A Heuristic Bitrate Control for Non-embedded Wavelet Image Encoders

O. López [1], M. Martínez-Rach[1], J. Oliver [2], M.P. Malumbres[1]
[1] Department of Physics and Computer Engineering, Miguel Hernandez University
Ave. Libertad s/n, 03202 ELCHE (SPAIN)
[2] Department of Computer Engineering, Technical University of Valencia
Camino de Vera s/n, 46022 VALENCIA (SPAIN)
E-mail: *otoniel@umh.es*

Abstract – *Most recently developed wavelet image emcoders support as native feature an embedded bitrate control that could be useful for applications where obtaining an accurate bitrate is critical. Well-known encoders like JPEG2000 and SPIHT employ embedded coding. However, this feature makes them highly complex, being a serious restriction for certain applications and devices (such as mobile multimedia communications involving PDAs, mobile phones, etc.). As a consequence, in the last years, authors proposed non-embedded encoders with very fast and low-memory-demanding encoding/decoding processes. In this paper, we propose rate control tools that are able to supply the required rate control accuracy with as low as possible complexity increase. Experimental tests show that good results are obtained by employing trivial models of the encoder engine, being able to modulate the accuracy degree with lightweight iterative versions.*

Keywords – *Wavelet image coders, fast image coding, rate control , mobile multimedia communications*

1. INTRODUCTION

The development of wavelet image encoders has followed different evolution steps. At the beginning the EZW [1] kicked off 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. If a coefficient in a tree is higher than a threshold determined by the current bit-plane, the tree is successively divided following some established partitioning rules. Both EZW and SPIHT need the computation of coefficient-trees to search for significant coefficients and they take various iterations focusing on a different bit-plane, which involves high computational complexity. In the final JPEG 2000 standard [3], 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.

As shown before, the use of complex mechanisms to improve coding efficiency and functionality 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 with 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]. PROGRES follows the same ideas of [5], arranging the coefficients in order to achieve resolution scalability.

In this paper, we propose three lightweight rate control tools for non-embedded encoders. These tools will predict the proper quantization values that lead to a final bitrate close to the target one. In particular, we propose several bitrate prediction methods with increasing complexity and accuracy. We have chosen LTW [5] for evaluation purposes, although other encoders with scalar quantization could also be used.

Our first proposal extracts some features from the source image and finds correlations with the quantization parameter for a specific target bitrate based on a standard set of representative images.

Afterwards, we define a simplified model of the encoding engine to determine an initial estimation that will be adjusted by means of curve fitting techniques based on a standard set of images. And finally, to increase the accuracy of the previous method, we propose a lightweight iterative version for bounding the estimation error.

2. LOWER-TREE WAVELET (LTW) CODING

In LTW [5], the quantization process is performed by 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.

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). The symbol set employed in our proposal is the following one: a *LOWER* symbol 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 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. For a significant coefficient, we simply use a symbol indicating the number of bits needed to represent it.

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 N^{th} (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.

Finally, in the second stage, subbands are encoded from the LL_N subband to the first-level wavelet subbands. 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,.

3. PROPOSED BITRATE CONTROL TOOLS

3.1. First-order entropy based rate control

The first method is based on the extraction of the first-order entropy. The estimation process is based on the correlation between entropy, bitrate and quantization parameters.

We employ the KODAK image set as a representative set for our purposes, and the LTW encoder with both Q and *rplanes* quantizers. As expected, there is a correlation between the source image entropy and the quantization parameters. Therefore, we can establish a relationship between the quantization parameters and the entropy for a given target bitrate by using curve and surface fitting techniques. Although this estimation method suffers for severe errors when working at low and high entropy values, the computational complexity is very low. First of all, the corresponding algorithm will determine an appropriate value for *rplanes* given a target bitrate and a source image entropy. Afterwards, the Q value will be adjusted through curve fitting, taking into account again the target bitrate and the corresponding entropy value.

3.2. Rate control based on a trivial coding model

It is very difficult to estimate with certain degree of accuracy the target bitrate by using only the image entropy. So, we decided to study how the encoder works in order to define a simplified model of the encoding engine. This model will lead us to an initial and fast estimation of the resulting bitrate for different values of the *rplanes* (from 2 to 7). In this

model, for each specific value of *rplanes*, the probability distribution of significant and no insignificant symbols is calculated. This way, an estimation of the bitrate produced by the arithmetic encoder and the number of bits required to store the sign and significant bits (which are binary coded) form the bitrate estimation (E_{bpp}). The resulting estimation gives a biased measure of the real bitrate for all operative bitrate range (from 0.0625 to 1 bpp). We reduce the error with a correction factor calculated from the image set.

After that, the target bitrate, 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, we noticed that the bitrate progression from current *rplane* to the next one follows a second order polynomial curve that share near the same minimum Q value for all the test images in the set (as shown in Fig. 1). Since we know two points of that curve (E_{bpp}(rplanes) and E_{bpp}(rplanes+1), we can build the corresponding expression that will supply the estimated value of Q for a given target bitrate.

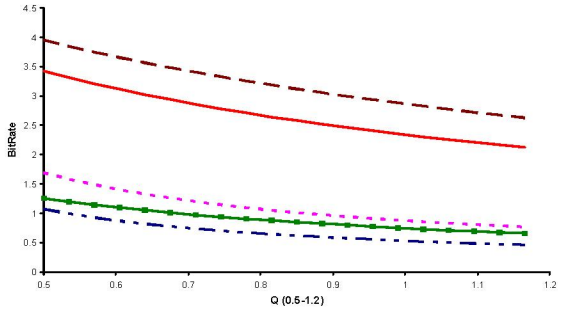

Fig. 1. Bitrate progression of five images from KODAK set from 2 to 3 *rplanes*.

3.3. Lightweight Iterative Rate Control

With the rate control described in the previous subsection, we can define an iterative version to reduce the estimation error with a moderate increase of computational complexity. Thus, depending on the application restrictions, we can get the proper trade-off between both factors: complexity and accuracy in the prediction. Now, we can define the maximum estimation error, and the algorithm will perform iterations until this condition is satisfied or a maximum number of iterations is reached.

In the first iteration, the proposed algorithm will estimate the *rplanes* and Q values for the target bitrate by using the algorithm described in the previous subsection. Then the source image will be coded with the quantization parameters found. If the resulting estimation error is lower than the maximum allowed, then the algorithm finishes, else we make a new Q estimation with the same *rplane* value based on the observed error. If we has reached the maximum number of iterations, the algorithm finishes, else the source image is encoded again with the new Q an a new iteration is performed

Although non-converging images were found in the Kodak set, the algorithm is able to detect non-converging estimations.

4. EXPERIMENTAL RESULTS

Using a C++ implementation of the LTW encoder, the different proposals were developed and tested on an Intel PentiumM 1.6 Ghz Processor. As a reference for the evaluation we have included a greedy rate control mechanism (FindRate). For determining the curve fitting and error adjustments in the first two methods we used the KODAK image set as a representative set of natural images. We restrict the limits of operation of our proposals in the range of 0.0625 to 1 bpp. The trade-off between precision and computational complexity will be also analyzed through the following expression:

$$E = \frac{1}{1 + \alpha R_{err} + (1-\alpha)T_E / T_C} \quad (1)$$

where R_{err} is the relative rate control error, T_E and T_C are the rate control estimation and coding times, respectively, and α determines the impact of accuracy and complexity on the efficiency computation (for our purposes α=*0.5*). Finally, we used Lena, Barbara, Godhill and Peppers test images (outside the Kodak set) to validate the proposed methods.

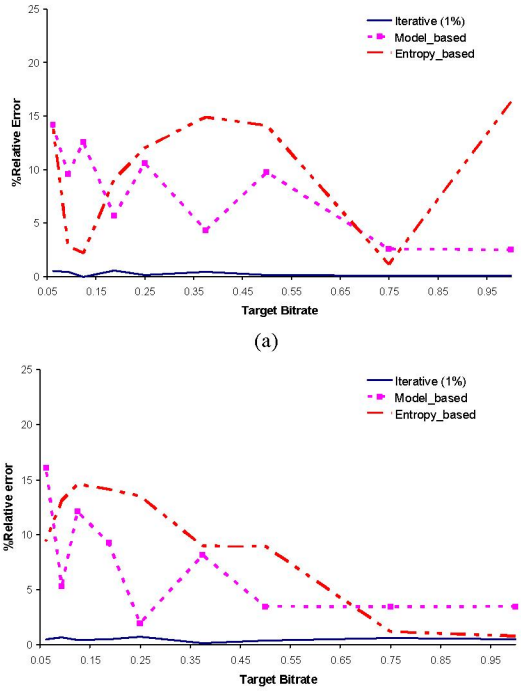

Fig. 2. Accuracy with (a) Lena and (b) Godhill.

In Fig.2, we show the accuracy of the proposed rate control methods with Lena and Godhill test images. Although not shown, the behavior is very similar in other test images. In general, the model-based method does not work efficiently at very low bitrates in the range from 0.0625 to 0.125 bpp. This behavior is due to the simplicity of the defined model, where there is no symbol differentiation in the insignificant coefficients set. So, at very low bitrates, there are a considerable number of coefficients that will not be encoded because they belong to a lower-tree, but which are not detected in the estimation. However, at moderate and low compression rates, the model-based proposal is more accurate than the one based on entropy.

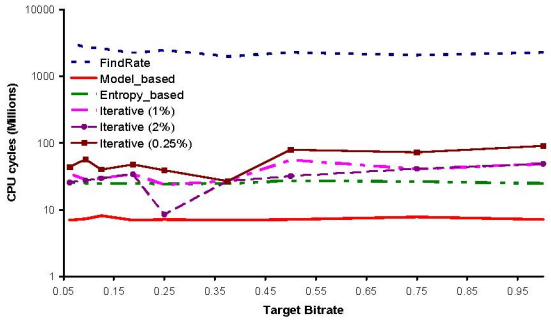

Fig. 3. Computational cost evaluation of different proposals with Godhill test image

In Fig.3, we measure the computational cost (in CPU cycles) of the proposed methods when coding the Godhill test image. As it can be seen, the model-based method is the fastest one, due to the simplicity of computations required for issuing an estimation.

The entropy-based proposal is 3 times slower than model-based, mainly due to the higher complexity of float type computations. In the case of iterative versions, we can observe that with a maximum 2% error, we obtain a very fast rate control estimator (even faster that the entropy based). The greedy reference (FindRate) is too complex for our purposes so we dismiss it. Also we can state that the computational cost is not dependent of the target bitrate, although in the iterative versions, the number of iterations may produce some deviations.

By using the expression (1), we evaluate the different proposals with a balance factor, α, equal to 0.5, in order to give the same importance to the accuracy and complexity of the evaluated proposals.

As shown in Fig. 4, for Lena test image, the best option will be the use of and iterative version at 2% of maximum error precision. This behavior is similar in the rest of test images, being the iterative at 1% a very close candidate for high detailed images like Barbara. Note that this is a trade-off, so the user will decide if he/she prefers accuracy, complexity or both depending on the application requirements.

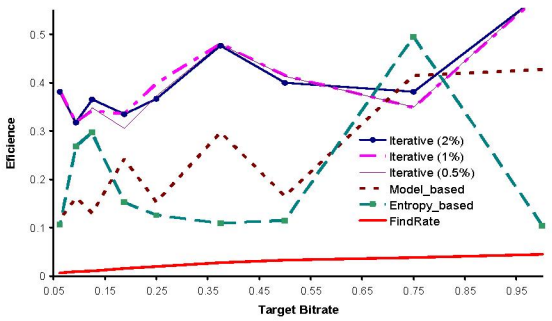

Fig. 4. Rate control efficiency for Lena test image.

5. CONCLUSIONS

We have proposed several rate control tools of increasing computational complexity, specially suited for non-embedded encoders that require for accurate and lightweight bitrate estimations. Also, the degree of accuracy may be tuned by the user taking into account the defined precision/complexity trade-off. The method based on an iterative version of a simplified model of the encoding process is the one that achieves best results. As future work, we are going to apply these methods to other non-embedded coders, and perform some refinements to the model-based method, in order to reduce even more the estimation error. Also, we are planning to apply these methods to video rate control.

ACKNOWLEDGEMENT

This work was funded by the Spanish Ministry of Education and Science under grant TIC2003-00339.

REFERENCES

[1] J.M. Shapiro, "Embedded Image Coding Using Zerotrees of Wavelet Coefficients," *IEEE Tran. on SP*, vol. 41, pp. 3445-3462, 1993.

[2] A. Said, A. Pearlman. "A new, fast, and efficient image codec based on set partitioning in hierarchical trees," *IEEE CSVT*, vol. 6, nº 3, pp.243-250, 1996.

[3] ISO/IEC 15444-1, JPEG2000 image coding system, 2000.

[4] W.A. Pearlman, "Trends of tree-based, set partitioning compression techniques in still and moving image systems," Picture Coding Symposium, pp. 1-8, 2001.

[5] J. Oliver, M. P. Malumbres, "Fast and Efficient Spatial Scalable Image Compression Using Wavelet Lower Trees," in Proc. IEEE Data Compression Conference, Snowbird, UT, 2003

[6] Y. Cho, W.A. Pearlman, A.Said, "Low complexity resolution progressive image coding algorithm: PROGRES," IEEE Int. Conference on Image Processing, 2005.