IEEE DATA COMPRESSION CONFERENCE – SNOWBIRD, UTAH, MARCH 2000

Fast Progressive Image Coding without Wavelets

Henrique S. Malvar
Microsoft Research
One Microsoft Way, Redmond, WA 98052
malvar@microsoft.com

Abstract

We introduce a new image compression algorithm that allows progressive image reconstruction – both in resolution and in fidelity, with a fully embedded bit-stream. The algorithm is based on bit-plane entropy coding of reordered transform coefficients, similar to the progressive wavelet codec (PWC) previously introduced. Unlike PWC, however, our new progressive transform coder (PTC) does not use wavelets; it performs the space-frequency decomposition step via a new lapped biorthogonal transform (LBT). PTC achieves a rate vs. distortion performance that is comparable (within 2%) to that of the state-of-the-art SPIHT (set partitioning in hierarchical trees) codec. However, thanks to the use of the LBT, the space-frequency decomposition step in PTC reduces the number of multiplications per pixel by a factor of 2.7, and the number of additions by about 15%, when compared to the fastest possible implementation of the "9/7" wavelet transform via lifting. Furthermore, since most of the computation in the LBT is in fact performed by a DCT, our PTC codec can make full use of fast software and hardware modules for 1-D and 2-D DCTs.

1. Introduction

Most modern image (picture) compression algorithms employ a wavelet transform (WT) followed by quantization and entropy encoding [1]–[3]. The WT is preferable over the discrete cosine transform (DCT) used in the popular JPEG (Joint Photographic Experts Group [4]) codec mainly because of three aspects. First, WTs do not have blocking artifacts like the DCT at low bit rates; second, WTs naturally allow for image reconstruction that is progressive in resolution; third, WTs lead to better energy compaction, and therefore better distortion/rate performance, than the DCT. The best performance of WT-based codecs can be attested by the fact that all JPEG-2000 candidate codecs were WT-based.

The better performance of the WT is not without a cost. There are two main disadvantages of WTs with respect to the DCT. First, lack of localized data access. A 2-D separable WT has to perform filtering across image columns, which can produce significant cache misses in a simple row-column filtering implementation. It is possible to compute the WT coefficients in a line-by-line approach, without explicit full-column filtering [5], but the coefficients will then be stored in an interleaved vertical order, which may slow their access in further stages of the compression algorithm.

Besides a high compression ratio for a given image quality, other features are important in a good image codec. Especially useful are codecs that allow progressive encoding with an embedded bitstream, such as the embedded zerotree wavelet (EZW) [1] and set partitioning in hierarchical trees (SPIHT) [2] codecs. With embedded bitstreams, the coefficients are encoded in bit planes, with the most significant bit planes being transmitted first. In that way, the decoder can stop decoding at any point in the bitstream, and it will recover an image that has the fidelity level corresponding to how many bit planes were received. Also, within each bit plane the WT coefficients are encoded into increasing order of resolution. In that way, the encoding for the same image at a lower resolution (e.g. half the pixels in both horizontal and vertical directions) can be obtained simply by parsing each bit plane and removing in each the trailing end bits corresponding to the highest WT resolution levels.

Embedded encoding can be achieved by bit-plane encoding [6] of scalar-quantized wavelet coefficients. The most significant bit planes will naturally contain many zeros, and therefore can be highly compressed via entropy coders such as arithmetic or even run-length coders. Bit-plane encoding is more efficient if we reorder the wavelet coefficient data in such a way that coefficients with small absolute values tend to get clustered together, increasing the lengths of the zero runs in the bit planes. Data structures such as significance trees (or zerotrees) [3] are very efficient in achieving such clustering of zeros. They are used in EZW, SPIHT, and other WT-based codecs [1]–[3].

An efficient approach for embedded bit-plane coding of wavelet coefficients was presented in [7]. Instead of significance trees, a spatio-temporal neighborhood relationship is used to group the bit planes in four subsequences. Each subsequence is then encoded with an elementary Golomb (EG) coder, which is an effective run-length encoder [8]. The algorithm in [7] performs quite closely to SPIHT, but has a lower complexity and can encode all bit lanes through a single pass through the wavelet coefficients. Another approach for bit-plane encoding of ordered wavelet coefficients is presented in [9].

In [10] we presented PWC (progressive wavelet coder), an algorithm for embedded WT-based image coding in which the zero clustering of WT coefficients is achieved simply by reordering blocks of wavelet coefficients in a data-independent fashion, without the use of significant trees. Such a simple algorithm actually matches the performance of SIPHT with binary symbol encoding [10]. Thus, we can say that the combination of coefficient reordering and adaptive run-length coding in PWC achieves roughly the same compression gain as the significance tree used in SPIHT (the bit rate in SPIHT can be reduced by an extra 5% by using adaptive arithmetic encoding, at the expense of increased computational complexity).

In this paper we introduce the PTC coder, in which we replace the wavelet transform of PWC by a particular structure of the hierarchical lapped transform (HLT) [11], in which we cascade 8×8 lapped biorthogonal transforms (LBT) [12] with 8×8 DCTs applied to the LBT coefficients. The HT coefficients are reordered in a way inspired by [14], so they form a space-frequency decomposition similar to that of a six-level WT. In that way, we maintain the scalability (in fidelity and resolution) and embedded bitstream features of PWC, while reducing considerably (by a factor of about two or more) the computational complexity of the space-frequency decomposition. Also, since the main computation engine of PTC is an 8×8 2-D DCT, specialized software or hardware modules for the 2-D DCT, such as those used in JPEG codecs, can be leveraged for PTC. This significant complexity reduction of PTC comes at a small price: a reduction of about 2.7% in coding efficiency. In many applications, it is worthwhile to sacrifice 2.5% of file sizes for a significant reduction in computational complexity.

In Section 2 we review the basics of the LBTs, and present the hierarchical construction used in the PTC codec, which is described in Section 3. We compare the computational complexity and performance of PTC, PWC, and SPIHT in Section 4. Finally, general conclusions are presented in Section 5.

2. Lapped Biorthogonal Transforms

The lapped orthogonal transform (LOT) [11], [12], was developed as an alternative to the DCT with reduced blocking artifacts and increased coding gain. By combining DCT coefficients of adjacent image blocks in an appropriate way, the LOT in fact projects the image onto a set of overlapping basis function that maintain orthogonality not only within a block but between neighboring blocks (“orthogonality of the tails” [11]). The functions decay to near zero at their ends, reducing blocking effects considerably. The extended region of support of the LOT basis functions (16 pixels for length-8 blocks) also lead to a small increase (~ 1dB) of the transform coding gain.

Blocking effects are strongly attenuated with the LOT, but not totally eliminated, because the low-frequency basis functions still have small discontinuities. One way to completely eliminate blocking effects is to use the lapped biorthogonal transform (LBT), which is based on a few modifications of the original LOT computation flowgraph [13]. The LBT flowgraph is shown in Figure 1. The construction of only one block is shown; the others are obtained simply by replicating the structure. Image boundaries are handled by a simple modification of the first and last blocks, as discussed in [11], so that no explicit data reflection is necessary and all operations are performed in-place. We form the LBT of a block by combining the DCT coefficients of adjacent blocks, mostly through trivial +1/–1 butterflies. The scale factors $\{a, b, c\}$ control the shape of the basis functions. As long as we use the inverse scaling factors $\{1/a, 1/b, 1/c\}$ in the inverse transform (right-to-left in Figure 1), the transform is guaranteed to be biorthogonal, i.e. in absence of quantization the input data is recovered exactly by the inverse transform.

The original LBT construction in [13] included only the scaling factors a and $1/a$. By introducing the additional scaling factors b and c we maximize the coding gain under the assumption that all coefficients are quantized with the same step size. The optimal scale factors are shown in the Table 1.

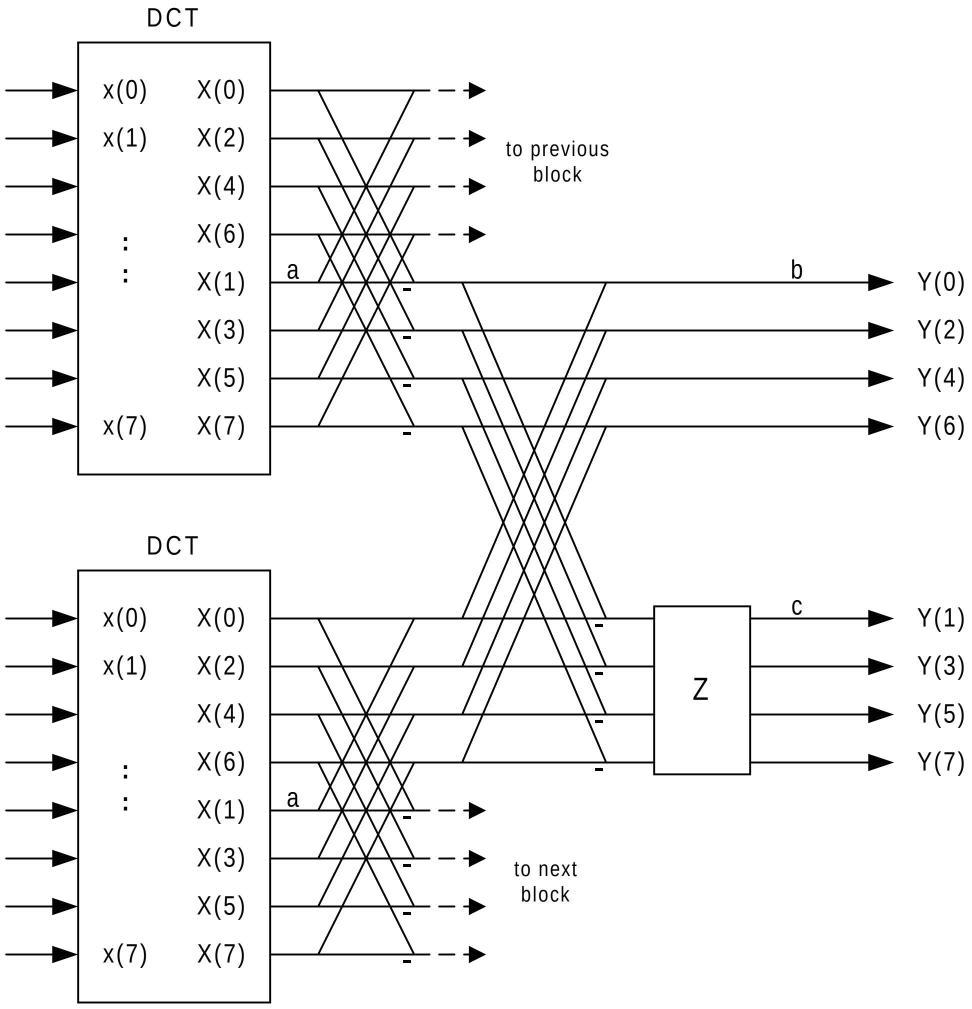

Figure 1. Flowgraph of the lapped biorthogonal transform. *Z* is a 4×4 orthogonal matrix, as described in [11]. The scale factors {a, b, c} control the shape of the basis functions. The inverse LBT is obtained with the same flowgraph, with processing from right to left and {a, b, c} replaced by {1/a, 1/b, 1/c}.

Parameter	Direct Transform	Inverse Transform
a	$\sqrt{2}$	$\sqrt{1/2}$
b	$\sqrt{3/4}$	$\sqrt{4/3}$
c	$\sqrt{4/5}$	$\sqrt{5/4}$

Table 1. Optimal scaling factors for the LBT

The LBT can be used in place of the DCT in any block-transform-based image coder. In fact, if we replace the DCT in JPEG by the LBT, blocking effects are eliminated and compression performance is improved, as reported in [13].

In order to achieve resolution scalability and to use a block transform in a codec originally designed for wavelets, we can reorder the block transform coefficients as shown in Figure 2. That ordering is similar to the one presented in [14], but not identical. It allows us to consider an 8×8 block transform as if it were a 3-level wavelet decomposition.

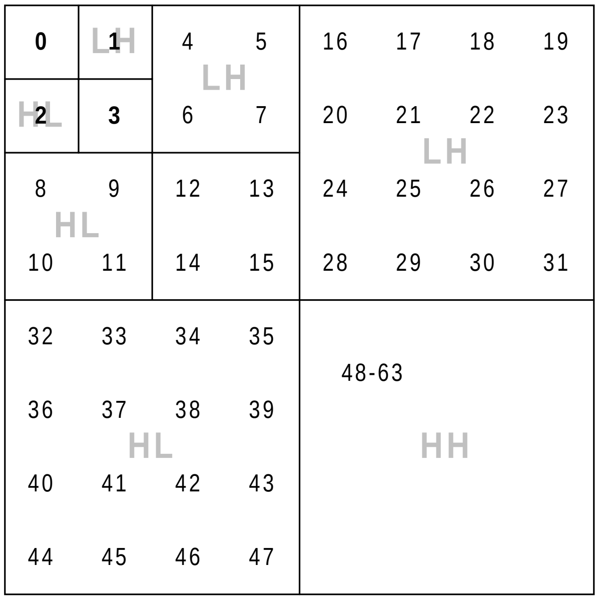

Figure 2. Coefficient reordering to map a block 8×8 transform into a 3-level wavelet-like tree. For example, the top right 4×4 block is sent to the LH image, and the 2×2 block to its left is sent to the LH image of the previous resolution level.

2.1. LBT-based Hierarchical Transform

In order to achieve the typical six levels of space-frequency resolution of a wavelet codec such as PWC, we can further decompose the LL image formed by all DC coefficients of the 8×8 LBT blocks. However, since blocking artifacts are already removed in the first LBT stage, we can use a standard DCT in this second decomposition. The idea is shown in the simplified diagram of Figure 3. Such hierarchical decompositions are discussed in more detail in [11].

The DCT coefficients of the second level of the hierarchical transform in Figure 3 are also reordered according to the mapping n Figure 2. In that way, we obtain a full six-level wavelet-like space-frequency decomposition. In the decoder, if we want to stop at any level of resolution, we simply zero the coefficients of the remaining LH, HL, and HH bands. Interpolation of the resulting image to the original resolution is automatically achieved by the inverse DCT/LBT transform.[1]

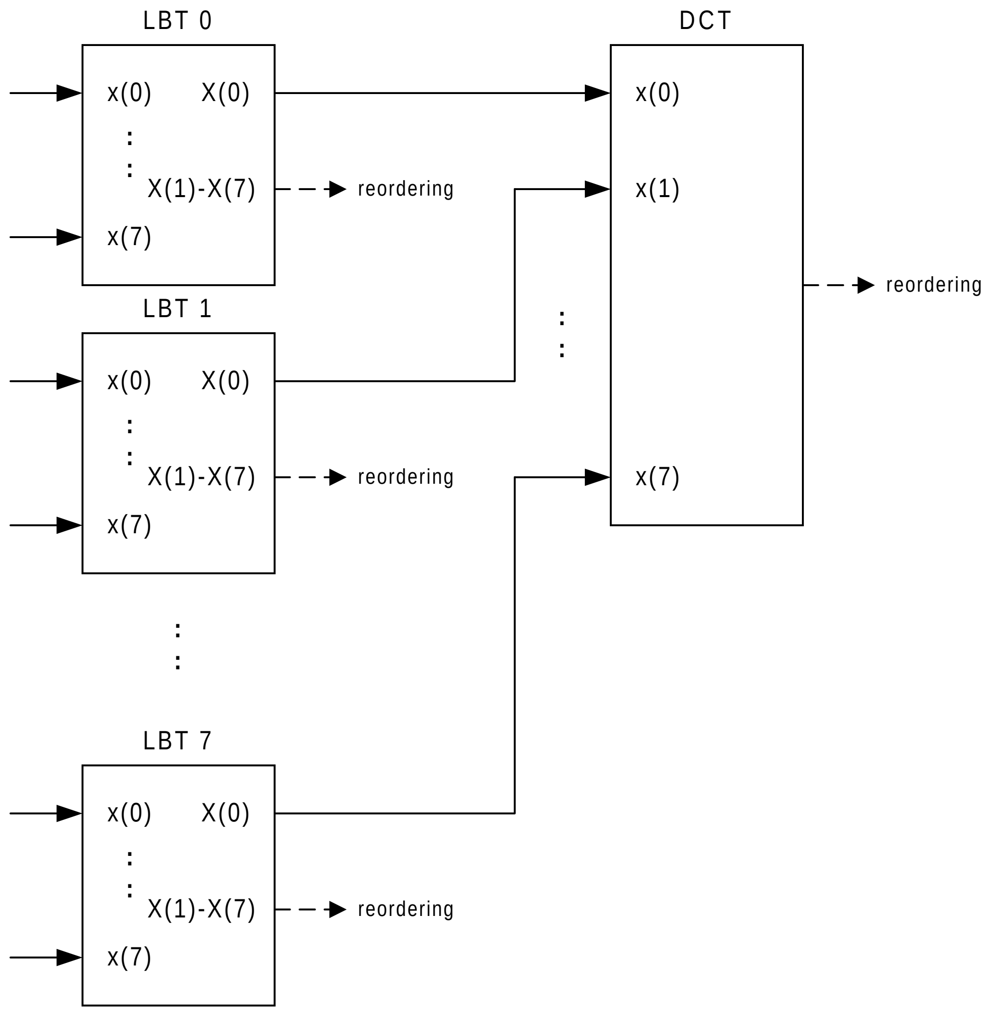

Figure 3. Six-level multiresolution decomposition (hierarchical transform) used in PTC. The DC coefficients of every eight LBT blocks are combined for the next transform stage. The DCT outputs are then reordered again to form the next three levels of the multiresolution decomposition.

[1] Note that the DCT basis functions are not very good interpolation filters. However, since the LBT functions are good filters [12] the interpolation procedure mentioned above works very well, typically with fewer aliasing artifacts than wavelet-based interpolation.

3. PTC Image Coding

In the previous section we described how to obtain a six-level space-frequency decomposition that's very similar to that of a wavelet transform, but using instead a hierarchical transform formed by LBTs followed by DCTs. A question that naturally arises is then: can we replace the 9/7 WT used in most wavelet codecs by the hierarchical LBT/DCT transform, without any other changes?

The answer is definitely positive. In fact, that is the main idea behind the embedded zerotree DCT (EZDCT) codec in [14]. EZ-DCT uses a 3-level wavelet-like decomposition based on reordered DCT coefficients, and the same coding strategy as SPIHT. Although it achieves better compression than JPEG, EZ-DCT still suffers from blocking artifacts. By replacing the DCT by an LBT, we can build an EZ-LBT codec, as described in [13]. The EZ-LBT codec has no blocking artifacts and achieves compression ratios that are very close (within 5% or less) to those of SPIHT. A similar idea was also used in [16], with a different LBT and a 3-level only decomposition.

We define the PTC codec as a PWC codec in which we replace the 9/7 biorthogonal wavelet transform by the LBT/DCT hierarchical transform described in Section 2, as shown in Figure 4. All other characteristics of PWC ("ping-pong" block reordering and adaptive run-length encoding of the quantized coefficients) are not changed. Therefore, we refer the reader to [10] for the details of PWC.

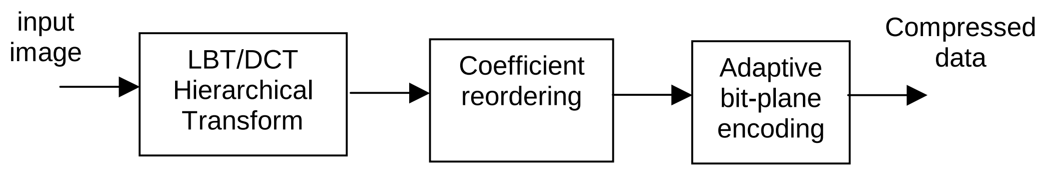

Figure 4. Simplified block diagram of a PTC image encoder.

4. Performance of the PTC Codec

We tested against the PWC [10] and SPIHT [2] codecs, using the grayscale versions of the 512×768 images in the Kodak test set [15]. Table 2 shows the resulting bit rate for a peak signal-to-noise ratio (PSNR) in the decoded images of 40.0 dB[2]. In that table, SPIHT-B refers to the binary encoded version, and SPIHT-A reefers to SPIHT with arithmetic coding. The Table also corrects a small error with respect to the PWC numbers reported in [10].

[2] The PSNR is defined as 20 log10 (255/e), in dB, where e is the root-mean-square reconstruction error.

Image	SPIHT - B	SPIHT - A	PWC	PTC
1	2.60	2.43	2.57	2.57
2	1.10	0.99	1.07	1.16
3	0.62	0.57	0.64	.63
4	1.09	1.01	1.07	1.09
5	2.34	2.20	2.36	2.45
6	1.77	1.66	1.77	1.76
7	0.80	0.72	0.80	0.85
8	2.63	2.45	2.61	2.71
9	0.74	0.68	0.77	0.77
10	0.87	0.80	0.90	0.92
11	1.58	1.47	1.58	1.63
12	0.87	0.79	0.88	0.88
13	3.23	3.04	3.17	3.23
14	2.07	1.92	2.03	2.1
15	0.94	0.86	0.96	0.97
16	1.17	1.09	1.17	1.17
17	1.03	0.95	1.05	1.08
18	2.07	1.93	2.04	2.07
19	1.47	1.37	1.44	1.45
20	0.78	0.71	0.80	0.85
21	1.47	1.37	1.46	1.50
22	1.54	1.44	1.51	1.52
23	0.38	0.35	0.39	0.42
24	1.89	1.77	1.88	1.97
Average	1.46	1.36	1.45	1.49

Table 2. Bit rates in bits/sample for image encoding algorithms at 40 dB PSNR. SPIHT-A uses arithmetic coding, and SPIHT-B uses binary encoding. PWC is described in [10] and PTC is the proposed coder. For the same data set and fidelity, JPEG uses an average of 1.62 bits/pixel.

From the results in Table 2 we see that the performance of the PTC codec is close to that of PWC (which in turn is almost identical to that of SPIHT-B). For the Kodak data set, PTC is about 2.7% worse than PWC, but still about 9.1 % better than JPEG. We have to be careful in comparing these numbers, since constant PSNR does not mean constant quality. Among the wavelet codecs using the same WT filters, same PSNR means same quality, because the reconstructed images are identical for the same PSNR. However, for PTC the reconstructed image looks different from that generated by a wavelet codec. One example is shown in Figure 5. We note that whereas a wavelet codec tends to blur some horizontal and vertical features, PWC preserves those better. On the other hand, PTC has a little more ringing than PWC. On informal tests, it seems that PTC produces a slightly sharper picture for the same PSNR as PWC. That may compensate for the ~3% compression ratio gap.

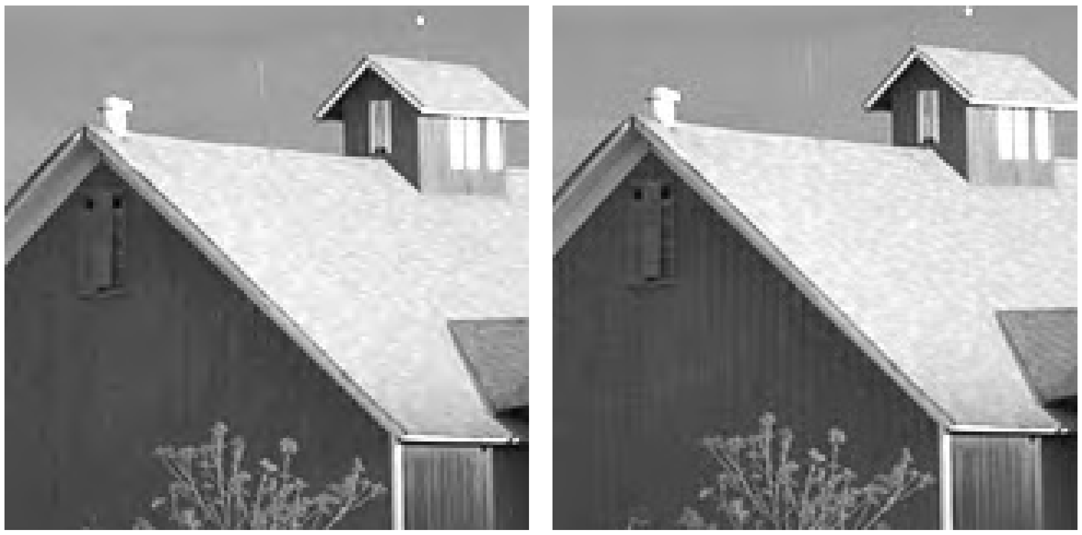

Figure 5. Decoded image # 22, PSNR = 36 dB. Left: PWC, Right: PTC.

The main advantage of PTC over PWC or other wavelet codecs is in the complexity of its space-frequency decomposition. Table 3 shows that the LBT/DCT hierarchical transform used in PTC reduces the number of multiplications per pixel by a factor of 2.7, while reducing the number of additions by about 14%.

Transform	**Multiplications per input pixel**	**Additions per input pixel**
9/7 Wavelet (via lifting)	15.8	15.8
LBT/DCT	5.8	13.7

Table 3. Computational complexity comparison: 9/7 WT versus LBT/DCT.

An alternative way to reduce the computational complexity in a wavelet codec would be to use shorter WT filters. For example, if we use a 5/3 biorthogonal WT [17] implemented via lifting, the computational complexity drops by a factor of two, i.e. 7.9 multiplications and additions per input pixel, better then the PTC. However, form the results in [17] we estimate that the 5/3 WT leads to an increase in bit rate of about 15% (compared to a bit less than 3% for the PTC).

5. Conclusion

We have introduced a progressive transform coder (PTC), which achieves a distortion vs. rate performance in image coding comparable to wavelet-based codecs. The main advantage of the PTC codec is it simplicity and lower computational complexity. With a compression ratio performance penalty of about 3%, PTC uses less than half the number of multiplications and 14% fewer additions per input pixel than the 9/7 biorthogonal wavelet transform used in many wavelet codecs. PTC also produces slightly sharper images.

References

[1] J. Shapiro, "Embedded image coding using zerotrees of wavelet coefficients," *IEEE Trans. Signal Processing,* vol. 41, pp. 3445–3462, Dec. 1993.

[2] A. Said and W. A. Pearlman, "A new and efficient image codec based on set partitioning in hierarchical trees," *IEEE Transactions on Circuits and Systems for Video Tech.*, vol. 6, pp. 243–250, June 1996.

[3] G. M. Davis and S. Chawla, "Image coding using optimized significance tree quantization," *Proc. Data Compression Conference,* Snowbird, UT, Mar. 1997, pp. 387–396.

[4] W. B. Pennebaker and J. L. Mitchell, JPEG Still Image Data Compression Standard. New York: Van Nostrand Reinhold, 1992.

[5] A. Ortega and C. Chrysafis, "Line based, reduced memory, wavelet image compression," *Proc. Data Compression Conference,* Snowbird, UT, Mar. 1998, pp. 398–407.

[6] J. W. Shwartz and R. C. Baker, "Bit-plane encoding: a technique for source encoding." IEEE Trans. Aerospace Electron. Syst., vol. 2, pp. 385–392, July 1966.

[7] E. Ordentlich, M. Weinberger, and G. Seroussi, "A low-complexity modeling approach for embedded coding of wavelet coefficients," *Proc. Data Compression Conference,* Snowbird, UT, Mar. 1998, pp. 408–417.

[8] G. G. Langdon, Jr., "An adaptive run-length encoding algorithm," *IBM Tech. Discl. Bull.,* vol. 26, pp. 3783–3785, Dec. 1983.

[9] E Schwartz, A Zandi, and M Boliek. "Implementation of compression with reversible embedded wavelets," *Proc. SPIE 40th Annual Meeting*, vol. 2564-04, July1995.

[10] H. S. Malvar, "Fast progressive wavelet coding," *Proc. Data Compression Conference,* Snowbird, UT, Mar.–Apr. 1999, pp. 336–343.

[11] H. S. Malvar, *Signal Processing with Lapped Transforms.* Boston, MA: Artech House, 1992.

[12] Programs in "C" for fast computation of block and lapped transforms can be found at http://www.research.microsoft.com/~malvar/software/.

[13] H. S. Malvar, "Biorthogonal and nonuniform lapped transforms for transform coding with reduced blocking and ringing artifacts," *IEEE Trans. Signal Processing,* vol. 46, pp. 1043–1053, Apr. 1998.

[14] Z. Xiong, O. G. Guleryuz, and M. T. Orchard, "A DCT-based embedded image coder," *IEEE Signal Processing Letters*, vol. 3, pp. 289–290, Nov. 1996.

[15] The original files for the Kodak PCD set of test images are available in raw format at the site ftp://ipl.rpi.edu/pub/image/still/KodakImages/.

[16] T. D. Tran and T. Nguyen, "A progressive transmission image coder using linear phase uniform filter banks as block transforms," *IEEE Trans. on Image Processing,* vol. 8, pp. 1493–1507, Nov. 1999.

[17] J. D. Villasenor, B. Belzer, and J. Liao, "Wavelet filter evaluation for image compression," *IEEE Trans. on Image Processing,* vol. 4, pp. 1053–1060, Aug. 1995.