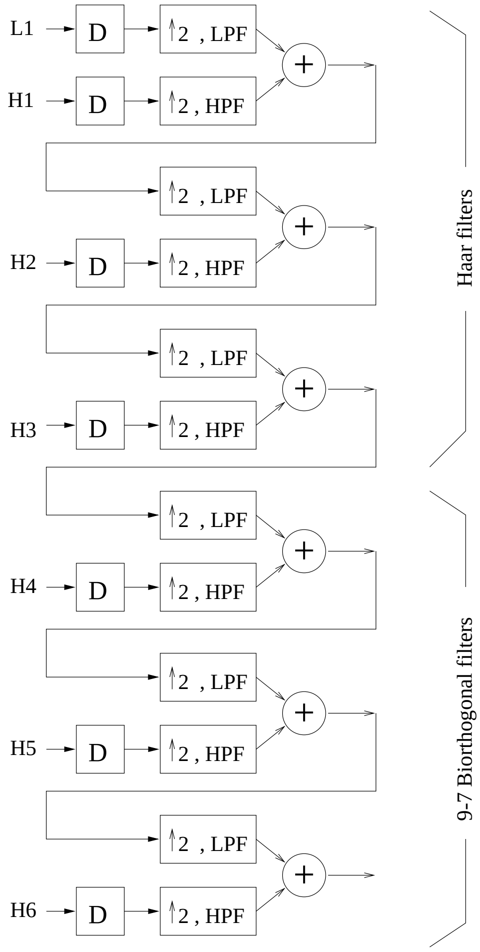

Figure 9: Block diagram of hybrid filter wavelet decoder for vertical direction.

4 Results and Conclusions

The 24-bit color Lena image was compressed by a factor of 100:1 down to 0.24 bpp by three different versions of the coder. We display only the luminance (Y) component of all images. Figure 10(a) shows the Y component of the original image, and Figure 10(b) shows the result of SPIHT compression. The PSNR is 30.77 dB. Here we report a color PSNR by first averaging the three component colors' MSEs to produce MSE_{avg} and then taking a logarithm as: $PSNR = 10\log_{10}(255^2/MSE_{avg})$. In Figure 10(c) and (d), the horizontal filtering is unchanged from that used in the original SPIHT algorithm. In (c), the vertical filtering uses 6 levels of decomposition with Haar filters, and the PSNR drops to 29.60 dB. In (d), the vertical filtering uses 3 levels of decomposition with the 9-7 filters, and an additional 3 levels with the Haar filters. The PSNR is 30.64 dB. A numerical comparison of the performance of various versions is given in Table 1.

Decomposition Levels with 9-7 filters	Decomposition Levels with Haar filters	PSNR (dB)
6	0	30.77
5	1	30.76
4	2	30.73
3	3	30.64
2	4	30.42
1	5	30.05
0	6	29.60

Table 1: PSNR results for color Lena image at 0.24 bpp, using the original SPIHT algorithm, and bit-stream reordering with various hybrid filters.

Acknowledgments

We thank Stefan Tjarnlund and Magnus Pettersson for their help in running numerical experiments and Tamas Frajka for producing the images for this paper. This work was supported in part by Hewlett-Packard Co. and by the National Science Foundation.

References

[1] J. M. Shapiro. Embedded image coding using zerotrees of wavelet coefficients. *IEEE Transactions on Signal Processing*, 41(12):3445–3462, December 1993.

[2] A. Said and W. A. Pearlman. A new, fast, and efficient image codec based on set partitioning in hierarchical trees. *IEEE Transactions on Circuits and Systems for Video Technology*, 6(3):243–250, June 1996.

[3] M. Pettersson and S. Tjärnlund. Color image compression for color printers using wavelet transform. Master's thesis, Linköping University, Dept. of Electrical Engineering, S-581 83 Linköping, Sweden, November 1997.


(a) (b) (c) (d)

Figure 10: a) original, b) SPIHT compression, c) all Haar vertically, d) 3x3 Haar