Enhanced Non-embedded Lower Tree Wavelet Encoder

O. López, M. Martínez-Rach, P. Piñol, M.P. Malumbres[1] y J. Oliver[2]

Resumen—**In the last years, several authors have developed very fast and simple non-embedded wavelet encoders that are able to get reasonable good performance with reduced computing requirements. These encoders have lost the SNR scalability and precise rate control capabilities. In this paper, we propose a new non-embedded LTW codec version (E-LTW) with precise rate control method and good R/D performance due to the use of intra band neighboring context modeling for sign coding.**

Palabras clave—**Image coding, Sign coding, Wavelet transforms, Rate control.**

I. Introduction

WAVELET transforms have proven to be very powerful tools for image compression. Many state-of-the-art image codecs, including the JPEG2000 image coding standard, employ a wavelet transform in their algorithms ([1], [2]). At the beginning the EZW [3] began the race for efficient image coding by introducing the idea of wavelet coefficient-trees and successive approximations, which can be implemented with bit-plane coding. SPIHT [2] is an advanced version of EZW, where coefficient trees are processed in a more efficient way. In this coder, coefficient-trees are partitioned depending on the significance of the coefficients belonging to each tree. Both EZW and SPIHT need the computation of coefficient-trees to search for significant coefficients and they need several iterations focusing on a different bit-plane, which involves high computational complexity. In the final JPEG 2000 standard [1], the proposed algorithm does not use coefficient-trees, but it performs bit-plane coding in code-blocks, with three passes per plane, so the most important information is first encoded. In order to overcome the disadvantage of not using coefficient-trees, it also uses an iterative optimization algorithm, based on the Lagrange multiplier method, along with a large number of contexts, which are very time-consuming.

The use of complex mechanisms to improve coding efficiency makes the encoder very slow and typically with high memory demands. This issue may be a serious limitation for certain kind of applications like the ones related to real-time multimedia communication services, where the coding/decoding times are constrained to the application requirements (e.g., reduced shot speed in digital cameras). Several authors have considered this issue in order to design fast and low-memory consuming coders. Most of these works reduce coding complexity by removing the embedded feature of the final bitstream, among other optimization issues.

A non-embedded version of SPIHT was first proposed to reduce complexity in tree-based wavelet coding [4]. In this modified SPIHT, once a coefficient is found to be significant, all the significant bits are encoded, avoiding the refinement passes. In [5], authors propose the LTW codec that, with similar ideas plus some optimizations, avoids bit-plane processing with very low memory requirements and similar R/D performance to the one obtained by embedded encoders like JPEG2000 and SPIHT. PROGRES, another very fast non-embedded encoder, has been recently proposed [6], following the same ideas of [5], arranging the coefficients in order to achieve resolution scalability.

In this paper, we propose an enhanced version of the LTW encoder (E-LTW) that includes a sign coding tool and a new precise rate-control method. These new features will improve the R/D behavior providing non-embedded encoders with a high accurate rate control tool.

II. LTW encoder

For the most part, digital images are represented with a set of pixel values, P. The LTW encoder can be applied to a set of coefficients C resulting from a dyadic decomposition $\Omega(\cdot)$, so that $C = \Omega(P)$. The most commonly used dyadic decomposition for image compression is the hierarchical wavelet subband transform [7], therefore an element $C_{i,j} \in C$, is called transform coefficient. In a wavelet transform, we denote the subbands resulting from the first level of the image decomposition as LH_1, HL_1 and HH_1, corresponding to horizontal, vertical and diagonal frequencies. The rest of the image transform is performed by recursive wavelet decomposition on the remaining low frequency subband, until a desired decomposition level (N) is achieved (LL_N is the remaining low frequency subband).

One of the main drawbacks in previous wavelet image encoders is their high complexity. Many times, this is mainly due to bit plane processing, which is performed across different iterations, using a threshold that focuses on a different bit plane in each iteration. In this way, it is easy to achieve an embedded bit-stream with progressive coding, since the more bit planes are added the more SNR resolution is obtained in the recovered image.

Although embedded bit-stream with SNR scalability is a nice feature in an image coder, it is not always needed and other alternatives, like spatial scal-

[1]Dpto. Física y Arquitectura de Computadores, Univ. Miguel Hernández, e-mail: `otoniel, mmrach, pablop, mels @umh.es`

[2]Dpto. de Informática de Sistemas y Computadores, Univ. Politécnica de Valencia, e-mail: `joliver@disca.upv.es`

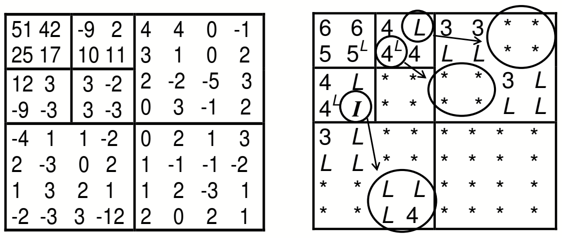
Fig. 1. Left: 2-level wavelet transform of an 8x8 example image, Right:Map Symbols

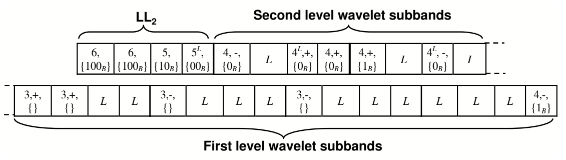

Fig. 2. Example image encoded using LTW

ability, may be more valuable according to the final purpose. In this section, we describe the LTW algorithm, which is able to encode wavelet coefficients without performing one loop scan per bit plane. Instead of it, only one scan of the transform coefficients is needed. Furthermore, in this section we present a very fast integer version of LTW.

In LTW, the quantization process is performed by means of two strategies: one coarser and another finer. The finer one consists of applying a scalar uniform quantization, Q, to wavelet coefficients. The coarser one is based on removing the least significant bit planes, *rplanes*, from wavelet coefficients.

A tree structure (similar to that of [2]) is used not only to reduce data redundancy among subbands, but also as a simple and fast way of grouping coefficients. As a consequence, the total number of symbols needed to encode the image is reduced, decreasing the overall execution time. This structure is called lower tree and it is a coefficient tree in which all its coefficients are lower than $2^{rplanes}$.

Our algorithm consists of two stages. In the first one, the significance map is built after quantizing the wavelet coefficients (by means of both Q and *rplanes* parameters). In Figure 1(right), we show the significance map built from wavelet decomposition shown at Figure 1(left). The symbol set employed in our proposal is the following one: a $LOWER$ symbol (L) represents a coefficient that is the root of a lower-tree, the rest of coefficients in a lower-tree are labeled as $LOWER\_COMPONENT$ ($*$), but they are never encoded because they are already represented by the root coefficient. If a coefficient is insignificant (i.e., lower than $2^{rplanes}$) but it does not belong to a lower-tree because it has at least one significant descendant, it is labeled as an $ISOLATED\_LOWER$ symbol (I). For a significant coefficient, we simply use a symbol indicating the number of bits needed to represent it (i.e. 4). For a significant coefficient that is root of a lower-tree we use a special symbol indicating the number of bits needed to represent it with an L superscript (i.e. 4^L).

Let us describe the coding algorithm. In the first stage (symbol computation), all wavelet subbands are scanned in 2×2 blocks of coefficients, from the first decomposition level to the Nth (to be able to build the lower-trees from leaves to root). In the first level subband, if the four coefficients in each 2×2 block are insignificant (i.e., lower than $2^{rplanes}$), they are considered to be part of the same lower-tree, labeled as $LOWER\_COMPONENT$. Then, when scanning upper level subbands, if a 2×2 block has four insignificant coefficients and all their direct descendants are $LOWER\_COMPONENT$, the coefficients in that block are labeled as $LOWER\_COMPONENT$, increasing the lower-tree size.

However, when at least one coefficient in the block is significant, the lower-tree cannot continue growing. In that case, a symbol for each coefficient is computed one by one. Each insignificant coefficient in the block is assigned a $LOWER$ symbol if all its descendants are $LOWER\_COMPONENT$, otherwise it is assigned an $ISOLATED\_LOWER$ symbol. On the other hand, for each significant coefficient, a symbol indicating the number of bits needed to represent that coefficient is employed. However, if all descendants of a significant coefficient are insignificant ($LOWER\_COMPONENT$), we use a special symbol indicating the number of bits needed to represent it and a superscript L (4^L in Figure 1(right)).

Finally, in the second stage, subbands are encoded from the LL_N subband to the first-level wavelet subbands, as shown at Figure 2. Observe that this is the order in which the decoder needs to know the symbols, so that lower-tree roots are decoded before its leaves. In addition, this order provides resolution scalability, because LL_N is a low-resolution scaled version of the original image and as more subbands are being received, the low-resolution image can be doubled in size. In each subband, for each 2×2 block, the symbols computed in the first stage are entropy coded by means of an arithmetic encoder. Recall that no $LOWER\_COMPONENT$ is encoded. In addition, significant bits and sign are needed for each significant coefficient and therefore binary encoded.

More details about the coding and decoding algorithms, along with a formal description and an example of use can be found in [5].

III. Enhanced LTW encoder (E-LTW)

As mentioned before, the E-LTW encoder is based on the LTW coding engine. The new features included in this new codec are the use of intra band neighboring context modeling for sign coding and an improvement of the model-based rate control method proposed in [8] that turns it into a high accurate rate control tool.

A. Sign Coding

In the former wavelet image encoders, sign coding of wavelet coefficients was not considered, because those coefficients located at the high frequency subbands form a zero-mean process, and therefore positive and negative coefficients are equally probable.

Schwartz, Zandi and Boliek were the first authors to consider sign coding, using the sign of one neighboring pixel in their context modeling algorithm [9]. The main idea behind this approach is to find correlations along and across edges.

The HL subbands of a multi-scale 2-D wavelet decomposition are formed from low-pass vertical filtering and high-pass horizontal filtering. The high-pass filtering detects vertical edges, and so, the HL subbands contain mainly vertical edge information. Oppositely defined are the LH subbands, that contain primarily horizontal edge information.

As explained in [10], given a vertical edge in an HL subband, it is reasonable to expect the transform coefficients along this edge to be positively correlated. Neighboring coefficients along the edge are considered valuable context information, and are expected to have the same sign as the coefficient being coded. It is also important to consider correlation across edges, being the nature of the correlation directly affected by the structure of the high pass filter. For Daubechies' 9/7 filters, wavelet coefficient signs are strongly negatively correlated across edges because this filter is very similar to a second derivative of a Gaussian, so, it is expected that wavelet coefficients will change sign as the edge is crossed.

The sign neighborhood correlation depends on the subband type (HL,LH,HH) as Deever assesses in [10]. After the analysis of the neighborhood probability distribution, we have determined that for HL subband, the neighbors which sign is more correlated with the sign of the current coefficient are N (North), NN (North-North) and W (West)[1]. Taking into account symmetry, for LH subband, those neighbors are W, WW and N and for the HH subband are N, W and NW, looking for diagonal edges. At this point, for each subband type we have a maximum of 3^3 neighbor sign combinations because each coefficient can be positive, negative or insignificant.

After having analyzed all the possible neighbors combinations for each subband type and for each wavelet decomposition level, we could make a prediction of the current coefficient sign. With the prediction of the current coefficient ($\hat{SC}_{i,j}[k]$) based on the neighborhood sign information, we encode the result of sign prediction (success or failure). That is, a binary valued symbol from $\hat{SC}_{i,j}[k] \cdot SC_{i,j}$. In order to compress as much as possible this binary valued symbol, we have used two context for each subband type. So as to minimize the zero order entropy of both contexts, we have distributed all sign coding predictions from the neighborhood between them. The selection criteria is to isolate in one context those combinations with the highest correctness prediction probability and highest number of occurrences. The rest of combinations are grouped into the other context. However, there are certain combinations with low correctness probability but with a great amount of occurrences, so we have to heuristically determine the convenience of including them in the first context or not.

The maximum bit gain due to sign compression is 17.35% when implemented in LTW for Barbara at 1 bpp and the minimum gain is 5.42% for Lena image at 0.125 bpp. This gain, in terms of R/D, implies an improvement up to 0.25 dB, being greater the improvement at low and medium compression rates.

B. Rate Control

The new rate control method is based on a modified version of the Model-based rate control algorithm presented in [8]. Given a source image and the target bitrate, the proposed model should supply an accurate estimation of both quantization parameters (Q and $rplanes$). As presented in [8], the proposed rate control method has an intrinsic average error of 5% at 1 bpp and 9% at 0.125 bpp. It is important to say that the resulting bit-rate is always lower than the target one leading to a PSNR performance loss.

In the Model-based rate control algorithm, for each specific value of rplanes (from 2 to 7), the probability distribution of significant and insignificant symbols is calculated. This way, an estimation of the bit-rate produced by the arithmetic encoder and the number of bits required to store the sign and significant bits (which were raw encoded in original LTW) form the bit rate estimation (E_{bpp}). The resulting estimation gives a biased measure of the real bit rate for all operative bit-rate range (from 0.0625 to 1 bpp). The error is reduced with a correction factor calculated from the Kodak image set [11].

After that, the target bit-rate, T_{bpp}, will establish the proper value of rplanes ($E_{bpp}(rplanes) > T_{bpp} > E_{bpp}$
$(rplanes + 1)$). In order to determine the proper value of Q, it seems that the bit rate progression from the current $rplane$ to the next one follows a second order polynomial curve with a common minimum. So, with the estimated values ($E_{bpp}(rplanes)$, $E_{bpp}(rplanes + 1)$ and the curve minimum ($Kmin$), they build the corresponding expression that will supply the estimated value of Q for a given target bit rate.

The modified rate control method should take into account the new sign coding stage and the intrinsic model error. The idea is to fix the underestimation error in order to match the final bit-rate with the target one. To do that, we first estimate a 'Delta_Q' reduction factor in such a way that the resulting bit-rate is under the target one, but this time very close to it. Then, we append the number of bits required to match the target bit-rate to the bitstream, by using those bits of the significant coefficients that correspond to bit-planes lower or equal to the $rplanes$ quantization parameter (bits that were removed after applying $rplanes$ coarser quantization). These extra bits are appended to the bitstream in a bit-plane order (from bit-plane '$rplane$' to 0) scanning the significant coefficients from the low frequency subbands

[1]Due to the LTW scan order restrictions, when encoding a significant coefficient, its South and East neighbors are not available, so they cannot participate in the sign prediction contexts.

to the highest ones (see Figure 3).

In Figure 4, we show the bit-rate accuracy of the proposed rate control method for Bike test image. Although not shown, the behavior is very similar in the other test images being the relative error below 0.5%.

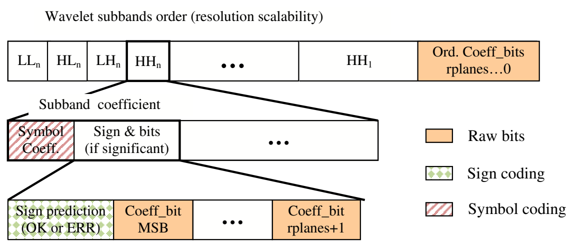

Fig. 3. E-LTW bistream format.

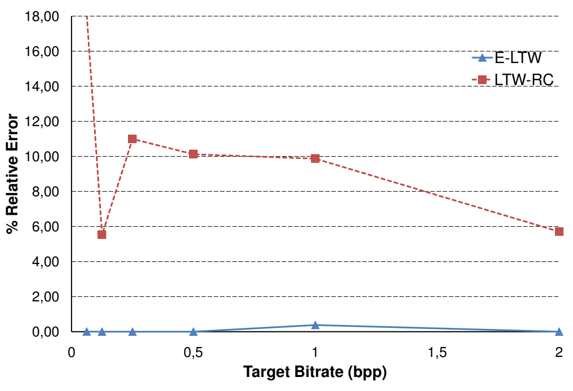

Fig. 4. Rate control accuracy for Bike test image.

TABLA I

PSNR (dB) GAIN FOR SEVERAL TEST IMAGES.

Bit-rate	E-LTW	LTW	Gain
Barbara (512x512)			
1	36.69	35.61	1.08
0.5	31.63	30.80	0.83
0.25	27.92	27.09	0.82
0.125	25.02	24.29	0.72
0.0625	23.28	23.02	0.26
Bike (2048x2560)			
1	37.82	36.97	0.85
0.5	33.28	32.36	0.92
0.25	29.45	28.40	1.04
0.125	26.09	25.09	1.00
0.0625	23.40	22.85	0.55

IV. RESULTS

In this section we analyze the behavior of the proposed encoder (E-LTW). We will compare E-LTW encoder versus JPEG2000 (Jasper 1.701.0), SPIHT (Spiht 8.01) and original LTW_RC (initial rate control version), in terms of R/D, coding and decoding delay and memory requirements. All the evaluated encoders have been tested on an Intel PentiumM Dual Core 3.0 GHz with 1 Gbyte RAM memory. The encoders binaries were obtained by means of Microsoft Visual C++ (2005 version) compiler with the same project options.

TABLA II

PSNR (dB) WITH DIFFERENT BIT-RATE AND CODERS.

Bit-rate	E-LTW	LTW-RC	JPEG2000	SPIHT
Barbara (512x512)				
1	36.69	35.61	37.11	36.41
0.5	31.64	30.80	32.14	31.39
0.25	27.92	27.09	28.33	27.58
0.125	25.02	24.30	25.25	24.86
0.0625	23.28	23.02	23.09	23.35
Lena (512x512				
1	40.34	40.34	40.38	40.46
0.5	37.29	36.76	37.27	37.25
0.25	34.18	33.65	34.05	34.15
0.125	31.14	30.59	30.82	31.10
0.0625	28.37	27.80	27.84	28.38

In Table I we show the PSNR (dB) gain when comparing E-LTW with LTW-RC. As it can be seen there is a great improvement in PSNR (1.08 dB for Barbara image). The maximum PSNR improvement obtained is 1.6 dB for Zelda image at 2 bpp. In Table II we show the E-LTW behavior in R/D when compared to SPIHT and JPEG2000. As shown, for high textured images like Barbara, JPEG2000 has a better behavior than E-LTW and SPIHT, being E-LTW better than SPIHT (up to 0.34 dB for Barbara image at 0.25 bpp). On the other hand, in images like Lena or Zelda, both SPIHT and E-LTW have a better behavior than JPEG2000 (up to 0.52 dB at high compression rates).

TABLA III

CODING DELAY (SECONDS) EXCLUDING DWT TIME.

Bit-rate	E-LTW	LTW-RC	JPEG2000	SPIHT
Barbara (512x512)				
1	0.051	0.037	0.080	0.042
0.5	0.031	0.023	0.076	0.026
0.25	0.026	0.018	0.074	0.018
0.125	0.015	0.010	0.073	0.014
0.0625	0.014	0.008	0.072	0.011
Cafe (2048x2560)				
1	0.914	0.648	2.623	0.920
0.5	0.527	0.382	2.543	0.521
0.25	0.349	0.225	2.507	0.323
0.125	0.198	0.158	2.518	0.221
0.0625	0.140	0.105	2.509	0.172

As it could be expected, the E-LTW uses more the arithmetic encoder than the LTW when coding the sign, so this fact implies a higher computational cost in the coding and decoding process as shown in Table III. The computational cost increase is a 40% on average. For high resolution images like Bike or Cafe, E-LTW still remains competitive respect to SPIHT and JPEG2000. Remark that the arithmetic encoder used in E-LTW is a non optimized version of Witten encoder [12], as oppossd to SPIHT that uses an optimized arithmetic encoder. In [13] an in depth comparison for several arithmetic encoders is presented and the arithmetic encoder used by SPIHT is at least six times faster than Witten's one. Notice that the arithmetic encoding process is responsible of 60% (E-LTW) and 51% (LTW) of the total encoding time. So, by using a faster arithmetic encoder implementation, the overall coding/decoding time would be significantly reduced on both encoders (see prediction in Table IV).

TABLA IV
Arithmetic encoder and total encoder time for Lena test image when coding at 2 bpp.

E-LTW (Arithmetic)	LTW-RC (Arithmetic)	E-LTW	LTW-RC
0.060	0.039	0.105	0.076
Reduction estimation (6 times fewer)			
0.010	0.006	0.054	0.043

In Table V, memory requirements of the encoders under test are shown. The original LTW-RC needs only the amount of memory to store the source image and an extra of 1.2 KB, basically used to store the histogram of significant symbols needed to accomplish the model-based rate control algorithm. On the other hand, the E-LTW version requires a few extra memory space to store the *rplanes* less significant bits of the significant coefficients. SPIHT requires more memory space than LTW-RC and E-LTW, and JPEG2000 needs twice the memory of LTW-RC for medium size images and three times for high definition images (results obtained with the Windows XP task manager, peak memory usage column).

TABLA V
Memory Requirements for evaluated encoders (KB)

Codec/image	SPIHT	JPEG2000	LTW-RC	E-LTW
Lena	3228	4148	2092	2212
Cafe	46776	65832	21632	25392

V. CONCLUSIONS

In this paper we present a new LTW encoder version (E-LTW), and we compare its performance with SPIHT and JPEG2000 encoders in terms of R/D performance, execution time and memory consumption. The E-LTW encoder exhibits good R/D performance (up to 0.52 dB at high compression rates compared to JPEG2000 and up to 0.34 dB for high textured images compared to SPIHT). The use of high context modeling for sign coding has a high computational cost (40% overhead) but E-LTW still remains competitive for high resolution images (similar coding time than SPIHT and up to 2.3 times faster than JPEG2000). Even more, the computational cost of our proposal is dominated by the arithmetic encoder processing, so there is enough room for becoming faster. Regarding memory requirements, E-LTW needs a little extra amount of memory to store the *rplanes* less significant bits. As future work we are planning to use a fast adaptive arithmetic encoder like [13] to significantly reduce the computational cost.

Acknowledgment

This work was funded by the Spanish Ministry of education and Science under grant DPI2007-66796-C03-03.

Referencias

[1] "JPEG2000 part I final int. std.," September 2000, ISO/IEC JTC 1/SC 29/WG 1 N1890.
[2] A. Said and A. Pearlman, "A new, fast and efficient image codec based on set partitioning in hierarchical trees," *IEEE Transactions on Circuits, Systems and Video Technology*, vol. 6, no. 3, pp. 243–250, 1996.
[3] J.M. Shapiro, "A fast technique for identifying zerotrees in the EZW algorithm," *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, vol. 3, pp. 1455–1458, 1996.
[4] W. A. Pearlman, "Trends of tree-based, set partitioning compression techniques in still and moving image systems," in *Picture Coding Symposium*, March 2001.
[5] J. Oliver and M. P. Malumbres, "Low-complexity multiresolution image compression using wavelet lower trees," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 16, no. 11, pp. 1437–1444, 2006.
[6] Yushin Cho, W. A. Pearlman, and A. Said, "Low complexity resolution progressive image coding algorithm: PROGRES (progressive resolution decompression)," in *IEEE International Conference on Image Processing*, September 2005.
[7] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies, "Image coding using wavelet transform," *IEEE Transactions on Image Processing*, vol. 1, no. 2, pp. 205–220, 1992.
[8] O. López, M. Martinez-Rach, J. Oliver, and M.P. Malumbres, "Impact of rate control tools on very fast non-embedded wavelet image encoders," in *Visual Communications and Image Processing 2007*, January 2007.
[9] Edward L. Schwartz, Ahmad Z, and Martin Boliek, "CREW: Compression with reversible embedded wavelets," in *In Proc SPIE*, 1995, pp. 212–221.
[10] Aaron Deever and Sheila S. Hemami, "What's your sign?: Efficient sign coding for embedded wavelet image coding," in *Proc. IEEE Data Compression Conf., Snowbird, UT*, 2000, pp. 273–282.
[11] CIPR, "http://www.cipr.rpi.edu/resource/stills/kodak .html," Center for image processing research.
[12] I.H. Witten, R.M. Neal, and J.G. Cleary, "Arithmetic coding for data compression," *Commun. ACM*, vol. 30, no. 6, pp. 520–540, 1987.
[13] Amir Said, "Comparative analysis of arithmetic coding computational complexity," Tech. Rep., Hewlett-Packard Laboratories HPL-2004-75, 2004.