J Sign Process Syst (2012) 68:203–216
DOI 10.1007/s11265-011-0598-6

Rate Control Algorithms for Non-Embedded Wavelet-Based Image Coding

Otoniel Mario López Granado · Miguel Onofre Martínez-Rach · Pablo Piñol Peral · José Oliver Gil · Manuel Pérez Malumbres

Received: 29 December 2010 / Revised: 19 May 2011 / Accepted: 19 May 2011 / Published online: 11 June 2011

Abstract During the last decade, there has been an increasing interest in the design of very fast wavelet image encoders focused on specific applications like interactive real-time image and video systems, running on power-constrained devices such as digital cameras, mobile phones where coding delay and/or available computing resources (working memory and power processing) are critical for proper operation. In order to reduce complexity, most of these fast wavelet image encoders are non-(SNR)-embedded and as a consequence, precise rate control is not supported. In this work, we propose some simple rate control algorithms for these kind of encoders and we analyze their impact to determine if, despite their inclusion, the global encoder is still competitive with respect to popular embedded encoders like SPIHT and JPEG2000. In this study we focus on the non-embedded LTW encoder, showing that the increase in complexity due to the rate control algorithm inclusion, maintains LTW competitive with respect to SPIHT and JPEG2000 in terms of R/D performance, coding delay and memory consumption.

Keywords Wavelet image coding · Fast image coding · Rate control · Non-embedded coding

O. M. López Granado (✉) · M. O. Martínez-Rach · P. Piñol Peral · M. P. Malumbres
Miguel Hernández University,
Avda. Universidad s/n, 03205 Elche, Spain
e-mail: otoniel@umh.es

M. O. Martínez-Rach
e-mail: mmrach@umh.es

P. Piñol Peral
e-mail: pablop@umh.es

M. P. Malumbres
e-mail: mels@umh.es

J. Oliver Gil
Universidad Politécnica de Valencia,
Camino de Vera, s/n, 46222 Valencia, Spain
e-mail: joliver@disca.upv.es

1 Introduction

In the last years, the coding efficiency of wavelet-based image encoders have been improved, achieving in this way, a reduction in the bandwidth or amount of memory needed to transmit or store a compressed image. Unfortunately, many of these coding optimizations involve higher complexity, requiring faster and more expensive processors. For example, the JPEG 2000 [9] standard uses a large number of contexts with an iterative time-consuming optimization algorithm (called PCRD) to improve coding efficiency. Other encoders (like the one proposed in [18]) achieve very good coding efficiency with the introduction of high-order context modeling, being the model formation a slow process. Even bit-plane coding employed in many encoders (like [16] and [3]) results in a slow coding process since an image is scanned several times, focusing on a different bit-plane in each pass, which in addition causes a high memory access overhead (cache miss rate increase).

The above mentioned encoders are designed to obtain high performance in rate-distortion terms and also a broader functionality, but unfortunately other parameters like complexity or memory resources are not considered as critical as the former ones.

Recently, several authors have shown interest in developing very fast and simple wavelet encoders that are able to get reasonable good R/D performance with reduced computing resources requirements. The objective of these fast and efficient image encoders is mainly targeted to interactive real-time applications running under resource constrained devices. In that scenario, the data must be encoded as soon as possible to fit the application constraints using the scarce available resources in the system (memory and processing power).

Basically, these encoders do not present any type of iterative method, and each coefficient is encoded as soon as it is visited. However, this results in the loss of SNR scalability and precise rate control capabilities (in other words, the image cannot be compressed to a specific user-defined file size). They simply apply a constant quantization to all wavelet coefficients, encoding the image at a constant and uniform quality, as it happened in the former JPEG standard [8], where only a quality parameter was available and no precise rate control was performed.

In [15] the first non-embedded encoder was proposed with the aim of reducing the complexity of a wavelet-based image encoder. This algorithm is a modified version of SPIHT [16], in which once a coefficient is found to be significant, all significant bits are encoded to avoid refinement passes (losing SNR-scalability and rate control, both available in original SPIHT). However, in this proposal, bit-plane processing is still needed in the sorting passes, and thereby the coding process is not speeded up too much.

One of the first fast non-embedded image encoders was LTW [13], a tree-based wavelet encoder that avoids bit-plane processing and predictive encoding techniques; instead of that, it uses a one-pass coefficient coding process with a very reduced number of contexts for arithmetic encoding.

In [2] it has been proposed another very fast non-embedded encoder called PROGRESS. It follows the same ideas of [13], avoiding bit-plane coding and using coefficient trees to encode wavelet coefficients in only one pass. In this encoder, all the coefficients and not only the zero coefficients, are arranged in trees. The number of bits needed to encode the highest coefficient in each tree is computed, and all the coefficients at the current subband level are binary encoded with that number of bits. Then, the following subband level is encoded (in decreasing order) simply by computing again the number of bits needed to represent each subtree at that level and using that number of bits again.

Another fast non-embedded image encoder is the BCWT encoder [7]. It offers high coding speed, low memory usage and good R/D performance. The key of BCWT encoder is its one-pass backward coding, which starts from the lowest level sub-bands and travels backwards. Map of Maximum Quantization Levels of Descendants (MQD Map) calculation and coefficient encoding are all carefully integrated inside this pass in such a way that there is as little redundancy as possible for computation and memory usage.

None of the above non-embedded encoders support rate control so, in this paper, we propose several rate control algorithms for them. We have chosen the LTW encoder to evaluate the rate control algorithms not only in terms of rate/distortion (R/D) performance but also in terms of coding delay and overall memory usage.

Our first rate control proposal extracts some features from the wavelet transformed image and finds correlations with the quantization parameter for a specific target bit-rate. The second proposal is based on a simple model of the encoding engine in a similar way than in [6] and [11] where statistical models were employed to accomplish rate control. To set the finer scalar uniform quantization parameter (Q), we model the bit-rate evolution with a second order polynomial function. Finally, to cope with the rate accuracy demands of certain applications, we propose an iterative version for bounding the estimation error with the minimum number of iterations.

The rest of the paper is organized as follows. In Section 2, the LTW algorithm is outlined. In Section 3, we describe the proposed rate control algorithms and evaluate their accuracy in Section 4. Then, in Section 5, we show the results of the global encoder system (including rate control) and compare it with SPIHT and JPEG 2000 in R/D performance, complexity and memory requirements. For further evaluation, we have compared its performance with a fully optimized version of JPEG2000 (Kakadu). Finally, in Section 6 some conclusions are drawn.

2 LTW: A Fast Non-embedded Image 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 [1], the wavelet coefficients are first quantized using two quantization parameters (*rplanes* and Q) and then encoded with arithmetic coding.

For the coding stage, if the absolute value of a coefficient and all its descendants (considering the classic quad-tree structure from [16]) is lower than a threshold value (2^{rplanes}), the entire tree is encoded with a single symbol, which we call LOWER symbol. 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, as well as a formal description can be found in [14].

In this work, the rate control will be achieved by properly tuning the quantization parameters. In particular the *rplanes* parameter for a coarser quantization (in which *rplanes* less significant bits are removed from each coefficient), and the *Q* parameter for a finer adjustment in a typical uniform scalar quantization process.

3 Rate Control Support for Non-embedded Encoders

In this section, we propose several lightweight rate control algorithms for non-embedded encoding, with increasing complexity and accuracy. These algorithms will predict the proper quantization values that lead to a final bit-rate close to the target one. Although these rate control algorithms could be applied to whatever non-embedded wavelet encoder, we will use the LTW encoder in the evaluation to observe their behavior in the overall encoding system.

3.1 Zero-order Entropy Based Rate Control Algorithm

This method is based on the zero-order entropy (Eq. 1) of the wavelet coefficients. The estimation of the quantization parameters is based on the correlation between entropy, target bit-rate and quantization parameters.

$$H(x) = -\sum_{x} p(x)\log_2(p(x)) \tag{1}$$

We use the Kodak image set [4] as a representative set of natural images for our purposes and the LTW encoder with both *Q* and *rplanes* quantization parameters. As there is a correlation between the wavelet coefficients entropy and the quantization parameters, we can establish a relationship between them for a given target bit-rate by means of curve and surface fitting techniques [5, 12, 17, 19]. In particular, surface fitting process was driven by polynomial bivariate (bit-rate and entropy) equations due to its low computational complexity. So, Eqs. 2, 3 and 4 represent the surface fitting expressions corresponding to the fine quantizer estimation ($Q_{\text{rplanes}}(x, y)$) for *rplanes* values of 2, 3 and 4 respectively. The variables 'x' and 'y' represent the wavelet coefficients entropy and the target bit-rate respectively, and constant values a, b, c, d, e, f, g, h, i and j have been computed through the aforementioned surface fitting methods using the wavelet coefficient entropy information extracted from the Kodak image set. For each equation, we also show the Coefficient of Determination (r^2) that measures the fitting goodness (ideally $r^2 = 1$).

$$\begin{aligned}
Q_{rp2}(x, y) &= a + bx + c/y + dx^2 + e/y^2 + fx/y \\
&\quad + gx^3 + h/y^3 + ix/y^2 + jx^2/y \\
a &= 23.99,\ b = -23.68,\ c = -1.27,\ d = 7.36, \\
e &= 0.06,\ f = 1.10,\ g = -0.72,\ h = 0.003, \\
i &= -0.06,\ j = -0.009 \\
&\qquad (r^2) = 0.949
\end{aligned} \tag{2}$$

$$\begin{aligned}
Q_{rp3}(x, y) &= a + b/x + c\ln y + d/x^2 + e(\ln y)^2 \\
&\quad + f(\ln y)/x + g/x^3 + h(\ln y)^3 \\
&\quad + i(\ln y)^2/x + j(\ln y)/x^2 \\
a &= 13.04,\ b = -69.08,\ c = -5.98,\ d = 129.67, \\
e &= 0.58,\ f = 19.07,\ g = -82.79,\ h = -0.07, \\
i &= -0.66,\ j = -16.31 \\
&\qquad (r^2) = 0.950
\end{aligned} \tag{3}$$

$$\begin{aligned}
Q_{rp4}(x, y) &= a + b/x + c\ln y + d/x^2 + e(\ln y)^2 \\
&\quad + f(\ln y)/x + g/x^3 + h(\ln y)^3 \\
&\quad + i(\ln y)^2/x + j(\ln y)/x^2 \\
a &= 5.29,\ b = -19.55,\ c = -2.29,\ d = 25.77, \\
e &= 0.15,\ f = 4.74,\ g = -11.72,\ h = -0.02, \\
i &= -0.02,\ j = -2.54 \\
&\qquad (r^2) = 0.968
\end{aligned} \tag{4}$$

The Entropy-based algorithm shown in Fig. 1 works as follows:

- First (E1), the *rplanes* value is determined as a function of the target bit-rate (T_{bpp}). We have experimentally determined the proper *rplanes* value for the working bit-rate ranges. Particularly, the best choice for a target bit-rate in the range 0.0625–0.5 bpp is *rplanes* = 4, *rplanes* = 3 in the range 0.5–1.5 bpp and *rplanes* = 2 in the range 1.5–2 bpp
- Second (E2), we apply the coarser quantization by removing the previously computed *rplanes* less significant bits from all wavelet coefficients.

```
Input: Wavelet Coefficients ($C_{i,j}$),
Target bit-rate ($Tbpp$),
Surface Fitting Equations for each rplanes value ($Eq_{rp}$)
Output: $Q$, rplanes
  (E1) Determine rplanes using $T_{bpp}$
  (E2) Remove the rplanes less significant bits to all
       wavelet coefficients ($C_{i,j}$)
  (E3) Calculate Wavelet Coefficients Entropy, $Se$
  (E4) Determine quantization parameter ($Q$)
       $Q = Q_{rplanes}(S_e, T_{bpp})$
```

Figure 1 Entropy-based algorithm.

- Third (E3), the zero-order entropy of the coarse quantized wavelet coefficients (S_e) is calculated.
- Finally (E4), the finer quantization parameter Q is obtained from the surface fitting equation corresponding to the selected *rplanes* value: $Q = Q_{\text{rplanes}}(S_e, T_{bpp})$.

3.2 Rate Control Based on a Trivial Coding Model

It is difficult to estimate the quantization parameters at a certain degree of accuracy for a particular target bit-rate by only using the zero order entropy of the wavelet coefficients. So, we decided to study how the encoder works in order to define a simplified statistical model of the encoding engine in a similar way as in [6] and [11]. In [6] authors propose an expensive rate allocation scheme based on the Lagrangian optimization problem that offers a low accurate rate control capability. In [11] another statistical model based in the generalized-Gaussian densities (GGD) approach is proposed, obtaining an expensive but high accurate rate control behavior.

In this work we will use the LTW coding engine in order to define a simple model that will be able to supply a fast and accurate estimation of the resulting bit-rate. The encoding model uses a two-tier quantization process based on (1) a coarse quantization where the *rplanes* least signifficant bits of all wavelet coefficients are removed and (2) a finer quantization applied to all wavelet coefficients through a scalar uniform quantization (Q). So, the proposed model of the encoding system will be defined in two parts:

Firstly, we need to determine the appropiate *rplanes* value for a given target bit-rate. For each image in the Kodak set, we perform the DWT and apply the coarse quantization for each specific *rplanes* value (from 2 to 7). Then, for each *rplanes* value, the coarse quantized DWT coefficients are classified to build the symbol map. The symbol map will be compossed by (1) non-significant symbols (zero coefficients) and (2) significant symbols with magnitude n (wavelet coefficients that require n bits to represent their value). Taking into account this representation, we compute the probability distribution function (pdf) of the LTW symbol map.

After that, we will obtain an estimation of the bit-rate required to encode the symbol map by means of its zero-order entropy (S_e).

Since we also know the number of significant wavelet coefficients and the number of bits needed for coding their value and sign, we can calculate the exact number of bits sent to the output bitstream, as they are raw binary encoded ($Bits_{\text{total}}$).

So, the final bit-rate estimation for each *rplane* value ($E_{bpp}(rplanes)$) is obtained by adding the arithmetic encoder estimation of the symbol map (S_e) to the raw encoding bit count of significant coefficients ($Bits_{\text{total}}$) (Eq. 5)

$$E_{bpp}(rplanes) = S_e(rplanes) + Bits_{\text{total}}(rplanes) \tag{5}$$

The resulting estimation gives a biased measure of the real bit-rate for all operative bit-rate range (from 0.0625 to 2 bpp) considered in this work. The model bit-rate estimation (E_{bpp}) uses a zero-order entropy model. However, LTW encoding scheme uses an adaptive arithmetic encoder with context modeling. This difference produces an error between the estimated bit-rate (E_{bpp}) and the target bit-rate as shown in Fig. 3. We observed that the bit-rate estimation depends on the symbol map entropy (S_e), so the lower the entropy the lower the estimation error. This leads us to reduce the estimation error by means of an adjustment function which will be defined from the entire Kodak image set.

In Fig. 2 we show the error adjustment function for $rplanes = 4$ (no finer quantization Q is considered at

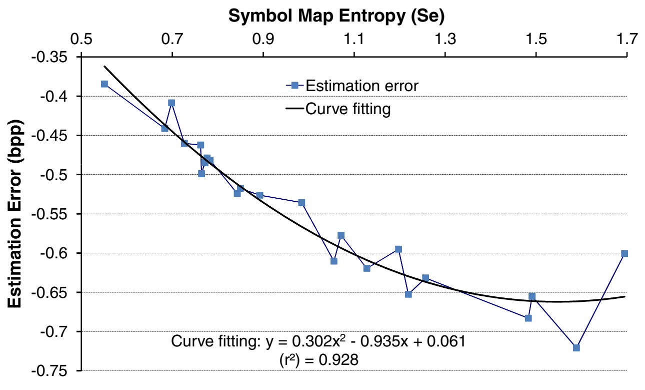


Figure 2 (Model-based)—Estimation error as a function of symbol map entropy (S_e) from the entire Kodak image set for $rplanes = 4$.

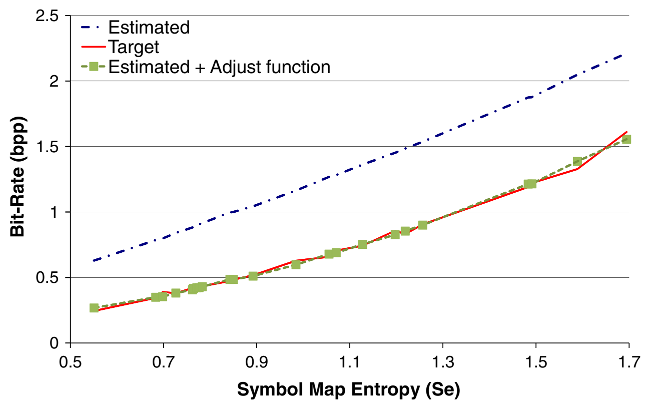


Figure 3 (Model-based)—Estimated vs Real bit per pixel for the entire Kodak image set ($rplanes = 4$).

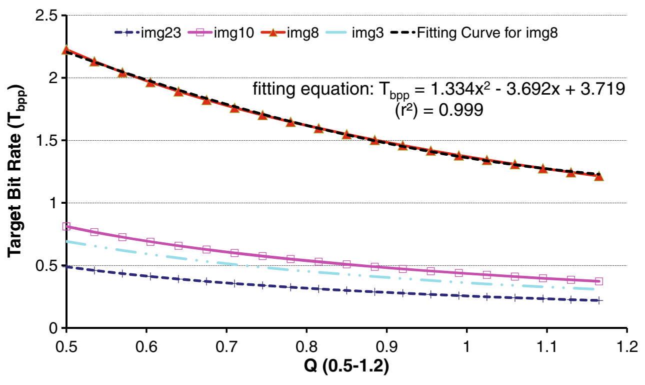


Figure 4 (Model-based)—Bit-rate progression of four images from the Kodak set from *rplanes* 3 to *rplanes* 4.

the moment). For each image of the Kodak set we measure the real estimation error, so by using curve fitting we define the adjustment error as a function of the symbol map entropy (S_e). In other words, this curve will determine the model estimation error for $rplanes = 4$. So, for each *rplanes* value we have obtained the adjustment function ($\Delta\,(rplanes, S_e)$) that we will apply to reduce the model estimation error. So, the bitrate estimation expresion will be rewrited as:

$$E_{bpp}(rplanes) = S_e(rplanes) + Bits_{\text{total}}(rplanes) + \Delta\,(rplanes, S_e(rplanes)) \qquad (6)$$

In Fig. 3 we show the estimated and target bit-rates resulting from encoding the whole Kodak image set with a *rplane* value of 4. As it can be seen, the estimation error is significantly reduced after applying the corresponding adjustment function.

After that, the target bit-rate (T_{bpp}) will determine the proper value of *rplanes* by choosing *rplanes* so that $E_{bpp}(rplanes) \geq T_{bpp} > E_{bpp}(rplanes + 1)$.

Once the *rplanes* value is determined, we have to estimate the scalar uniform quantization value (Q) that will produce a bit-rate as close as possible to the target bit-rate. For this purpose, we observed that the bit-rate progression from *rplane* value to $rplane + 1$ value, follows a second order polynomial curve ($y = A * x^2 + B * x + C$) that shares near the same x-value of the vertex ($K_{\min} = \frac{-B}{(2*A)}$) for all images in the Kodak set (see Fig. 4). Since we know three points of that quadratic polynomial curve $E_{bpp}(rplanes)$, $E_{bpp}(rplanes + 1)$ and the curve vertex ($K_{\min}$), we can build the corresponding expression that will supply the estimated value of Q for a given target bit-rate.

As mentioned, the $K_{\min}$ value is not exactly the same for all images in the Kodak set, so we have estimated this value taking into account the symbol map entropy (S_e) of all images in the Kodak set, in a similar way to that in the error adjustment function. We could have used the mean $K_{\min}$ value from the Kodak set, but in order to obtain a more accurate estimated value of $K_{\min}$ we use a curve fitting method instead. In Fig. 5 we show the x-value of $K_{\min}$ progression as a function of the symbol map entropy, S_e, of all images in the Kodak set for $rplanes = 2$. We have obtained a fitting equation for each *rplanes* value (from 2 to 7).

The whole algorithm, shown in Fig. 6, works as follows:

- First (E1), we estimate the resulting bit-rate after applying only the coarser *rplanes* quantization to wavelet coefficients for *rplanes* values from 2 to 7 ($E_{bpp}(rplanes)$).
- Second (E2), we apply the corresponding error adjustment functions to these estimations.
- Third (E3), we set the appropriate *rplanes* value for the requested target bit-rate (T_{bpp}). ($E_{bpp}(rplanes) \geq T_{bpp} > E_{bpp}(rplanes + 1)$).

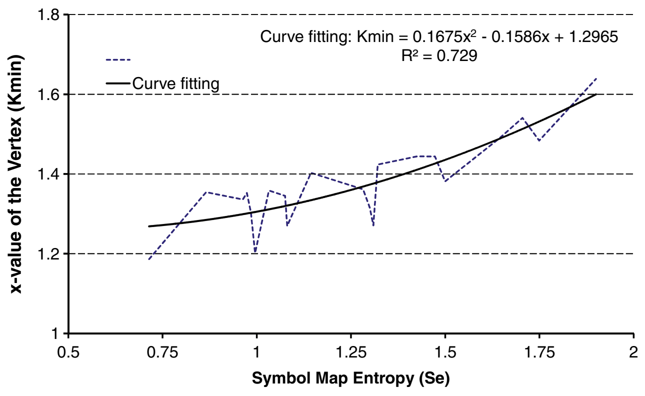


Figure 5 (Model-based)—*Kmin* as a function of symbol map entropy from the entire image Kodak set for $rplanes = 2$.

```
Input: Wavelet Coefficients ($C_{i,j}$),
Target bit-rate ($T_{bpp}$)
Output: $Q$, $rplanes$
  (E1) for each $rplanes$ in [2..7]
      for each $C_{i,j}$ coefficient
          $nbits_{i,j} = \lceil \log(|C_{i,j}|) \rceil$
          if $nbits_{i,j} > rplanes$
              $Symbol_{(nbits_{i,j}-rplanes)} += 1$
              $Bits_{total} += nbits_{i,j} - rplanes$
          else
              $Symbol_{non-significant} += 1$
      Calculate the Symbol Map Entropy, $S_e$
      $E_{bpp} = (Bits_{total}/sizeof(image)) + S_e$
  (E2) for each $rplanes$ in [2..7]
      Apply_Adjust_Function:
      $E_{bpp} = E_{bpp} + \Delta(rplanes, S_e)$
  (E3) Determine $rplanes$
      $E_{bpp}(rplanes) \geq T_{bpp} > E_{bpp}(rplanes + 1)$
  (E4) Compute $K_{min}$ value with $S_e$ and $rplanes$
  (E5) Determine quantization parameter ($Q$)
      Obtain A,B,C using $E_{bpp}(rplanes)$,
      $E_{bpp}(rplanes + 1)$ and $K_{min}$ values and solve
      $T_{bpp} = A.Q^2 + B.Q + C$
```

Figure 6 Model-based algorithm.

- Next (E4), we compute the $K_{\min}$ value using the symbol map entropy (S_e) and *rplanes*.
- Then (E5), we obtain the quadratic expression for determining the value of Q by using the Newton interpolation algorithm.
- Finally, we solve the expression, so we obtain the estimated Q value.

3.3 Lightweight Iterative Rate Control

With the Model-based rate control algorithm described in the previous subsection we can define an iterative version to reduce the estimation error with a moderate computational complexity increase. Thus, depending on the application requirements, we can get the proper trade-off between both rate control factors: complexity and accuracy. Now, we can define the maximum allowed estimation error (MAE) as a relative or absolute error and the algorithm will perform coding 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 bit-rate by using the algorithm described in the previous subsection (see Fig. 6). Then the source image will be coded with the quantization parameters found. If the resulting bit-rate error is lower than the maximum allowed error

```
Input: Wavelet Coefficients ($C_{i,j}$), Target bitrate ($T_{bpp}$)
and Maximum Allowed Error (MAE)
Output: $Q$, $rplanes$
  (E1) Obtain rplanes and Q using algorithm from
Fig. 6
  (E2) $Real_{bpp} = Encode(rplanes, Q)$
       $error = T_{bpp} - Real_{bpp}$
  (E3) Iterative stage
      i=0
      while (($error > MAE$) and
              (i < MAXITERATIONS))
          $NewT_{bpp} = T_{bpp}$
          if i in [1..3]
              $NewT_{bpp}$ += error
          else
              Q=Newton($Points_{(Last\_three)}$)
          $Real_{bpp} = Encode(rplanes, Q)$
          $error = T_{bpp} - Real_{bpp}$
          Points[i]=(Q,$Real_{bpp}$)
```

Figure 7 Iterative algorithm.

(MAE), then the algorithm finishes, otherwise, it performs a new Q estimation based on the observed error. This is done during the first three iterations obtaining pair values of Q and real bit-rate. In the following iterations, if needed, the new three real points obtained are used to compute a new quadratic polynomial curve for Q by means of the Newton interpolation algorithm which will be more accurate than the previous one (see algorithm in Fig. 7).

4 Rate Control Evaluation

Using a C++ implementation of the LTW encoder, the different proposals were developed and tested on an Intel PentiumM 1.6 Ghz Processor. To determine the curve fitting and error adjustments in the first two methods, we have used the Kodak image set as a representative set of natural images. We restrict our proposals to work in the range from 0.0625 to 1 bpp. Finally, we used Lena (512 × 512), Barbara (512 × 512), Goldhill (512 × 512) and Peppers (512 × 512) test images (outside the Kodak set) to validate the proposed methods.

In Fig. 8 we can see that, for all bit-rates in the range 0.625 to 1 bpp, the Model-based algorithm gets the best results. Although for several images in the Kodak set the Entropy-based algorithm performs better, the maximum error peaks in the Model-based algorithm are significantly lower than in the Entropy-based one.

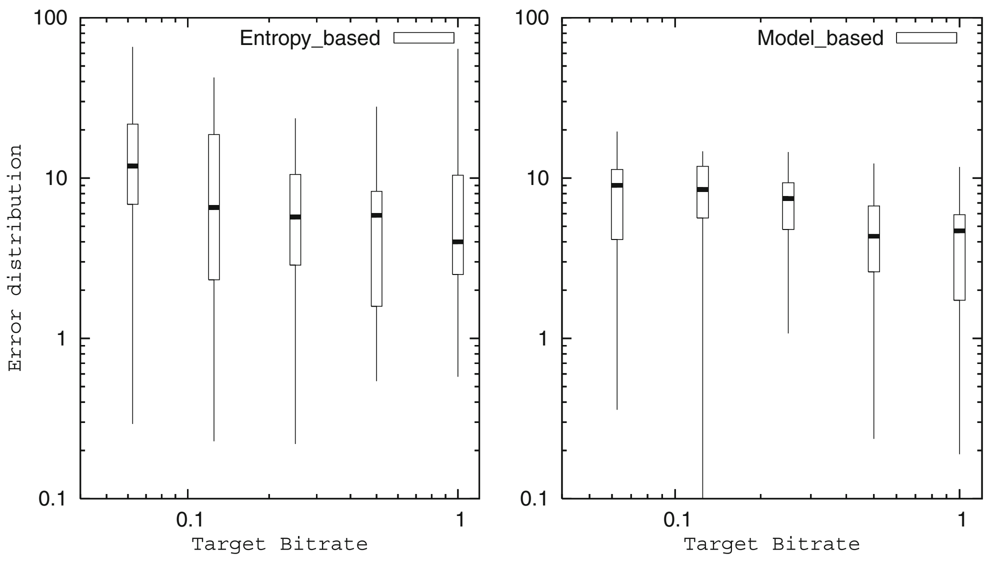


Figure 8 Entropy-based vs Model-based % error prediction for the entire Kodak set. *Vertical lines* indicate maximum and minimum error values, while boxes indicate quartiles q1-q3, and q2 (*median*) corresponds with *horizontal lines* inside the boxes.

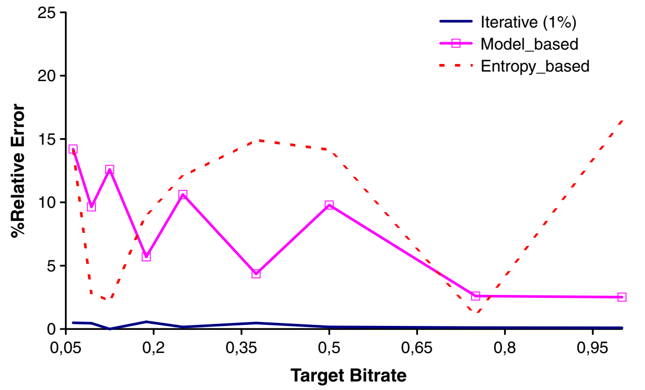
The Model-based algorithm yields a lower error on the estimation process over the Kodak image set than the Entropy-based algorithm. Furthermore, the choice of the *rplanes* parameter has been included in the estimation process. Table 1 shows the average estimation error of both the Entropy-based and the Model-based algorithm at different target bit-rates. The Model-based relative error produced is around 5% on average at low compression rates and it grows up to 9% at higher compression rates. This is due to the high slope of the quadratic expression used to obtain Q (see Fig. 4) when the *rplanes* parameter grows, so, a slight change over Q parameter implies a high variation on the final bit-rate and as a consequence, a higher estimation error.

In Fig. 9, we show the bit-rate accuracy of the proposed rate control methods for Lena and Goldhill test images. Although not shown, the behavior is very similar in the other non-Kodak test images. In general, the Model-based method does not work efficiently at very low bit-rates in the range from 0.0625–0.125 bpp. This behavior is due to the model simplicity where there is no symbol differentiation in the insignificant coefficients set. In particular the roots (*LOWER* symbols) and members of lower trees, which are very common symbols at these compression rates, are not handled separately. However, at moderate and low compression rates, the Model-based proposal is more accurate than the Entropy-based one.

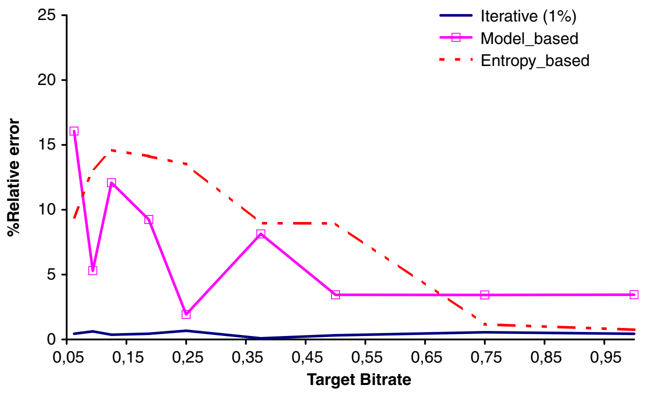
In Fig. 10, we measured the computational cost (in CPU cycles) of the proposed methods when coding the Goldhill test image. As it can be seen, the Model-based

Table 1 Average % relative estimation error.

	Entropy-based	Model-based
1 bpp	15.35	4.46
0.5 bpp	16.76	5.11
0.25 bpp	14.25	7.48
0.125 bpp	44.64	8.50



(a) Lena



(b) GoldHill

Figure 9 Bit-rate accuracy for **a** Lena and **b** GoldHill images.

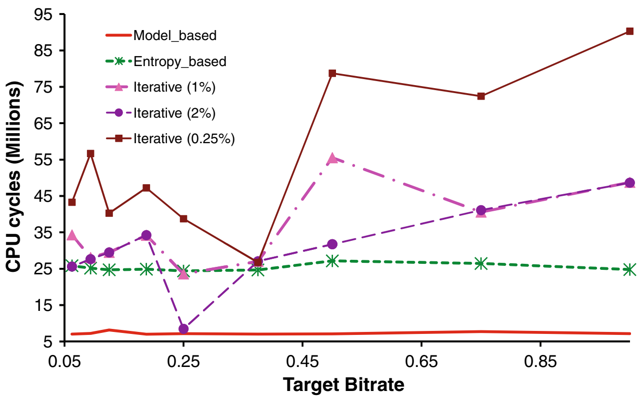


Figure 10 Complexity evaluation of different proposals for GoldHill image. x% in the iterative version indicates the maximum percent error allowed.

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 the Model-based one, mainly due to the higher complexity of float type computations. We want to remark that, in the Entropy-based proposal the zero-order entropy of all wavelet coefficients must be computed while in the Model-based we only need to compute the symbol map entropy, which is in fact a reduced number of symbols. In the case of iterative versions, we can observe that with a maximum 2% relative error, we obtain a very fast rate control estimator (sometimes faster than the entropy based). Also, we can state that the computational cost is not dependent on the target bit-rate, although in the iterative versions, the number of iterations may produce some deviations.

5 Global Performance Evaluation

In order to analyze the impact of rate control proposals when applied to LTW encoder, we have performed several experiments comparing the obtained results with the original encoder. In addition to R/D performance, we also analyze other performance metrics like coding delay and memory consumption.

So as to perform a fair evaluation, we have chosen SPIHT (original version), JPEG2000 (Jasper 1.701.0) and LTW version 1.1, since their source code is available for testing. The correspondent binaries were obtained by means of Visual C++ (version 6.0) compiler with the same project options. The test images used in the evaluation were: Lena (512 × 512), Barbara (512 × 512), GoldHill (512 × 512), Cafe (2560 × 2048) and Woman (2560 × 2048).

Table 2 shows the coding delay for all the encoders under evaluation. LTW_RC is the Model-based rate control version of LTW (described in Section 3.2). LTW_RCi is the iterative rate control version of LTW (described in Section 3.3) with a relative (%value) and an absolute (ABS suffix) rate control maximum allowed error that we experimentally have fixed to $\pm 0.04\, bpps$. We discard the first rate control method (entropy-based) due to its lower accuracy with respect to the Model-based one. As expected, JPEG2000 is the

Table 2 Comparison of coding and decoding delays, excluding DWT (time in million of CPU cycles).

Codec/ bit-rate	SPIHT	JPEG2000	LTW orig.	LTW-RC	LTW-RCi 2%	LTW RCi-ABS
CODING Lena (512×512)						
0.125	20.82	158.79	9.75	**8.316**	20.25	10.03
0.25	29.12	161.92	14.09	**11.726**	27.08	13.42
0.5	45.82	167.14	22.46	**18.356**	62.07	40.74
1	79.56	175.64	41.46	**36.656**	75.75	38.61
DECODING Lena (512×512)						
0.125	11.3	11.46	8.28	**6.77**	6.77	6.77
0.25	19.38	18.11	13.39	**10.8**	10.8	10.8
0.5	34.9	30.78	23.48	**18.84**	18.84	18.84
1	66.8	50.99	46.52	**39.64**	39.64	39.64
CODING Cafe (2560×2048)						
0.125	469.73	4546.51	210.41	**192.06**	265.49	265.81
0.25	687.67	4527.09	299.78	**255.80**	670.28	327.44
0.5	1128.43	4591.08	459.48	**392.74**	947.77	950.7
1	2017.8	4736.99	733.21	**630.70**	1444.62	1443.28
DECODING Cafe (2560×2048)						
0.125	232.53	234.10	182.97	**160.88**	160.88	160.88
0.25	397.98	362.96	295.56	**252.02**	252.02	252.02
0.5	745.69	593.92	491.96	**431.59**	431.59	431.59
1	1453.89	1040.39	830.87	**738.91**	738.91	738.91

Table 3 PSNR (dB) with different bit-rate and coders.

Codec/ bit-rate	SPIHT	JPEG 2000	LTW orig.	LTW RC	LTW-RCi 2%	LTW-RCi ABS	Kakadu 5.2.5
Lena (512×512)							
0.125	31.10	30.81	**31.28**	30.59(−0.016)	31.06(−0.001)	30.59(−0.016)	30.95
0.25	34.15	34.05	**34.33**	33.65(−0.027)	34.05(−0.004)	33.65(−0.026)	34.11
0.5	37.25	37.26	**37.39**	36.76(−0.049)	37.15(−0.008)	37.08(−0.011)	37.30
1	40.46	40.38	**40.55**	40.34(+0.035)	40.20(−0.001)	40.34(+0.035)	40.40
Cafe (2560×2048)							
0.125	20.67	20.74	20.76	20.63(−0.001)	20.63(−0.001)	20.63(−0.001)	**20.78**
0.25	23.03	23.12	**23.24**	22.60(−0.026)	23.08(+0.002)	22.60(−0.027)	23.15
0.5	26.49	26.79	**26.85**	26.04(−0.046)	26.53(−0.006)	26.53(−0.007)	26.84
1	31.74	**32.03**	32.02	30.89(−0.097)	31.64(−0.010)	31.64(−0.014)	**32.03**
Barbara (512×512)							
0.125	24.86	**25.25**	25.21	24.30(−0.026)	25.04(+0.002)	24.21(−0.027)	25.24
0.25	27.58	28.33	28.04	27.09(−0.036)	27.76(−0.001)	27.09(−0.036)	**28.36**
0.5	31.39	32.14	31.72	30.80(−0.055)	31.47(−0.002)	31.34(−0.012)	**32.17**
1	36.41	37.11	36.67	35.61(−0.101)	36.39(−0.004)	36.25(−0.021)	**37.15**
Woman (2560×2048)							
0.125	27.33	27.33	**27.51**	27.19(−0.007)	27.30(−0.003)	27.19(−0.007)	27.36
0.25	29.95	29.98	**30.15**	29.45(−0.028)	30.02(+0.004)	29.45(−0.027)	30.05
0.5	33.59	33.62	**33.82**	32.94(−0.050)	33.55(−0.002)	33.55(−0.001)	33.64
1	38.27	38.43	**38.52**	37.52(−0.098)	38.32(−0.003)	38.32(−0.002)	38.43

slowest encoder and the original LTW is one of the fastest encoders. As shown in Table 2, the LTW_RC version does not introduce a great overhead and it has an acceptable accuracy. If this rate control algorithm precision is not enough for the application, LTW_RCi is the candidate at the expense of an increasing complexity. In general, all the rate control versions of LTW are faster than SPIHT, specially the non-iterative version, LTW_RC, that performs the encoding process twice as fast as SPIHT.

Although it could be thought that original LTW should be faster than the LTW_RC version due to rate control overhead, the results show just the opposite. The reason about this behavior is based on the differences between the quantization processes of both encoders. An image is encoded with the original LTW encoder using a fixed value of *rplanes* parameter ($rp = 2$) and moving the Q parameter through a wide range ($[0.5 - \infty[$). However, the LTW_RC version uses the estimated value for *rplanes* parameter (from 2 to 7) limiting the value of the Q parameter to a shorter range ($[0.5 - 1.2]$). So, as more non-significant symbols are produced before applying the finer quantization parameter (Q) with this method, the algorithm becomes faster because more multiplication operations are avoided. However, as a 'side effect', coding efficiency decreases slightly, as we will see later.

In the iterative rate control versions, we have found two ways of defining the maximum allowed error: a relative or an absolute MAE (Maximum Allowed Error). The relative maximum error shows a non linear behavior, since rate control precision of 1% is not the same at 2 bpp than at 0.125 bpp. For very low bit-rates, achieving an accuracy of 1% has no effects to R/D performance. The maximum absolute error is fixed independently of the target bit-rate, so it produces different relative errors at different bit-rates. It is important to take into account that proposed rate control methods have an average precission error around 5% at 1 bpp and 9% at 0.125 bpp, as previously shown.

Table 3 shows the R/D evaluation of the proposed encoders. In general, the original LTW encoder obtains very good performance results, especially in Lena and Woman test images. The iterative rate control versions of LTW have slightly lower PSNR performance than SPIHT and JPEG2000, being the LTW_RCi at 2% the one that better R/D behavior shows. Table 3 also shows the absolute bit-rate error in brackets for all LTW rate control versions. The lower performance of the rate control algorithm versions is mainly due to the

Table 4 Memory Requirements for evaluated encoders (KB).

Codec/ image	SPIHT	JPEG2000	LTW orig.	LTW-RC	LTW-RCi
Lena	3228	4148	**2048**	2092	3140
Cafe	46776	65832	**21576**	21632	42188

achieved final bit-rate that is always lower than the target one (obviously, the more accuracy, the better R/D performance).

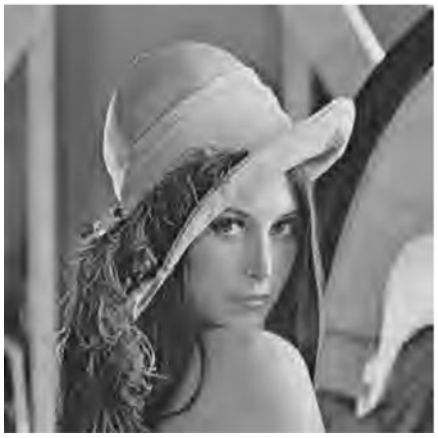
In Table 4, memory requirements of the encoders under test are shown. The original LTW needs only the amount of memory to store the source image.

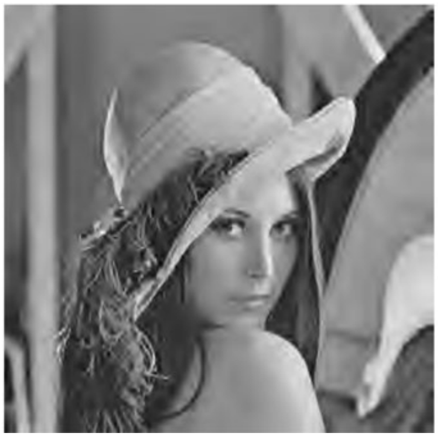
(a) 31.06

(b) 31.10

(c) 30.81

(d) Original


(e) 28.35


(f) 28.38

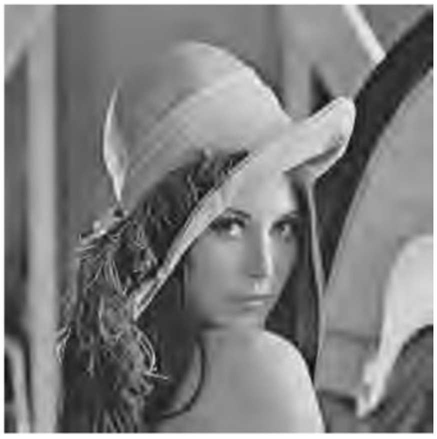

(g) 27.84

Figure 11 Lena compressed at 0.125 bpp **a** LTW_RCi_2%, **b** SPIHT, **c** JPEG2000 and Lena compressed at 0.0625 bpp **e** LTW_RCi_2%, **f** SPIHT, **g** JPEG2000.

LTW_RC requires also 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 LTW_RCi version requires twice the memory space than LTW and LTW_RC, since at each iteration the original wavelet coefficients must be restored to avoid a new DWT time-consuming procedure. SPIHT requires near the same memory than LTW_RCi, and JPEG2000 needs three times the memory of LTW.

Figure 11 shows Lena test image (512×512) compressed at 0.125 bpp and 0.0625 bpp with (a, f) LTW_RCi, (b, g) SPIHT and (c, h) JPEG2000. Although SPIHT encoder is in terms of PSNR slightly better than LTW_RCi and JPEG2000, subjective test does not show perceptible differences between reconstructed versions of Lena image. At 0.0625 bpp the difference in PSNR between LTW_RCi or SPIHT and JASPER is near 0.5 dB, but this difference is only visible if we carry out a zoom over the eyes zone as it can be seeing in Fig. 12. Both SPIHT and LTW_RCi have a similar behavior.

5.1 Optimized Encoders

All LTW versions were developed finding the optimizations for maximizing R/D performance, so its software code is not optimized, just like JPEG2000 reference software. However, we have compared its performance with respect to a full optimized implementation of the JPEG 2000 algorithm: Kakadu [10], in order to evaluate whether a full optimization of LTW is worth the effort. For that purpose, we have used the 5.2.5 version, one of the fully optimized Kakadu versions which includes multi-thread and multi-core hardware support, processor intrinsics like MMX/SSE/SSE2/SIMD and fast multicomponent transforms.

As shown in Fig. 13, LTW_RC is a very fast encoder even though not being fully optimized. The speed of LTW_RC lies on the simple engine coding model.

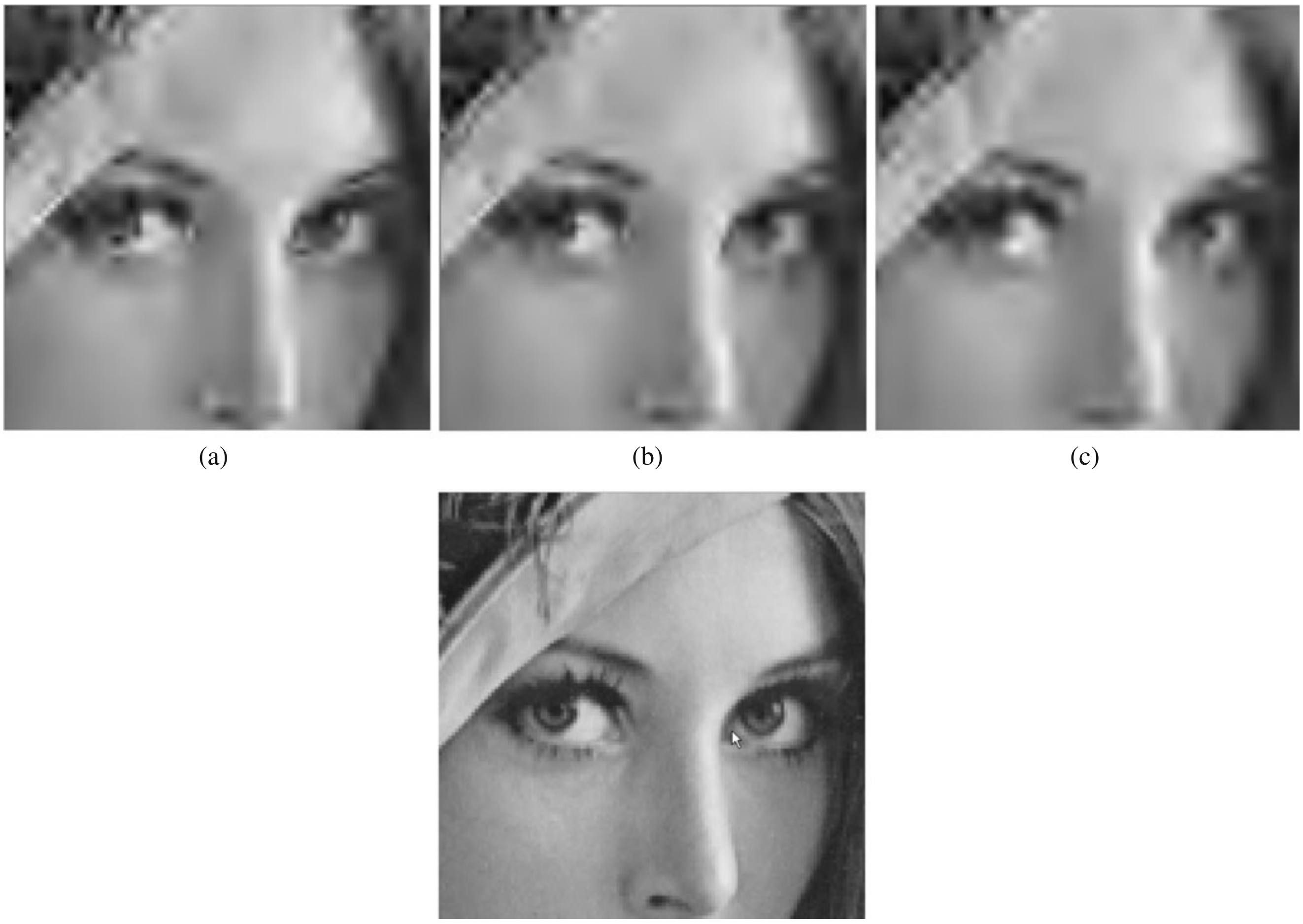


Figure 12 Zoom over eyes zone in reconstructed Lena at 0.0625 bpp - **a** LTW_RCi_2%, **b** SPIHT, **c** JPEG2000.

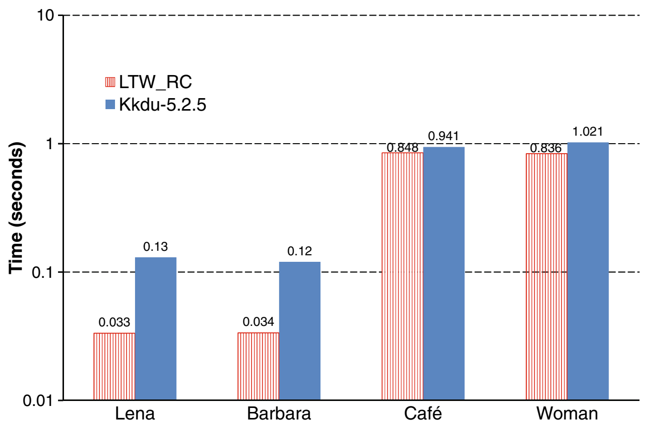


Figure 13 Execution time comparison (*end-to-end*) of the coding process at 0.5 bpp.

LTW_RC is on average 1.2 times as fast as Kakadu-5.2.5 for images like Cafe or Woman. For smaller images like Lena or Barbara, LTW_RC is on average 3.2 times as fast as Kakadu-5.2.5.

Regarding to memory requirements, LTW_RC needs only the amount of memory to store the source image and 1.2 KB to store the model histogram as mentioned before, while Kakadu memory requirements are independent of the image size due to its DWT block-based implementation and they are on average 1660 KB.

In terms of R/D, there are slight differences between both codecs as Table 3 shows. For images with lots of high frequency components, like Barbara, Kakadu provides a better PSNR than LTW, but for images like Lena or Woman, LTW outperforms Kakadu. So, a full optimization of LTW codec will certainly increase the coding speed and it will reduce the memory requirements even more, making the codec a very competitive still image coding solution.

6 Conclusions

In this paper, we have presented three different rate control algorithms suitable for non-embedded wavelet-based encoders. We have implemented them over the LTW encoder in order to evaluate their behavior and compare the performance results with JPEG 2000 and SPIHT encoders in terms of R/D, execution time and memory consumption. Furthermore, we have shown that we can add rate control functionality to non-embedded wavelet encoders without a significant increase of complexity and little performance loss. Among the proposed simple rate control algorithms, the LTW_RC proposal is the one that exhibits the best trade-off between R/D performance, coding delay (twice as fast as SPIHT and 8.8 times as fast as JPEG 2000) and overall memory usage (similar to original LTW).

Also, we have compared LTW_RC coder with a highly optimized version of JPEG2000 (Kakadu), resulting competitive in terms of coding delay (up to 3.9 times as fast as Kakadu for medium size images) with slightly lower R/D performance. So, a full optimization process will make LTW_RC even faster and with lower memory requirements. These optimizations will be mainly focused on the DWT coding step by using fast and low memory demanding DWT techniques like line-based or block-based ones and exploiting the parallel capabilities of modern processors (like multithreading and SIMD instructions).

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

References

1. Antonini, M., Barlaud, M., Mathieu, P., & Daubechies, I. (1992). Image coding using wavelet transform. *IEEE Transaction on Image Processing, 1*(2), 205–220.
2. Cho, Y., & Pearlman, W.A. (2007). Hierarchical dynamic range coding of wavelet subbands for fast and efficient image compression. *IEEE Transactions on Image Processing, 16*, 2005–2015.
3. Chrysafis, C., Said, A., Drukarev, A., Islam, A., & Pearlman, W. (2000). SBHP—A low complexity wavelet coder. In *IEEE international conference on acoustics, speech and signal processing*.
4. CIPR: http://www.cipr.rpi.edu/resource/stills/kodak.html. Center for Image Processing Research.
5. Davis, P. J. (1975) *Interpolation and approximation*. Dover Publications.
6. Grottke, S., Richter, T., & Seiler, R. (2006). Apriori rate allocation in wavelet-based image compression. In *Second international conference on automated production of cross media content for multi-channel distribution, 2006. AXMEDIS '06* (pp. 329–336). doi:10.1109/AXMEDIS.2006.12.
7. Guo, J., Mitra, S., Nutter, B., & Karp, T. (2006). Backward coding of wavelet trees with fine-grained bitrate control. *Journal of Computers, 1*(4), 1–7. doi:10.4304/jcp.1.4.1-7.
8. ISO/IEC 10918-1/ITU-T Recommendation T.81 (1992). Digital compression and coding of continuous-tone still image.
9. ISO/IEC 15444-1 (2000). *JPEG2000 image coding system*.
10. Kakadu, S. (2006). http://www.kakadusoftware.com.
11. Kasner, J., Marcellin, M., & Hunt, B. (1999). Universal trellis coded quantization. *IEEE Transactions on Image Processing, 8*(12), 1677–1687. doi:10.1109/83.806615.
12. Lancaster, P. (1986). *Curve and surface fitting: An introduction*. Academic Press.
13. Oliver, J., & Malumbres, M. (2001). A new fast lower-tree wavelet image encoder. In *Proceedings of international conference on image processing, 2001* (Vol. 3, pp. 780–783). doi:10.1109/ICIP.2001.958236.

14. Oliver, J., & Malumbres, M. P. (2006). Low-complexity multiresolution image compression using wavelet lower trees. *IEEE Transactions on Circuits and Systems for Video Technology, 16*(11), 1437–1444.
15. Pearlman, W. A. (2001). Trends of tree-based, set partitioning compression techniques in still and moving image systems. In *Picture coding symposium*.
16. Said, A., & Pearlman, A. (1996). A new, fast and efficient image codec based on set partitioning in hierarchical trees. *IEEE Transactions on Circuits, Systems and Video Technology, 6*(3), 243–250.
17. Table Curve 3D 3.0 (1998). http://www.systat.com. Systat Software Inc.
18. Wu, X. (2001). *The transform and data compression handbook, chap. Compression of wavelet transform coefficients*, (pp. 347–378). CRC Press.
19. Zhidkov, N., & Kobelkov, G. (1987). *Numerical methods*. Moscow: Nauka.

Otoniel Mario López Granado received his M.S. degree in computer science from the University of Alicante (Spain) in 1996. Between 1997–2003 he worked as programmer analyst in an important industrial informatics firm. In 2003 he joined to the Computer Engineering Department at Miguel Hernandez University (UMH), Spain, as an assistant professor. Then, he received the Ph.D. degree in computer science in 2010. Currently, his research and teaching activities are related to multimedia networking (audio/video coding and network delivery).

Miguel Onofre Martínez-Rach received his M.S. degree in computer science from the University of Alicante (Spain) in 1995. Between 1996–2003 he worked as programmer analyst and datawarehouse consultant in a multinational computer hardware manufacturer. In 2003 he joined to the Computer Engineering Department at Miguel Hernandez University (UMH), Spain, as an assistant professor. He is finishing his Ph.D. degree in computer science. Currently, his research and teaching activities are related to multimedia networking (audio/video coding and network delivery).

Pablo Piñol Peral received his M.S. degree in computer science from the Universidad Politécnica de Valencia (UPV) in Spain in 1997. He worked for six years as a software developer in areas such as client-server database applications and web programming. In 2003 he joined the Universidad Miguel Hernández (UMH) in Spain as an assistant professor where he has been teaching computer networks, programming, and information systems security. He is currently a Ph.D. student in this university and his research activities are related to multimedia networking (audio/video coding and network delivery).

Jose Oliver Gil received the M.S. and Ph.D. degrees in computer engineering from the Universidad Politécnica de Valencia (UPV) in Spain, in 1998 and 2006, respectively, receiving Outstanding Ph.D. Award from UPV in 2008. He is with the Department of Computer Engineering (DISCA) at UPV since 1999, where he holds a lecturer position gained by means of the "programa cantera" in 2001. Since then, he teaches computer networks and multimedia networks at the Faculty of Computer Science, and he has served as a referee for several major conferences and journals. His research interests include image and video coding, and wireless transmission.

Manuel Pérez Malumbres received his B.S. degree in computer science from the University of Oviedo (Spain) in 1986. In 1989 he joined to the Computer Engineering Department (DISCA) at Technical University of Valencia (UPV), Spain, as an assistant professor. Then, he received the M.S. and Ph.D. degrees in computer science from UPV, at 1991 and 1996, respectively. In 2000 he became an associate professor and founded the Computer Networks Group (GRC) being the group director until June 2005. In September 2005, he moved to Miguel Hernandez University holding the same position. Currently he is technical committee member of IEEE Multimedia Communications Group (from 2004) and associate editor of the *Signal, Image and Video Processing* journal. He is author of more than 100 conference and journal publications and several networking books for undergraduate CS courses. Currently, his research and teaching activities are related to multimedia networking (audio/video coding and network delivery) and wireless technologies (MANETs and sensor networks).