E-LTW: AN ENHANCED LTW ENCODER WITH SIGN CODING AND PRECISE RATE CONTROL

O.López, M.Martínez,P.Piñol,M.P.Malumbres

Universidad Miguel Hernández
Physics and Computer Science
Avda. Universidad s/n, 03202 Elche, Spain

*J. Oliver**

Universidad Politécnica de Valencia
Computer Science
Camino de Vera s/n, 46022 Valencia, Spain

ABSTRACT

Traditional embedded coding systems involve high complexity algorithms, requiring fast and expensive processors. 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.

Index Terms— Image coding, Wavelet transforms

1. 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.

*Thanks to Spanish Ministry of education and Science under grant DPI2007-66796-C03-03 for funding.

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.

2. TEXTURE CODING: FAST & EFFICIENT LTW ENCODER

LTW is a tree-based wavelet image encoder, with state-of-the-art coding efficiency, but less resource demanding than other encoders in the literature. The basic idea of this encoder is very simple: after computing a dyadic wavelet transform of an image, the wavelet coefficients are first quantized and then encoded with arithmetic coding.

In LTW, the quantization process is performed by means of two strategies: one coarser and another finer. The finer

one consists in 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.

For the coding stage, if the absolute value of a coefficient and all its descendants (considering the classic quad-tree structure from [2]) is lower than a threshold value ($2^{rplanes}$), the entire tree is encoded with a single symbol, which we call *LOWER* symbol (indicating that all the coefficients in the tree are lower than $2^{rplanes}$ and so they form a lower-tree). But if a coefficient is lower than the threshold and not all its descendants are lower than it, that coefficient is encoded with an *ISOLATED_LOWER* symbol. On the other hand, for each wavelet coefficient higher than $2^{rplanes}$, we encode a symbol indicating the number of bits needed to represent that coefficient, along with a binary coded representation of its bits and sign (note that the *rplanes* less significant bits are not encoded).

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

3. 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 [7] that turns it into a high accurate rate control tool.

3.1. 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 [8]. The main idea behind this approach is to find correlations along and across edges.

As explained in [9], 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 [9]. 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.

3.2. Rate Control

The new rate control method is based on a modified version of the Model-based rate control algorithm presented in [7]. 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 [7], 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.

The modified rate control method should take into account

[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.

Bit-rate	E-LTW	LTW-RC	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

Table 1. PSNR (dB) gain for several test images.

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**

Table 2. PSNR (dB) with different bit-rate and coders.

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

Table 3. Coding delay (seconds) excluding DWT time.

the new sign coding tool 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 to the highest ones (see Figure 1).

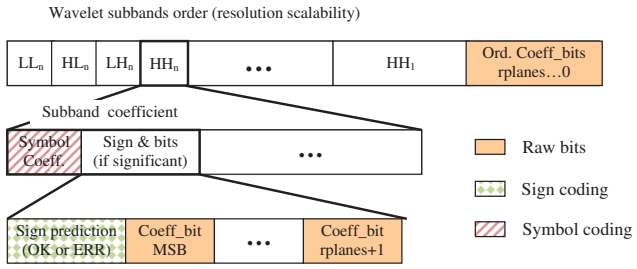

Fig. 1. E-LTW bistream format.

4. 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.

In Table 1 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[2]. In Table 2 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).

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 3. 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

[2]PSNR differences between LTW-RC and the original LTW presented in [5] are due to the use of higher *rplanes* values in the rate control stage on LTW-RC and E-LTW, whereas in Original LTW, the *rplanes* value is 2 and the finer quantization parameter Q grows without a limit.

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

Table 4. Arithmetic encoder time for Lena test image when coding at 2 bpp.

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

Table 5. Memory Requirements for evaluated encoders (KB)

used in E-LTW is a non optimized version of Witten encoder [10], as opposed to SPIHT that uses an optimized arithmetic encoder. In [11] 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 on Table 4).

In Table 5, 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 metric).

5. 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, coding delay 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 [11] to significantly reduce the computational cost.

6. REFERENCES

[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] 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.

[8] Edward L. Schwartz, Ahmad Z, and Martin Boliek, "CREW: Compression with reversible embedded wavelets," in *In Proc SPIE*, 1995, pp. 212–221.

[9] 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.

[10] 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.

[11] Amir Said, "Comparative analysis of arithmetic coding computational complexity," Tech. Rep., Hewlett-Packard Laboratories HPL-2004-75, 2004.