Design Options on the Development of a New Tree-Based Wavelet Image Coder

Jose Oliver and M.P. Malumbres

Department of Computer Engineering (DISCA),
Technical University of Valencia,
Camino de Vera 17, 40617 Valencia, Spain
{joliver, mperez}@disca.upv.es

Abstract. A wide variety of wavelet-based image compression schemes have been reported in the literature. Every new encoder introduces different techniques that can be taken into account for designing a new image encoder.
In this paper, we present several design options and optimizations that will be used along with the LTW algorithm. However, these optimizations can be applied to other image compression algorithms or implementations. They improve the LTW rate/distortion performance in about 0.4 dB without increasing its computational cost, showing the importance of selecting good design options during the development of a new image encoder. [1]

1 Introduction

Wavelet image encoders have been proved to be the best compression schemes in terms of rate/distortion (R/D) optimization. Many wavelet image encoders have been reported in the literature, and each of them proposes different strategies to achieve the best R/D performance, along with other desirable capabilities (embedded bit stream, fast processing, symmetric encoder/decoder, robust bit stream, etc.) Therefore, every new encoder introduces different techniques that can be taken into account when designing a new image encoder.

Hence, Shapiro's EZW [1] introduces the concept of zerotrees as a way to explode similarities between subbands in a wavelet decomposition. Another important strategy used by Shapiro is performing successive-approximation in order to achieve a specific bit rate, and thus an embedded bit stream. Said and Pearlman [2] take these ideas and propose a new technique: clustering of zerotrees. They propose grouping zerotree symbols at the significance map by means of a sorting algorithm, getting better R/D performance.

Recently, the JPEG 2000 standard was released [3]. The proposed algorithm does not use zerotrees, but it performs bit-plane processing with three passes per bit-plane achieving finer rate control. A new R/D optimization algorithm and

[1] This work was supported by the Generalitat Valenciana under grant CTIDIB/2002/019

a large number of contexts are also used. In general, some ideas introduced by previous authors can be complemented with new ones in order to overcome problems or deficiencies presented in previous proposals. In this way, in [4] we propose a different wavelet image encoder, called Lower-Tree Wavelet (LTW) coder, as a new alternative to improve the computational cost and R/D performance of previous wavelet-based image coders.

In section 2 a description of the LTW algorithm is shown in order to allow the reader to be able to understand the proposed optimizations. In section 3 new design options for the LTW are analyzed. Finally, in section 4 some conclusions are drawn.

2 The Lower Tree Wavelet (LTW) Encoder

Tree oriented wavelet image encoders are proved to efficiently transmit or store the wavelet coefficient set, achieving great performance results. In these algorithms, two stages can be established. The first one consists on encoding the significance map, i.e., the location and amount of bits required to represent the coefficients that will be encoded (significant coefficients). In the second stage, significant transform coefficients are encoded.

One of the main drawbacks in previous tree oriented wavelet image encoders is their high computational cost. That is mainly due to the bit plane processing at the construction of the significance map, performed along different iterations, using a threshold that focuses on a different bit plane in each iteration. Moreover, the bits of the significant coefficients are also bit plane processed.

The LTW algorithm is able to encode the significance map without performing one loop scan per bit plane. Instead of it, only one scan of transform coefficients is needed. The LTW can also encode the bits of the significant transform coefficients in only one scan.

The quantization process for LTW is performed by two strategies: one coarser and another finer. The finer one consists on applying a scalar uniform quantization to the coefficients, and it is performed before the LTW algorithm. On the other hand, the coarser one is based on removing bit planes from the least significant part of the coefficients. It belongs to the LTW encoder. We define *rplanes* as the number of less significant bits that are going to be removed from every transform coefficient in the LTW.

At the encoder initialization, the maximum number of bits needed to represent the higher coefficient (*maxplane*) is calculated and output to the decoder. The *rplanes* parameter is also output. With this information, we initialize an adaptive arithmetic encoder that will be used to transmit the number of bits required to encode each coefficient.

In the next stage, the significance map is encoded as follows. All the subbands are scanned in zig-zag order and for each subband all the coefficients are scanned in small blocks. Then, for each coefficient, if it is significant (i.e., it is different from zero if we discard the first *rplanes* bits) the number of bits required to

represent that coefficient is encoded with an adaptive arithmetic encoder. As coefficients in the same subband have similar magnitude, the adaptive arithmetic encoder is able to encode very efficiently the number of bits of significant coefficients. On the other hand, if a coefficient is not significant and all its descendents are not significant, they form a lower-tree, and the symbol $LOWER$ is encoded. This coefficient and all its descendents are fully represented by this symbol and are not processed any more. Finally, if the coefficient is insignificant but it has at least one significant descendent, the symbol $ISOLATED\_LOWER$ is encoded.

The last stage consists on encoding the significant coefficients discarding the first *rplanes* bits. In order to speed up the execution time of the algorithm, we do not use an arithmetic encoder in this part, what results in a very small lost in performance.

Notice that in this section the algorithm has only been outlined, and a complete and formal description of LTW and all of its parameters can be found in [4].

3 Design Options in LTW

During the development of a wavelet image encoder, many design options appear. In this section, we are going to analyze some of them. For the dyadic wavelet decomposition, Daubechies biorthogonal B9/7 filter will be applied. The standard Lena image from the RPI site (8 bpp, 512x512) will be used for all the tests.

3.1 Quantization Process

Let us focus on the quantization process. In section 2 it has been explained how the bit rate and its corresponding distortion factor can be modified by means of two quantization parameters, one finer (Q) and another coarser (*rplanes*). LTW is not naturally embedded, it is the price that we have to pay for the lower complexity. Instead of it, the bit rate is adjusted using two quantization parameters in a similar way as in the widely used JPEG standard. However, loss of SNR scalability is compensated with a different feature not present in EZW and SPIHT, spatial scalability, which is desirable in many multimedia applications. On the other hand, notice that SNR scalability can be achieved from spatial scalability through interpolation techniques.

Despite of the use of two quantization methods within the LTW, the final quantization of a wavelet coefficient can be expressed mathematically by the following equations (let us call c_Q the initial coefficient and the quantized coefficient):

$$if\ (c > 0)\ c_Q = \left\lfloor \left(\left\lfloor \frac{c}{2Q} + 0.5 \right\rfloor + K \right) / 2^{rplanes} \right\rfloor . \quad (1)$$

$$if\ (c < 0)\ c_Q = \left\lceil \left(\left\lceil \frac{c}{2Q} - 0.5 \right\rceil - K \right) / 2^{rplanes} \right\rceil . \quad (2)$$

$$if\ (c = 0)\ c_Q = 0\ . \tag{3}$$

Notice that these expressions include a uniform scalar quantization (with a constant factor K and rounding) and the quantization arising from the removal of *rplanes* bits in the LTW algorithm. The constant factor K can be used to take some border values out of the quantization dead-zone and should be a low value (0 or 1). On the other hand, the coefficients which are recovered on the decoder side will be (let us call c_R a coefficient recovered from c_Q):

$$if\ (c_Q > 0)\ c_R = (2((2c_Q + 1)2^{rplanes-1} - K) - 1)Q\ . \tag{4}$$

$$if\ (c_Q < 0)\ c_R = (2((2c_Q - 1)2^{rplanes-1} + K) + 1)Q\ . \tag{5}$$

$$if\ (c_Q = 0)\ c_R = 0\ . \tag{6}$$

In both dequantization processes, i.e. in the standard scalar dequantization and in the dequantization from the bit-plane removing, the c_R value is adjusted to the midpoint within the recovering interval, reducing the quantization error.

Observe that this quantization process is very similar to the strategy employed in JPEG 2000; a uniform quantizer with deadzone. However, some differences can be drawn. For example, we use two quantization parameters (*rplanes* and Q), applying the uniform quantization twice, and we always perform a midpoint reconstruction, while JPEG 2000 decoders use to apply a reconstruction bias towards zero.

3.2 Analyzing the Adaptive Arithmetic Encoder

As coefficients in the same subband have similar magnitude, and due to the order we have established to scan the coefficients, an adaptive arithmetic encoder [5] is able to encode very efficiently the number of bits of the transform coefficients (i.e. the significance map used by the LTW algorithm). That is why this mechanism is essential in the R/D performance of the encoder.

A regular adaptive arithmetic encoder uses one dynamic histogram to estimate the current probability of a symbol. In order to improve this estimation we can use a different histogram depending on the decomposition level of the wavelet subband. It makes sense because coefficients in different subbands tend to have different magnitude, whereas those in the same decomposition level have similar magnitude.

Table 1, column a) shows the performance of the original LTW published in [4], in terms of image quality (PSNR in dB) for different bit rates (in bpp). In column b) we can see the results of using a different histogram on every decomposition level. It results beneficial with no penalty in computational cost.

If we review the LTW algorithm in section 2, *maxplane* is the maximum number of bits needed to represent the higher coefficient in the wavelet decomposition, being this value the one used in the initialization of the arithmetic encoder. At this point, we can define $maxplane_L$ as the maximum number of bits needed to represent the higher coefficient in the descomposition level L. So

these values can be used to adjust the initialization of every arithmetic encoder, provided all $maxplane_L$ symbols are output to the decoder. The drawback introduced here is manifestly compensated by the improvements achieved, as shown in Table 1 column c).

As we mentioned previously, coefficients in the same subband have similar magnitude. In order to take better profit from this fact, different histograms may be handled according to the magnitude of previously coded coefficients, i.e., according to the coefficient context. Last column in Table 1 shows the result of establishing two different contexts: the first one is used when the left and upper coefficients are insignificant or they are close to insignificant, and the other context is applied otherwise.

Several other actions can be tackled directly on the adaptive arithmetic encoder. On the one hand, the maximum frequency count (see more details in [5]) proposed by the authors is 16384 (when using 16 bits for coding). Practical experiences led Shapiro to reduce this value to 1024. On the other hand, another parameter that can be adjusted is how many the histogram is increased with every symbol. If this value is greater than one, the adaptive arithmetic encoder may converge faster to local image features, but increasing it too much may turn the model (pdf estimation) inappropriate, resulting in poorer performance.

Table 1. PSNR (dB) with different options in the arithmetic encoder

opt/rate	a)	b)	c)	d)
1	40.12	40.19	40.26	40.38
0.5	37.01	37.06	37.12	37.21
0.25	33.93	34.00	34.07	34.16
0.125	31.02	31.07	31.13	31.18

3.3 Clustering Symbols

Transform coefficients may be grouped forming clusters. This is due to the nature of the wavelet transform, where a coefficient in a subband represents the same spatial location as a block of 2x2 coefficients in the following subband. Fig. 1 represents a significance maps in LTW. On the left column, the symbol L means $LOWER$, the symbol * represents a coefficient previously coded by any $LOWER$ symbols (and it does not need to be encoded), and any numeric value is a symbol representing the number of bits needed to encode that coefficient. Fig. 1a) lower map represents the last subband whereas upper map is just the previous one. We can easily check that each L symbol in this subband leads to a 2x2 * block in the last subband. Also, it can be observed that blocks of 2x2 $LOWER$ symbols tend to appear in the last subband. Thus, a new symbol that groups these blocks of L symbols will reduce the total number of symbols to be

encoded. This way, we can define a new symbol called $LOWER\_CLUSTER$, which aims at grouping clusters of L symbols. Fig. 1b) represents this significance map introducing $LOWER\_CLUSTER$ (C) symbols.

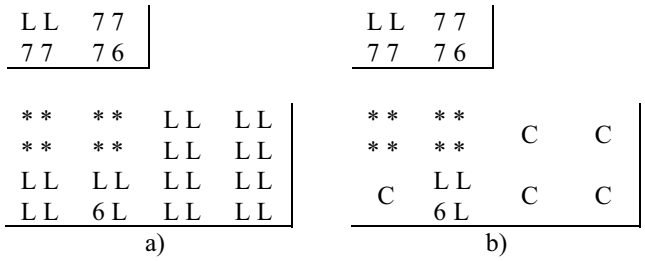

Fig. 1. Significance maps

Experimental tests using this strategy resulted in 1287 L symbols, 645 numeric symbols and 3435 new C symbols in the last subband. With the introduction of this new symbol, we passed from 15027 to 1287 L symbols, reducing the total number of symbols in 10305. However, the final file size increased from 7900 to 8192 bytes. In Table 1 this option corresponds with column b). As shown, introducing this symbol is not a good idea, if we compare it with column a), the best option of previous arithmetic encoder analysis. The reason is that if we do not use the new symbol, the L symbol is very likely (96%), whereas introducing the C symbol softens this probability concentration, with 64% for the C symbol and 24% for the L. As we know, an adaptive arithmetic encoder works better with probability concentration.

Table 2. PSNR (dB) with different clustering options

opt/rate	a)	b)	c)	d)
1	40.38	40.06	40.42	40.50
0.5	37.21	36.98	37.28	37.35
0.25	34.16	34.00	34.23	34.31
0.125	31.18	31.13	31.26	31.27

In order to overcome this problem, instead of a C symbol, two kind of numeric symbol can be used. A regular numeric symbol shows the number of bits needed to encode a coefficient, but a clustering numeric symbol not only indicates the number of bits but also the fact that the descendant cluster is a $LOWER\_CLUSTER$ (group of 2x2 L symbols). In this manner, likely symbols are not dispersed (because a particular numeric symbol do not use to be likely)

and clusters of L symbols are implicitly represented with no need to encode any symbol (anyhow, the numeric symbol was previously represented). In Table 2, column c) shows the advantage of using this method.

However, clustering may still be more favorable if it is used along with other strategies. For example, when three symbols in a cluster are $LOWER$, and the remaining symbol is close to be insignificant, the whole cluster may also be considered $LOWER\_CLUSTER$, supporting the construction of new lower-trees which will be encoded with just one symbol. Another similar nice strategy is saturating some coefficient values, from a perfect power of two to the same value decreased in one. In this manner, the introduced error for that coefficient is in magnitude only one, while it needs one less bit to be stored, resulting in higher final performance. These improvements are shown in the last column, in Table 2.

4 Conclusions

In this paper, we have presented several optimizations to be used along with the LTW algorithm. As shown in Table 3, they improve its R/D performance with Lena in about 0.4 dB, showing the importance of selecting good design options during the development of a new image encoder. R/D results for Goldhill are also shown in this table. Moreover, its computational cost is not increased, resulting from 2 to 3.5 times faster than JPEG 2000 and SPIHT, as it is shown in Fig 2 and 3. Larger images, like the Caf image included within the JPEG 2000 test images, offer even greater improvement in execution time. Notice that all the implementations are written under the same circumstances, using plain C++. The reader can easily perform new tests using the implementation of the LTW that can be downloaded from *http://www.disca.upv.es/joliver/LTW/LTW.zip*.

Table 3. PSNR (dB) with different bit rates and codecs

	Lena				GoldHill			
codec/ rate	SPIHT	JPEG 2000	Prev. LTW	LTW	SPIHT	JPEG 2000	Prev. LTW	LTW
1	40.41	40.31	40.12	40.50	36.55	36.53	36.40	36.74
0.5	37.21	37.22	37.01	37.35	33.13	33.19	32.98	33.32
0.25	34.11	34.04	33.93	34.31	30.56	30.51	30.39	30.67
0.125	31.10	30.84	31.02	31.27	28.48	28.35	28.41	28.60

In particular, LTW improves SPIHT in 0.15 dB (mean value), and Jasper [6], an official implementation of JPEG 2000 included in the ISO/IEC 15444-5, is also surpassed by LTW in 0.25 dB (mean value), whereas our algorithm is naturally faster and symmetric.

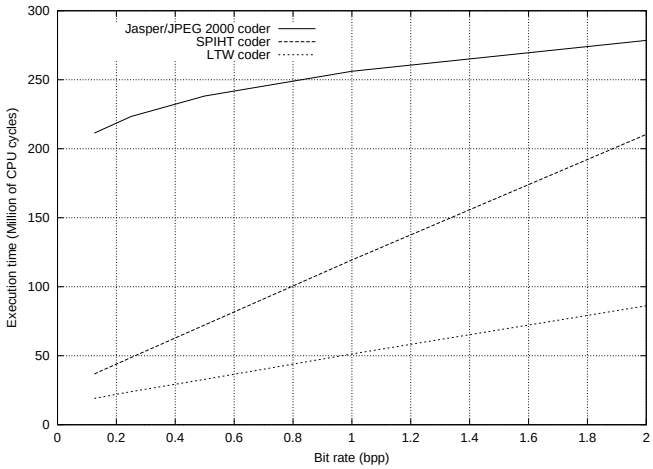

Fig. 2. Execution time comparision for coding Lena using Jasper/JPEG 2000, SPIHT and LTW

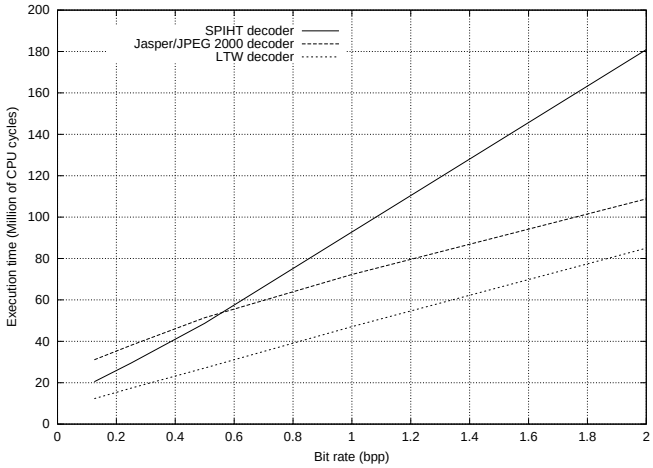

Fig. 3. Execution time comparision for decoding Lena using Jasper/JPEG 2000, SPIHT and LTW

References

1. J.M. Shapiro. Embedded Image Coding Using Zerotrees of Wavelet Coefficients, IEEE Transactions on Signal Processing, vol. 41, pp. 3445-3462, Dec. 1993.
2. A. Said, A. Pearlman. A new, fast, and efficient image codec based on set partitioning in hierarchical trees, IEEE Trans. circ. and systems for video tech., vol. 6, n 3, June 1996.
3. ISO/IEC 15444-1: JPEG2000 image coding system, 2000.
4. J. Oliver, M.P. Malumbres. A new fast lower-tree wavelet image encoder, IEEE International Conference on Image Processing, Thessaloniki, Greece, vol 3. pp. 780-784, Sept.2001.
5. I.H. Witten, R.M. Neal, J.G. Cleary. Arithmetic coding for compression, Commun. ACM, vol 30. pp. 520-540, 1986.
6. M. Adams. Jasper Software Reference Manual (Version 1.600.0), ISO/IEC JTC 1/SC 29/WG 1 N 2415, Oct. 2002.