2-Step Scalar Deadzone Quantization for Bitplane Image Coding

Francesc Aulí-Llinàs, *Member, IEEE*

***Abstract*—Modern lossy image coding systems generate a quality progressive codestream that, truncated at increasing rates, produces an image with decreasing distortion. Quality progressivity is commonly provided by an embedded quantizer that employs uniform scalar deadzone quantization (USDQ) together with a bitplane coding strategy. This paper introduces a 2-step scalar deadzone quantization (2SDQ) scheme that achieves same coding performance as that of USDQ while reducing the coding passes and the emitted symbols of the bitplane coding engine. This serves to reduce the computational costs of the codec and/or to code high dynamic range images. The main insights behind 2SDQ are the use of two quantization step sizes that approximate wavelet coefficients with more or less precision depending on their density, and a rate-distortion optimization technique that adjusts the distortion decreases produced when coding 2SDQ indices. The integration of 2SDQ in current codecs is straightforward. The applicability and efficiency of 2SDQ is demonstrated within the framework of JPEG2000.**

***Index Terms*—2-step scalar deadzone quantization, general embedded quantization, bitplane image coding, JPEG2000.**

I. Introduction

QUALITY progressivity is a feature provided by many image coding systems that permits the truncation of a codestream in a set of increasing rates at which the image distortion decreases strictly. This is of utility to applications that need to transmit, decode, or transcode images because it allows the partial processing of the codestream without needing to re-encode. Quality progressivity has been thoroughly studied and adopted by modern coding systems and standards in different forms. First wavelet-based coding engines like EZW [1] or SPIHT [2], for instance, generate an embedded codestream that can be truncated at any point providing the best possible quality for that rate. JPEG2000 standard [3] constructs a highly scalable codestream in which explicitly defined layers of quality can be identified and decoded providing optimal quality for that decoding rate.

Despite adopting different forms, most mechanisms that provide quality progressivity employ an embedded quantizer. An embedded quantizer is a procedure, or a device, that splits the quantization indices of a (transformed) image in short words. Each word is a suffix of the previous ones (if any) so that they can be consecutively transmitted and combined by the dequantizer to reconstruct the original indices with more or less precision depending on the transmitted words. Embedded quantization has been approached from different points of view. Progressively refinable vector quantization schemes are studied in [4]–[10], scalar quantization schemes that are adaptively adjusted as more data are transmitted are investigated in [11]–[13], and the best size for the deadzone of uniform scalar quantizers is determined in [14]. Embedded and multistage trellis coded quantization schemes [15] are explored in [16]–[21], and ordering strategies for wavelet data are examined in [22].

Early work on quantization suggested that a uniform scalar deadzone quantization (USDQ) scheme may be appropriate for a variety of sources [23]–[25]. Currently, most codecs employ USDQ together with a bitplane coding (BPC) strategy. Such a scheme splits the quantization indices into words of one bit that correspond to the binary representation of the indices. Bits from all indices are transmitted from the most significant bit to the least significant bit [26]. This is interpreted by the decoder as a multistage quantization procedure in which each stage produces quantization intervals half the size of the previous ones.

USDQ+BPC has become popular due to its competitive coding performance and the convenient use of the binary representation. Nonetheless, USDQ+BPC is not specifically designed to achieve optimal coding performance for a selected range of decoding rates, and it permits only small variations on the quantization scheme. Motivated by these issues, our previous work [27] introduced a more flexible scheme named general embedded quantization (GEQ). The focus of that work is to explore the performance, in terms of coding efficiency and quantizer complexity, that can be achieved by GEQ. Empirical evidence suggests that well-designed GEQ schemes can achieve same coding performance as that of USDQ+BPC while requiring fewer quantization stages.

Unfortunately, the approach of [27] requires structural modifications when introduced in conventional codecs. Mainly, these modifications are needed due to 1) the abandonment of the indices' binary representation, 2) a different order of coding passes, and 3) the selective operation of the quantizer on coefficients that vary at each stage. This entails a complete re-modulation of the codec because bitplane coding strategies can not be employed with such a quantization scheme. In addition, bit-wise operations, which are commonly exploited in software and hardware architectures to accelerate the coding process, can neither be utilized.

The purpose of this work is to introduce an embedded quantization scheme that –with*out* requiring modifications in the bitplane coding engine– decreases the stages of the

Dr. Francesc Aulí-Llinàs is with the Department of Information and Communications Engineering, Universitat Autònoma de Barcelona, Spain (phone: +34 935813571; fax: +34 935814477; e-mail: fauli@deic.uab.cat). This work has been partially supported by the Spanish Government (MINECO), by FEDER, and by the Catalan Government, under Grants RYC-2010-05671, TIN2012-38102-C03-03, and 2009-SGR-1224.

quantizer and achieves same coding performance as that of USDQ+BPC. The main advantages of the proposed scheme are that it reduces the computational load of the codec and permits the coding of high dynamic range images. The main insight behind it is a quantizer with 2 step sizes that are selectively employed depending on the magnitude of the coefficients. The use of distortion-optimization techniques that adjust the distortion decreases produced when coding 2SDQ indices is fundamental to achieve competitive coding performance. This paper continues our previous work [28] providing extended rate-distortion analysis, an efficient implementation that uses distortion estimators, and an experimental section with more results and different types of evaluations.

The paper is organized as follows. Section II reviews bitplane coding and describes the main idea behind GEQ. Section III introduces the proposed quantization scheme and details its implementation. Section IV demonstrates the advantages of the proposed quantizer through experimental results that assess coding performance, coding passes executed, and symbols emitted. The last section summarizes this work and provides conclusions.

II. Overview of current approaches

A. Bitplane image coding

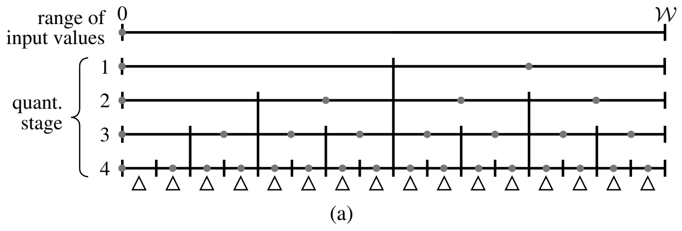
Let ω be a coefficient of a wavelet-transformed image that undergoes quantization through USDQ with base step size Δ. The quantizer partitions the range of input values into uniform intervals of width Δ except for the interval that contains zero (i.e., $(-\Delta, 0] \cup [0, \Delta)$), which is called deadzone and has width 2Δ because all coefficients within are mapped to zero. The operation carried out by USDQ at the encoder is expressed as

$$\upsilon = \left\lfloor \frac{|\omega|}{\Delta} \right\rfloor , \tag{1}$$

where $\lfloor\cdot\rfloor$ denotes the floor operation. Let $[b_{M-1}, b_{M-2}, ..., b_1, b_0]$, $b_i \in \{0,1\}$, be the binary representation for the quantization index υ, with M denoting a sufficient number of bits to represent all coefficients. The collection of bits b_j from all coefficients is called bitplane. Bitplane coding strategies code bits from the most significant bitplane $M-1$ to the least significant bitplane 0. The first non-zero bit of the binary representation of υ is denoted as b_s and is referred to as the significant bit. The sign of the coefficient is coded immediately after b_s, so that the dequantizer can reconstruct the coefficient somewhere in the indexed quantization interval. If $b_{j'}$ denotes the last available bit of υ, the reconstruction procedure carried out at the decoder is expressed as

$$\hat{\omega} = \begin{cases} 0 & \text{if } j' > s \\ \text{sign}(\omega)\ (\hat{\upsilon} + \delta)\Delta 2^{j'} & \text{otherwise} \end{cases} , \tag{2}$$

where $\hat{\upsilon} = [b_{M-1}, b_{M-2}, ..., b_{j'}]$, and $\delta \in [0,1)$ adjusts the reconstruction value $\hat{\omega}$ within its quantization interval. Typically, $\delta = 1/2$.



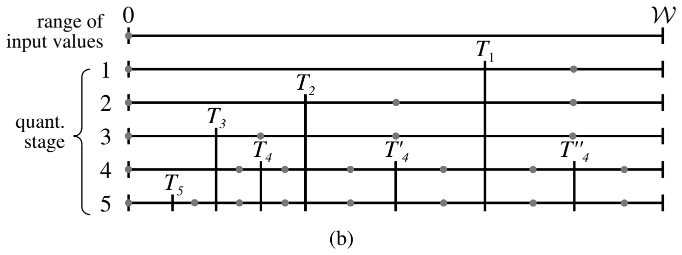


Fig. 1: Illustration of the quantization intervals employed by different embedded quantization schemes. The gray circles represent the reconstruction points when using $\delta = 1/2$. Only the magnitude of coefficients is depicted (omitting the sign) since symmetry about zero is assumed. (a) and (b) depict the USDQ+BPC scheme and the practical GEQ [27], respectively.

Fig. 1(a) illustrates the quantization intervals produced by the USDQ+BPC scheme. The top horizontal line of the figure represents the range of the input values in absolute value, i.e, $|\omega| \in [0, \mathcal{W}]$ with $\mathcal{W}$ denoting the largest magnitude of the coefficients to be quantized. The first quantization stage – represented as the second topmost horizontal line in Fig. 1(a)– partitions $[0, \mathcal{W}]$ into two intervals of same width. When using mid-point reconstruction, coefficients within $[\mathcal{W}/2, \mathcal{W}]$ are reconstructed in the middle of the interval as depicted with the gray dot in the figure. Coefficients within $[0, \mathcal{W}/2)$ are reconstructed as 0. Each quantization stage or, equivalently, the coding of each bitplane, halves the previous intervals. The procedure continues in this fashion until the width of all intervals is Δ, which occurs when all bitplanes are coded.

Bitplane coding of USDQ indices is often fractioned in multiple coding passes per bitplane, which helps to produce a more optimized codestream in terms of rate-distortion [29]. In general, most engines employ, at least, two coding passes. The first coding pass is devoted to significance coding, scanning coefficients that were not significant in previous bitplanes. The second pass is devoted to refinement coding, adjusting with more precision the magnitude of significant coefficients.

B. General embedded quantization

GEQ is defined as a multistage quantization scheme that uses quantization intervals of arbitrary width. To do so, the quantizer employs a threshold at each stage that approximates with more precision the coefficients whose magnitude lies within the same interval as that of the threshold. Let T_k denote the threshold employed in quantization stage k and let $[T_l, T_h)$ denote the interval in which threshold T_k lies (i.e., $T_l < T_k < T_h$). Quantization stage k operates *only* on coefficients $|\omega| \in [T_l, T_h)$, coding whether they are smaller than T_k or not. Conceptually, this splits quantization interval

$[T_l, T_h)$ in two (i.e., $[T_l, T_k)$ and $[T_k, T_h)$), approximating the magnitude of coefficients within with more precision. A detailed description of the GEQ coding procedure is found in [27].

The main advantage of GEQ is that widens the possibilities to design the quantizer. Note, for instance, that by restricting thresholds to be a multiple of a given step size Δ^*, the quantizer can chose among $\Gamma = \lfloor \mathcal{W}/\Delta^* \rfloor$ different thresholds. Assuming that all thresholds are distinct, there are then $\Gamma!/(\Gamma - K)!$ different quantizers of K quantization stages. Our previous work [27] explores the efficiency of GEQ exhaustively, disclosing those schemes that achieve the best rate-distortion performance. Inspired by the design of such quantizers, then a practical approach of GEQ is proposed and tested in the framework of JPEG2000.

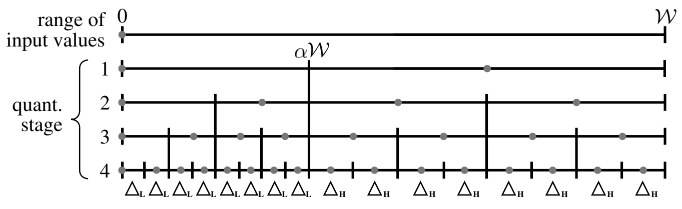
The practical GEQ proposed in [27] produces quantization intervals of width twice larger for coefficients $|\omega| \geq \mathcal{W}/3$ than for coefficients $|\omega| < \mathcal{W}/3$. Fig. 1(b) depicts the quantization intervals produced by the practical GEQ. The first three quantization stages carry out significance coding, with thresholds $T_1 = \frac{2}{3}\mathcal{W}$, $T_2 = \frac{1}{3}\mathcal{W}$, and $T_3 = \frac{1}{6}\mathcal{W}$. As indicated in the figure, the fourth stage of the quantizer halves all quantization intervals except the deadzone. The procedure continues interleaving one stage of significance coding with one stage of refinement coding until the target rate, or the target distortion, is achieved.

Unfortunately, the practical GEQ needs to be implemented in the coding engine by substituting the emission of bits corresponding to the binary representation of coefficients by symbols corresponding to the conditional that checks whether the magnitude of the coefficient lies below or above T_k. This prevents the use of the binary representation of the indices and, by extension, of bitplane coding strategies. In addition, the coding pass order needs to be modified to perform significance coding *exclusively* at the first three stages of the quantizer, and new data structures that store the quantization interval of each coefficient have to be added to permit the selective operation of the quantizer on some of the coefficients. All these operations are needed in the bitplane coding engine, which is commonly the most elaborated module of the codec (see [30], [31], for instance).

III. Proposed quantizer

A. Design

We define 2-step scalar deadzone quantization (2SDQ) as a scheme that employs two quantization step sizes as they are illustrated in Fig. 2. The intervals produced in the last stage of 2SDQ are intentionally similar to those of the practical GEQ, since this interval partitioning was proven to be most effective in [27]. 2SDQ forms these intervals in a way that permits their use in bitplane coding strategies. The quantizer employs step size Δ_L for coefficients $|\omega| < \alpha\mathcal{W}$, and step size Δ_H for coefficients $|\omega| \geq \alpha\mathcal{W}$, producing intervals with two different widths. We restrict step sizes to $\Delta_H > \Delta_L$, so that coefficients whose magnitude is greater than $\alpha\mathcal{W}$ are quantized more roughly than coefficients whose magnitude is smaller than $\alpha\mathcal{W}$. More important, the consequent use of


range of input values
0
$\mathcal{W}$
$\alpha\mathcal{W}$
quant. stage
1
2
3
4
Δ_L Δ_L Δ_L Δ_L Δ_L Δ_L Δ_L Δ_L Δ_H Δ_H Δ_H Δ_H Δ_H Δ_H Δ_H Δ_H

(a)

Fig. 2: Illustration of the quantization intervals employed by the proposed scheme.

bitplane coding compels to the partitioning of the intervals above and below $\alpha\mathcal{W}$ into the same number of subintervals. This restricts the size of Δ_H and Δ_L to

$$\Delta_H = \frac{(1-\alpha)\Delta_L}{\alpha} \; . \tag{3}$$

From the bitplane coding perspective, this scheme uses the widest step size Δ_H for quantized coefficients that are significant at the most significant bitplane, and Δ_L otherwise.

The operation carried out at the encoder is expressed as

$$\upsilon' = \begin{cases} \left\lfloor \dfrac{|\omega|}{\Delta_L} \right\rfloor & \text{if } |\omega| < \alpha\mathcal{W} \\ \left\lceil \dfrac{\alpha\mathcal{W}}{\Delta_L} \right\rceil + \left\lfloor \dfrac{|\omega| - \alpha\mathcal{W}}{\Delta_H} \right\rfloor & \text{otherwise} \end{cases} \; , \tag{4}$$

where $\lceil \cdot \rceil$ denotes the ceiling operation. This expression produces 2SDQ indices that are encoded by a conventional bitplane coding engine. The decoder reconstructs the coefficients according to

$$\hat{\omega}' = \begin{cases} 0 & \text{if } j' > s \\ \operatorname{sign}(\omega)\ (\hat{\upsilon}' + \delta)\Delta_L 2^{j'} & \\ & \text{if } j' \leq s \ \text{ and } \ \hat{\upsilon}' 2^{j'} < \left\lceil \dfrac{\alpha\mathcal{W}}{\Delta_L} \right\rceil \\ \operatorname{sign}(\omega)\ \left[\ \alpha\mathcal{W} + \left((\hat{\upsilon}' + \delta) 2^{j'} - \left\lceil \dfrac{\alpha\mathcal{W}}{\Delta_L} \right\rceil \right) \Delta_H \ \right] & \\ & \text{otherwise,} \end{cases} \tag{5}$$

where $\hat{\upsilon}'$ denotes the binary representation of υ' up to bit j'.

Our implementation of the 2SDQ scheme uses a Δ_L that is similar to that Δ employed by USDQ. This means that coefficients $|w| < \alpha\mathcal{W}$ are approximated similarly by USDQ and 2SDQ. Due to (3), this also implies that the Δ_H employed by 2SDQ is significantly larger than Δ, so coefficients whose magnitude is greater than $\alpha\mathcal{W}$ are approximated more roughly than when using USDQ.

The overall quantization error produced by 2SDQ is akin to that of USDQ. This is explained due to the distribution of coefficients in wavelet subbands. We recall that the probability

TABLE I: Evaluation of the cumulative pdf of coefficients in some wavelet subbands. The vertical and horizontal frequencies of the subband are indicated with two letters denoting high- (H) or low-frequencies (L), followed with the decomposition level in subscript. Each cell of the table is the cumulative probability up to $\sigma\mathcal{W}$.

	$\int_{-\sigma\mathcal{W}}^{\sigma\mathcal{W}} f(\omega)\ d\omega$						
$\sigma =$	0.01	0.02	0.05	0.10	0.15	0.30	0.60
“Portrait” (natural image)							
HL_1	68%	76%	86%	93%	96%	99.4%	99.9%
LH_2	59%	71%	81%	90%	94%	98%	99.9%
HH_3	58%	66%	76%	86%	90%	97%	99.8%
HL_4	63%	72%	86%	93%	96%	98%	99.8%
LH_5	60%	70%	83%	91%	95%	98%	99.9%
“Barcelona” (aerial image)							
HH_1	35%	41%	72%	89%	95%	99.6%	99.9%
LH_2	23%	32%	55%	76%	87%	98%	99.9%
HL_3	23%	31%	51%	71%	84%	97%	99.9%
HH_4	16%	22%	38%	52%	70%	92%	99.6%
LH_5	14%	20%	35%	55%	69%	92%	99.7%

density function (pdf) of wavelet coefficients has a Laplace-like shape [32] with long and thin tails. Such a pdf indicates that the density of coefficients with large magnitudes is much lower than the density of coefficients with small magnitudes. See, for instance, in Table I the cumulative pdf found in some wavelet subbands. The results reported in this table are generated applying five levels of irreversible 9/7 wavelet transform to two images of the corpus employed in Section IV. If $f(\omega)$ denotes the pdf of coefficients in a subband, each cell of the table reports the coefficient density in the range $(-\sigma\mathcal{W}, \sigma\mathcal{W})$ or, more precisely, $\int_{-\sigma\mathcal{W}}^{\sigma\mathcal{W}} f(\omega)\ d\omega$. Although that it depends on the image, the wavelet transform, and the subband, in general, more than 98% of the coefficients in a wavelet subband are smaller than $0.3\mathcal{W}$. Therefore, to reconstruct large-magnitude coefficients roughly does not have a significant impact on the overall quantization error due to the scarcity of such coefficients. The principal advantage of using a large Δ_H is that the number of stages of the quantizer is decreased. This is the main insight behind 2SDQ, which results in a codec that, performing fewer coding passes, achieves similar coding performance to that of USDQ.

α determines the coefficients that are quantized with Δ_L and with Δ_H. Our experience indicates that a good choice is to quantize 98% of coefficients, or more, with Δ_L. In general, $\alpha = 0.3$ is an appropriate choice for a large variety of images, wavelet filters, and subbands, so it is used in the experiments of Section IV.

B. Implementation

2SDQ can be implemented in any wavelet-based coding system that employs bitplane coding. Herein, it is tested in the framework of the JPEG2000 standard (ISO/IEC 15444-1). JPEG2000 is chosen due to its widespread use, advanced features, and excellent coding performance. Fig. 3 depicts the main stages of a typical JPEG2000 implementation [3]. The first group of operations decorrelates the image information through a color and a wavelet transform, and quantizes wavelet data using USDQ. Then, the image is conceptually partitioned in small sets of wavelet coefficients called codeblocks. The second group of operations codes the quantized coefficients within codeblocks using a three-coding-pass bitplane coding engine. Symbols emitted by the coding engine are fed to the arithmetic coder MQ. The last operations build the final codestream selecting bitstream segments of codeblocks through rate-distortion optimization (RD-opt) techniques. They also code auxiliary information.

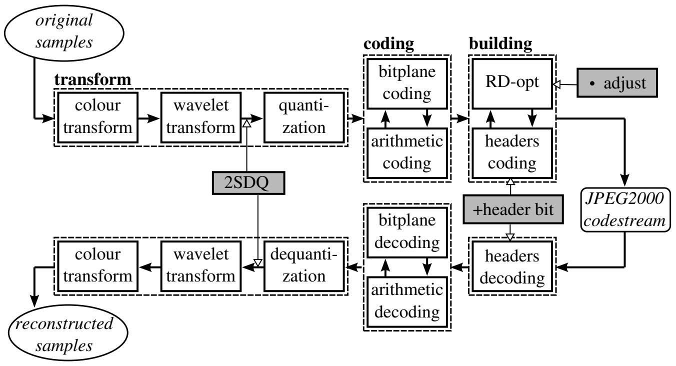
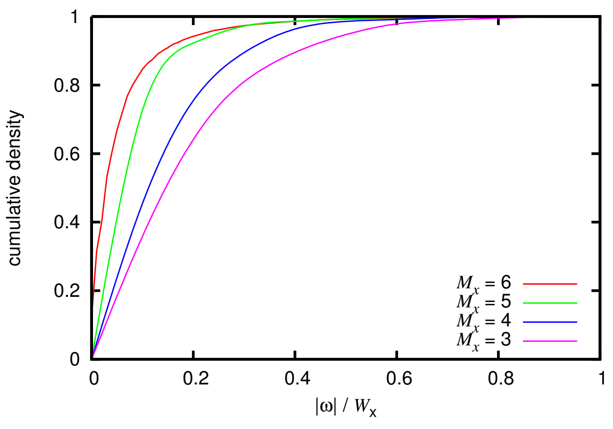


Fig. 3: Main stages of a typical JPEG2000 implementation. 2SDQ is integrated in the coding pipeline through the operations in gray.

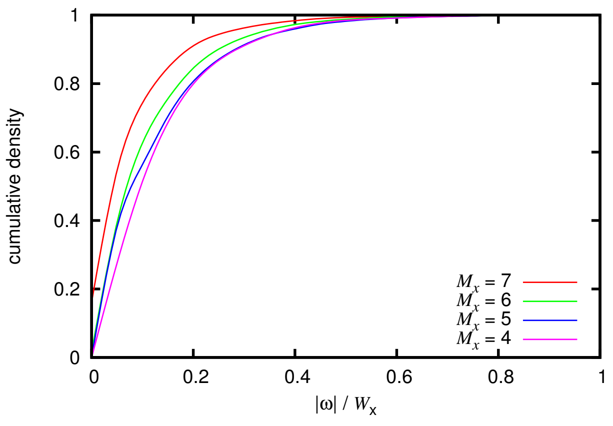
2SDQ is integrated in the framework of JPEG2000 introducing as few changes as possible. The gray boxes depicted in Fig. 3 are the operations that are introduced in the coding pipeline when applying 2SDQ. Rather than replacing the original quantization stage of JPEG2000 –which may not be suitable in some implementations–, 2SDQ is introduced through a strategy that converts wavelet coefficients to 2SDQ indices. This operation is labeled “2SDQ” in the figure.

Before describing the 2SDQ stage, let us explain the quantization operation carried out by JPEG2000. In a conventional JPEG2000 implementation, the coefficients in each subband are quantized using USDQ with step size Δ as detailed in Equation (1). This produces quantization indices $\upsilon \in [0, 2^{M_z})$, with M_z denoting a sufficient number of bits to represent all quantized coefficients in subband z. Δ can be chosen so that the coefficient of largest magnitude within the subband, say $\mathcal{W}_z$, approaches 2^{M_z}. Nonetheless, the quantized coefficients within a codeblock, referred to as $\upsilon[\chi]$ for codeblock x, may suffice with a smaller number of bits, i.e., $\upsilon[\chi] \in [0, 2^{M_x})$, with $M_x \leq M_z$. This is taken into account in JPEG2000 by transmitting M_x to the decoder, so that the bitplane coding engine avoids coding $M_z - M_x$ bitplanes that contain zeroes for that codeblock.

Herein, 2SDQ is applied on selected codeblocks by decreasing in $\mathcal{R}_x$ the number of magnitude bits that are used to code the coefficients via USDQ. When 2SDQ is applied on the



(a)



(b)

Fig. 4: Evaluation of the cumulative pdf of coefficients in codeblocks with different M_x. Data belong to codeblocks of subbands (a) HL$_1$ of "Portrait" and (b) HH$_1$ of "Barcelona".

codeblock, $M'_x = M_x - \mathcal{R}_x$ bitplanes are coded. The step sizes Δ_L and Δ_H are then

$$\Delta'_L = \Delta \frac{\alpha 2^{M_x}}{2^{M'_x - 1}} = \Delta \alpha 2^{\mathcal{R}_x + 1} \tag{6}$$

and

$$\Delta'_H = \Delta (1-\alpha) 2^{\mathcal{R}_x + 1} \ . \tag{7}$$

The codeblocks in which 2SDQ is applied and their corresponding M'_x are signaled in the headers of the codestream as depicted with the operation labeled "+header bit" in Fig. 3. The extra bits necessary to transmit this information increase negligibly the length of the final codestream.

Even though 2SDQ might be applied using other techniques, our experience indicates that the application of 2SDQ in codeblocks as described before enhances its efficiency since the number of dismissed bits $\mathcal{R}_x$ can be adjusted independently for each codeblock. The pdf of coefficients is again the reason behind this strategy. Fig. 4 reports the cumulative pdf of the subbands of the first decomposition level reported in Table I. The pdf of codeblocks with different M_x is reported separately in this figure, so each plot is the *average* cumulative pdf for codeblocks with same M_x. To ease the comparison, the horizontal axis of the figure is normalized to the nominal range, i.e., we plot $|\omega|/\mathcal{W}_x$ on the horizontal axis, with $\mathcal{W}_x$ denoting the coefficient of largest magnitude in the codeblock. Results indicate that codeblocks with few magnitude bits (i.e., with a small M_x) have a more uniform distribution than codeblocks with large M_x. As described before, 2SDQ approximates roughly large-magnitude coefficients, so its use is more appropriate in codeblocks having many bitplanes since the density of large-magnitude coefficients is low.

As seen in the next section, the selection of codeblocks in which 2SDQ is applied can be carried out using different strategies. The conversion from wavelet coefficients to 2SDQ indices is carried out as follows. The encoder applies the 2SDQ stage just after the wavelet transform (see Fig. 3) performing the operation expressed as

$$\upsilon'' = \begin{cases} \dfrac{|\omega|}{\alpha 2^{\mathcal{R}_x+1}} & \text{if } |\omega| < \Delta\alpha 2^{M_x} \\[2ex] \Delta 2^{M'_x - 1} + \dfrac{|\omega| - \Delta\alpha 2^{M_x}}{(1-\alpha)2^{\mathcal{R}_x+1}} & \text{otherwise} \end{cases} . \tag{8}$$

Then υ'' is quantized in the JPEG2000 quantization stage (i.e., as in (1) but replacing $|\omega|$ by υ'') and coded through conventional bitplane coding. The (partially) transmitted index to the decoder is transformed in the JPEG2000 dequantization stage producing 2SDQ indices that are referred to as $\hat{\upsilon}''$ (i.e., (2) is applied replacing $\hat{\omega}$ by $\hat{\upsilon}''$). 2SDQ indices are converted to wavelet coefficients just before the wavelet transform through

$$\hat{\omega} \approx \begin{cases} \text{sign}(\omega)\ |\hat{\upsilon}''| \alpha 2^{\mathcal{R}_x+1} & \text{if } |\hat{\upsilon}''| < \Delta 2^{M'_x-1} \\ \text{sign}(\omega)\ \left[\ \Delta\alpha 2^{M_x} + (|\hat{\upsilon}''| - \Delta 2^{M'_x-1})(1-\alpha)2^{\mathcal{R}_x+1}\ \right] & \\ & \text{otherwise.} \end{cases} \tag{9}$$

Contrarily to Equations (4) and (5), the two expressions above multiply α by 2^{M_x} instead of using $\mathcal{W}$ because $\mathcal{W}$ is not commonly available in implementations. In general, this does not penalize performance.

We note that the proposed 2SDQ scheme may also be seen as a compander [33]–[35] that employs the piecewise functions expressed in (8) and (9) as the compression and expanding functions, respectively. Appendix A expands this point of view further.

C. *Rate-distortion optimization aspects*

JPEG2000 employs RD-opt techniques to construct the final codestream [36], [37]. When the distortion metric is Mean Squared Error (MSE), RD-opt methods commonly determine the image distortion achieved at the end of coding pass l computing the squared difference between the original coefficients and the reconstructed coefficients of the codeblock, i.e.,

$$D_l = G_z^2 \sum_{\chi} (\omega[\chi] - \hat{\omega}[\chi])^2 \ , \tag{10}$$

with G_z denoting the energy gain factor of subband z to which the codeblock belongs. Then, the distortion decrease produced when transmitting coding pass l is determined as $\nabla D_l = D_{l-1} - D_l$. The distortion decrease together with the increase in rate is employed by optimization procedures such as the generalized Lagrange multiplier [38] to select the bitstream segments that are included in the final codestream.

In practice, the encoder neither keeps in memory the original coefficients nor reconstructs them at each coding pass to compute the squared error as it is formulated in (10). Generally, the squared difference between the original and the reconstructed coefficients is approximated using the quantization indices according to

$$D_l \approx G_z^2 \Delta^2 \sum_{\chi} \left((\upsilon[\chi] + \delta) - \begin{cases} 0 & \text{if } j' > s \\ (\hat{\upsilon}[\chi] + \delta) 2^{j'} & \text{otherwise} \end{cases} \right)^2 \ . \tag{11}$$

In this expression, $\upsilon[\chi] + \delta$ is the reconstructed coefficient when all bits are transmitted to the decoder, whereas $(\hat{\upsilon}[\chi] + \delta)2^{j'}$ is the reconstructed coefficient when bits up to j' are transmitted. Expression (11) is computationally simpler than (10) since $\upsilon[\chi]$ is employed by the bitplane coding engine and $(\hat{\upsilon}[\chi]+\delta)2^{j'}$ can be computed through bit-wise operations when $\delta = 1/2$. The power of two is the only computationally complex operation, though it can also be avoided as described below.

The RD-opt method must take into account that 2SDQ produces a deviation on these quantities. Before determining this deviation, we first define β as the scale factor of conventional USDQ indices to indices in which the 2SDQ stage is applied, more precisely, $\beta = \upsilon/\upsilon'''$, with υ''' representing the index obtained via (8) and (1). Fig. 5 reports β for different values of α when $\mathcal{R}_x = 1$. Indices $\upsilon''' < 2^{M'_x - 1}$ are *uniformly* scaled with respect to USDQ indices. For such indices, $\beta_L = \alpha 2^{\mathcal{R}_x + 1}$, which is seen in the figure as the uniform β from 0 to $2^{M'_x-1}$. Indices $\upsilon''' \geq 2^{M'_x-1}$ are *logarithmically* scaled with respect to USDQ indices. As indicated in the figure, β grows logarithmically from $2^{M'_x-1}$ to $2^{M'_x}$, being $\beta = \alpha 2^{\mathcal{R}_x+1}$ at $2^{M'_x-1}$ and $\beta = 2^{\mathcal{R}_x}$ at $2^{M'_x}$.

β indicates that the deviation produced on the squared error computed in (11) when coding 2SDQ indices has to be determined differently for indices that are smaller or greater than $2^{M'_x-1}$. The deviation for indices $\upsilon''' < 2^{M'_x-1}$ is β_L^2. The deviation for indices $\upsilon''' \geq 2^{M'_x-1}$ is different depending on their magnitude due to the non-uniform scale factor seen in Fig. 5. Even so, we recall that coefficient densities corresponding to indices $\upsilon''' \geq 2^{M'_x-1}$ are very low, so an estimate is enough for our RD-opt purposes. This estimate is computed as the average scale factor in the interval according to

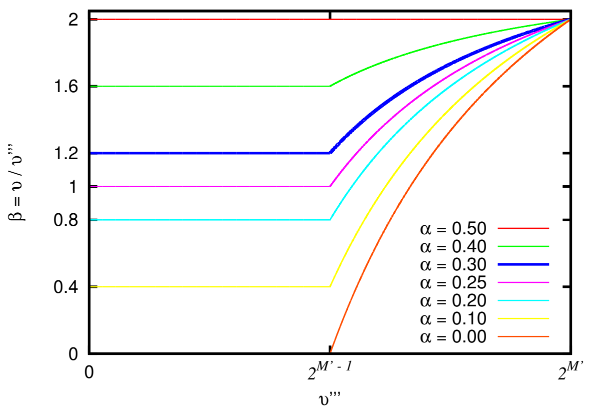


Fig. 5: Evaluation of the scale factor β. Experiments of Section IV employ $\alpha = 0.3$. Results are reported for $\mathcal{R}_x = 1$.

$$\begin{aligned} \beta_H &= \int_{\Delta\alpha 2^{M_x}}^{\Delta 2^{M_x}} f(|\omega|) \ \beta \ d\omega \\ &\approx \int_{\Delta\alpha 2^{M_x}}^{\Delta 2^{M_x}} f(|\omega|) \frac{\left\lfloor \frac{|\omega|}{\Delta} \right\rfloor}{2^{M'_x - 1} + \left\lfloor \frac{|\omega|/\Delta - \alpha 2^{M_x}}{(1-\alpha)2^{\mathcal{R}_x + 1}} \right\rfloor} \ d\omega \\ &\approx (\alpha 2 \ln 2 - \alpha + 1 - \ln 2) 2^{\mathcal{R}_x + 1} \ . \end{aligned} \tag{12}$$

The above equation is expressed using coefficients ω and densities $f(\omega)$ as they are defined previously to allow integration. The result on the third line of the expression is computed assuming a uniform distribution in the interval (i.e., $f(\omega) = 1/(1-\alpha)\Delta 2^{M_x}$). This simplifies the calculation without penalizing performance.

Through β_L and β_H, the squared error determined in Equation (11) is reformulated for codeblocks using 2SDQ according to Equation (13).

Even though implementations may employ (13), the use of the recently introduced distortion estimators [39] can further reduce the computational complexity of this expression. The main idea behind the distortion estimators is to approximate distortion decreases through the number of significant and refinement coefficients coded at each coding pass. Distortion estimators $\nabla\mathcal{D}_{j'}^{sig}$ and $\nabla\mathcal{D}_{j'}^{ref}$ are respectively defined as the average squared error decrease that is produced when a coefficient is found significant, or is refined, at bitplane j'. The details on how these estimators are determined can be found in [39]. The distortion decrease at coding pass l is then determined according to

$$\nabla D_l \approx G_z^2 \Delta^2 (\nabla\mathcal{D}_{j'}^{sig} \cdot \#S_l + \nabla\mathcal{D}_{j'}^{ref} \cdot \#R_l) \ , \tag{14}$$

where $\#S_l$ and $\#R_l$ denote the number of significant and refinement coefficients coded in coding pass l, respectively. The main advantage with respect to the classic approach embodied in (11) is that (14) neither requires the (partial) reconstruction of the coefficient nor the power of two. The only operation carried out in (14) during bitplane coding is to

$$D_l \approx G_z^2 \Delta^2 \begin{cases} \beta_L^2 \sum_\chi \left((\upsilon[\chi] + \delta) - \begin{cases} 0 & \text{if } j' > s \\ (\hat{\upsilon}[\chi] + \delta) 2^{j'} & \text{otherwise} \end{cases} \right)^2 & \text{if } \upsilon[\chi] < 2^{M'_x - 1} \\ \beta_H^2 \sum_\chi \left((\upsilon[\chi] + \delta) - \begin{cases} 0 & \text{if } j' > s \\ (\hat{\upsilon}[\chi] + \delta) 2^{j'} & \text{otherwise} \end{cases} \right)^2 & \text{otherwise} \end{cases} \tag{13}$$

count the number of significant/refinement coefficients at each coding pass, which is computationally inexpensive.

Distortion estimators can be used together with 2SDQ. Due to the uniform scale factor of indices $\upsilon''' < 2^{M'_x - 1}$, the deviation on $\nabla \mathcal{D}_{j'}^{sig}$ and $\nabla \mathcal{D}_{j'}^{ref}$ for such coefficients is β_L^2. For indices $\upsilon''' \geq 2^{M'_x - 1}$, the deviation produced for significance coding is determined according to

$$\begin{aligned} &\beta_{Hsig}^2 = \\ &\frac{\int_{\alpha 2^{M_x}}^{2^{M_x}} f(|\omega|) \left[\omega^2 - \left(\omega - \left(\alpha 2^{M_x} + (1-\alpha) 2^{M_x - 1} \right) \right)^2 \right] d\omega}{\int_{2^{M'_x - 1}}^{2^{M'_x}} f(|\omega|) \left[\omega^2 - \left(\omega - \left(2^{M'_x - 1} + 2^{M'_x - 2} \right) \right)^2 \right] d\omega} \\ &= (\alpha^2 + 2\alpha + 1) \frac{2^{2\mathcal{R}_x + 2}}{9} \ . \end{aligned} \tag{15}$$

The numerator and denominator in the second line of this expression are the average squared error decrease that is produced when coefficients quantized with USDQ and 2SDQ are found significant, respectively. In the numerator, ω^2 is the squared error produced for the coefficient when none bit is transmitted, whereas $\left(\omega - \left(\alpha 2^{M_x} + (1-\alpha) 2^{M_x - 1} \right) \right)^2$ is the squared error produced when the significant bit is transmitted. The denominator is derived accordingly. Again, $f(\omega)$ is assumed uniform in (15). For notational simplicity, coefficients are assumed to be normalized by the quantization step size Δ in (15). The deviation for refinement coding is derived similarly, resulting in $\beta_{Href}^2 = (\alpha^2 - 2\alpha + 1) 2^{2\mathcal{R}_x + 2}$. Then, the squared error determined in Expression (14) is reformulated for codeblocks using 2SDQ as

$$\nabla D_l \approx G_z^2 \Delta^2 \begin{cases} \beta_{Hsig}^2 \cdot \nabla \mathcal{D}_{j'}^{sig} \cdot \# S_l & \text{if } j' = M'_x - 1 \\ \beta_L^2 \cdot \nabla \mathcal{D}_{j'}^{sig} \cdot \# S_l \ + & \\ \beta_L^2 \cdot \nabla \mathcal{D}_{j'}^{ref} \cdot (\# R_l - \# R_{M'_x - 1}) \ + & \\ \beta_{Href}^2 \cdot \nabla \mathcal{D}_{M'_x - 1}^{ref} \cdot \# R_{M'_x - 1} & \text{otherwise} \end{cases} \tag{16}$$

where $\# R_{M'_x - 1}$ is the number of coefficients found significant in the most significant bitplane of the codeblock. Experimental evidence indicates that the performance achieved with (13) and (16) is virtually the same, so our implementation computes distortion decreases via (16). It is represented in Fig. 3 as the operation labeled "· adjust"

D. Summary

To summarize, the integration of 2SDQ into the coding pipeline of JPEG2000 requires three operations at the encoder and two operations at the decoder. The first operation transforms wavelet coefficients to 2SDQ indices. This operation requires one or two floating point operations per coefficient at the codeblocks in which 2SDQ is applied. This increases in less than 1% the computational costs of a conventional JPEG2000 implementation. The codeblocks in which 2SDQ is applied are signaled in the headers of the codestream, so the decoder can identify them. This second operation has negligible computational costs. The third operation multiplies the distortion decreases computed by the RD-opt method by the deviations determined above. This operation does not imply significant computational resources either, and is applied at the encoder only.

IV. Experimental results

The proposed method is evaluated in terms of coding performance achieved, coding passes executed, and symbols emitted by the bitplane coding engine. The results achieved with 2SDQ are compared with those achieved with a compliant JPEG2000 implementation using USDQ. The images employed in the experiments are chosen from different corpora: "Portrait" (2048×2560) and "Flowers" (2731×2048) are natural images that belong to the ISO 12640-1 and ISO 12640-2 corpus, respectively, "Barcelona" (4096×4096) is an aerial image provided by the Cartographic Institute of Catalonia (ICC) [40] that belongs to the remote sensing community, and "Hip" (2048×2495) is a computer radiology provided by the UDIAT Centre Diagnostic [41] that belongs to the medical community. All images are 8 bit, gray scale. Coding parameters are: 5 levels of irreversible 9/7 wavelet transform, codeblock size of 64×64, single quality layer codestreams, and no precincts. The codestream generated when 2SDQ is in use is not JPEG2000 compliant.

A. Optimal coding performance

First, 2SDQ is employed with the aim to achieve same coding performance as that of USDQ while minimizing the number of coding passes performed by the bitplane coding engine. This may help to reduce the computational load of the codec since to execute more coding passes commonly implies additional computational time [29]. Our JPEG2000 implementation [42] determines the USDQ step size Δ according to the L$_2$-norm of the subband synthesis basis, which is a common practice in JPEG2000. In this context, 2SDQ is applied in codeblocks with $M_x \geq 5$ dismissing one bitplane

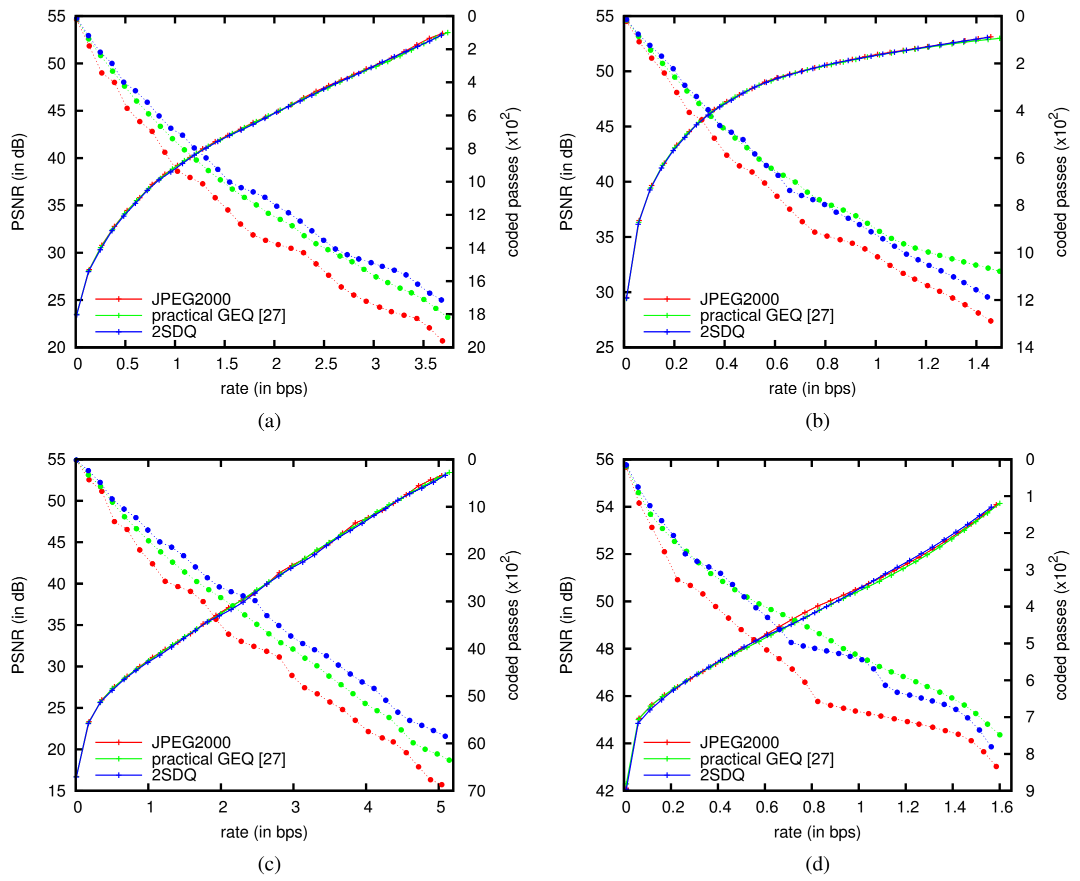
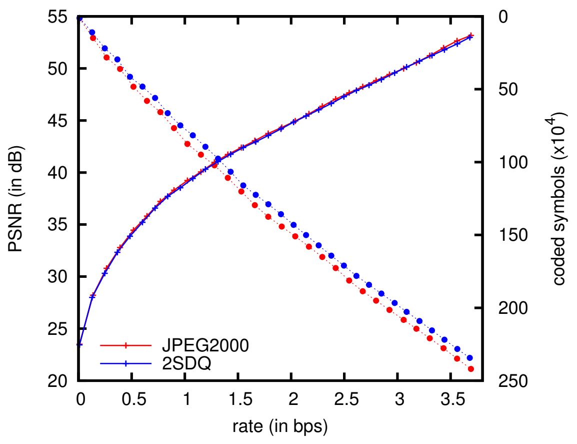


Fig. 6: Evaluation of the coding performance and coding passes executed by JPEG2000, the practical GEQ, and 2SDQ. The setting of 2SDQ is aimed to achieve optimal coding performance in this test. (a) “Portrait” (b) “Flowers” (c) “Barcelona” (d) “Hip”.

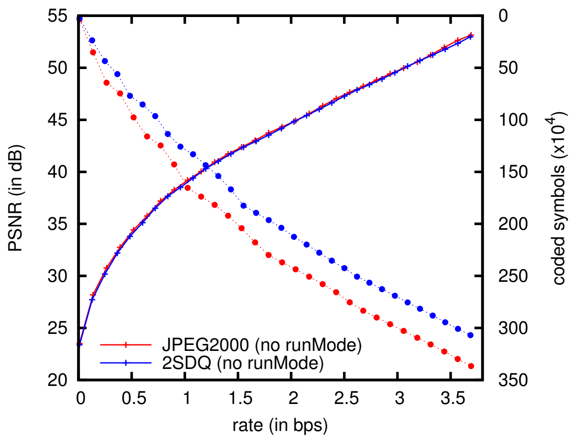
(i.e., $\mathcal{R}_x = 1$), except for the “Hip” image, in which 2SDQ is applied in codeblocks with $M_x \geq 4$ dismissing one bitplane as well.[1] This strategy is applied on all codeblocks of the image except those within the lowest frequencies subband (LL), since the pdf in LL is not Laplace-like. We recall that LL corresponds to the smallest resolution level, containing few coefficients. Fig. 6 depicts the results achieved. The horizontal axis of these figures is the rate, reported in bits per sample (bps), whereas the left vertical axis is the Peak Signal to Noise Ratio (PSNR) and the right vertical axis is the number of coding passes executed. For clarity, the right axis is reversed (higher means fewer coding passes). For comparison, these figures also report the performance achieved by the practical GEQ [27]. The coding performance achieved by the three approaches, namely, JPEG2000, the practical GEQ, and 2SDQ is very similar for the four images evaluated. JPEG2000 is the approach that executes more coding passes, whereas the practical GEQ and 2SDQ achieve similar results. In this context, results indicate that 2SDQ achieves same coding performance as that of JPEG2000 while executing significantly fewer coding passes. For the “Hip” image at 1 bps, for instance, 2SDQ codes 22% fewer coding passes than JPEG2000 while achieving same PSNR. Results also suggest that 2SDQ achieves virtually same results as those of the practical GEQ [27], indicating the (near-)optimality of the proposed 2SDQ scheme.

Even though the previous tests indicate that 2SDQ decreases the coding passes executed, it is of interest to evaluate the number of symbols that the bitplane coding engine emits. In general, emitted symbols are directly related to the computational time spent by the bitplane coding procedure. Coding passes executed and symbols emitted are *not* directly related since coding engines may use strategies to code more than one coefficient per emitted symbol [3]. Fig. 7(a) reports the results achieved for the “Portrait” image. For completeness, the figure also reports coding performance, which corresponds to that reported in Fig. 6(a). These results suggest that the number of symbols emitted by the 2SDQ is lower than that of JPEG2000, though the differences are not as significant as those found when evaluating coding passes. This is caused due to the run mode of JPEG2000. The run mode of JPEG2000 is a coefficient scanning mode that is activated under some special circumstances that occur mostly at the most significant bitplanes of the codeblock, in which most coefficients are not significant [3]. When it is activated, the run mode codes the significance state of four adjacent coefficients emitting a single binary symbol. The 2SDQ embeds more information at the

[1]The “Hip” image has low contrast, which causes that most codeblocks have lower M_xs than the codeblocks of the other images.



(a)



(b)

Fig. 7: Evaluation of the coding performance and symbols emitted by JPEG2000 and 2SDQ, for the "Portrait" image when using and not using the run mode of JPEG2000 in (a) and (b), respectively.

most significant bitplanes of the codeblock, so the run mode is not activated as often as when using USDQ. This is seen in Fig. 7(b), which reports the same evaluation as that of Fig. 7(a) but when the run mode of JPEG2000 is deactivated. In this figure, the differences between JPEG2000 and 2SDQ are more akin to those found in Fig. 6(a). Similar results hold for the other images.

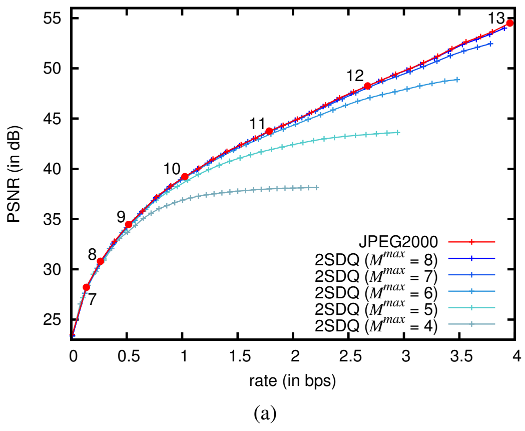
B. Limited bit-depth coding

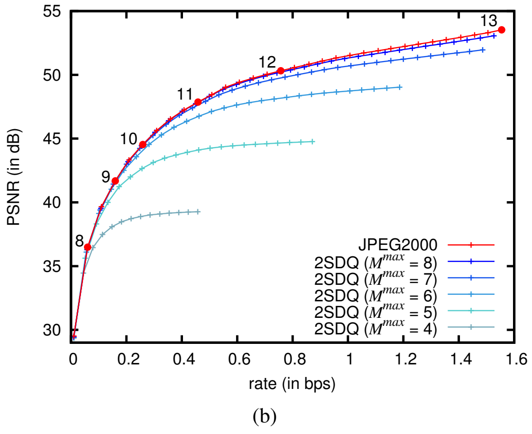
Next, 2SDQ is employed in an application that limits the number of magnitude bits available to code the coefficients. Among others, this requirement may appear in devices with memory constrained resources, when images with a very high bit-depth have to be coded, or when the use of ROI-specific techniques that enlarge the dynamic range of the image [43] are in use. 2SDQ is applied in codeblocks that have $M_x > M^{max}$ removing $\mathcal{R}_x = M_x - M^{max}$ magnitude bits, with M^{max} denoting the maximum number of magnitude bits allowed. Again, this is only applied on subbands other than the LL. Fig. 8 reports the coding performance achieved when 2SDQ is applied using $M^{max} = 8, 7, 6, 5,$ and 4. For comparison, the performance achieved by a conventional JPEG2000 implementation –without limits on the number of magnitude bits used– is also reported in this figure. To provide an idea on the number of bitplanes coded by this conventional JPEG2000 implementation, the JPEG2000 plot depicts the coding end of each bitplane with a dot and a label indicating the bitplane number. Studies on rate-distortion optimization [29], [37] show that the bitstream segments of all codeblocks corresponding to one bitplane are included in the final codestream before including segments of the immediately lower bitplane. Therefore, the dots depicted in the JPEG2000 plot of Fig. 8 indicate the number of magnitude bits that have been used by a conventional JPEG2000 implementation until that rate. Note that by using 8 magnitude bits (i.e., $M^{max} = 8$), 2SDQ achieves a coding performance similar to that achieved with a conventional JPEG2000 implementation using 13 bits, for all images reported. The coding performance achieved by 2SDQ is penalized when M^{max} is reduced, mostly from medium to high rates. Even so, 2SDQ codes an image with a very high quality using very few bitplanes. See, for instance, that using only 4 magnitude bits 2SDQ codes the "Portrait" image achieving a quality of 37 dB, whereas JPEG2000 needs almost 10 bits to achieve a similar quality.

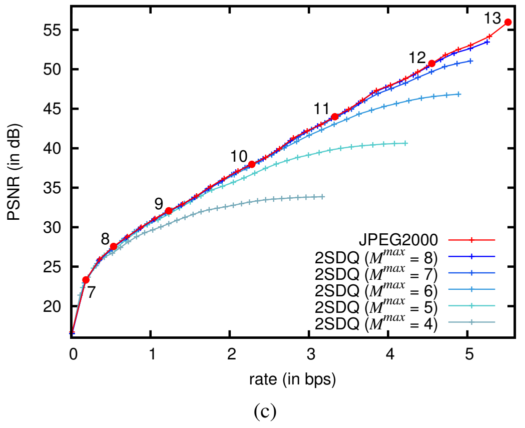
Although a conventional JPEG2000 implementation codes the image as stated before, a smarter JPEG2000-compliant strategy to code images using a limited dynamic range is to code only the most significant M^{max} bitplanes of each codeblock. Fig. 9 reports the performance achieved by such a strategy and 2SDQ. In this figure, the results achieved by 2SDQ are the same as those reported in Fig. 8(a). For each 2SDQ plot using a specific M^{max}, the figure depicts a JPEG2000 plot with the same point type that reports the performance achieved by this JPEG2000-compliant strategy. To avoid cluttering the figure, the key is simplified. The results suggest that, in this context, 2SDQ also achieves better coding performance than that of JPEG2000. The smaller the M^{max}, the larger the difference. At 2 bps, for instance, the difference between 2SDQ and JPEG2000 when $M^{max} = 5$ is almost 5 dB. Similar results hold for the other images.

V. Conclusions

Embedded quantization is a mechanism employed by most modern image coding systems to generate a quality progressive codestream. Commonly, embedded quantization is achieved by means of a bitplane coding (BPC) strategy that codes (wavelet) coefficients quantized through uniform scalar deadzone quantization (USDQ). Recent work has shown that the use of general embedded quantization (GEQ) can achieve virtually same coding performance as that of USDQ+BPC while reducing the number of quantization stages of the scheme. Unfortunately, the practical implementation of the GEQ in current systems requires modifications in the bitplane coding engine, which is the most sophisticated piece of the codec. This work overcomes this drawback through 2-step scalar deadzone quantization (2SDQ). The structure of the proposed 2SDQ scheme allows the use of an unmodified bitplane coding engine. This is achieved through a quantizer that applies two different step sizes depending on the magnitude of the







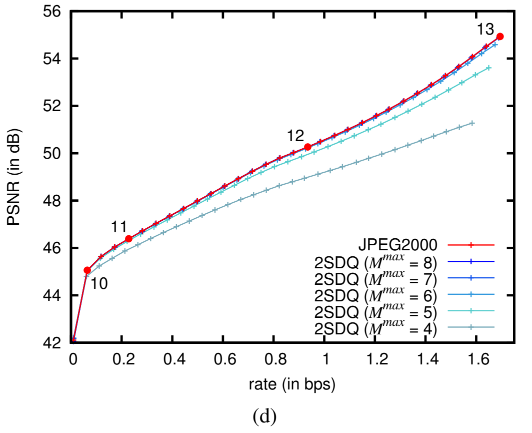


Fig. 8: Evaluation of the coding performance achieved by JPEG2000 and 2SDQ. 2SDQ is employed to decrease the number of magnitude bits required by the coding engine. (a) "Portrait" (b) "Flowers" (c) "Barcelona" (d) "Hip".

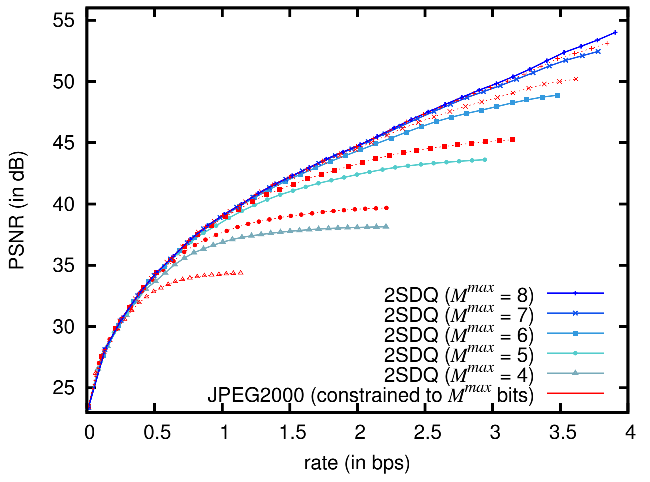


Fig. 9: Evaluation of the coding performance achieved by 2SDQ and a JPEG2000 implementation with the number of magnitude bits constrained to M^{max}, for the "Portrait" image.

coefficients, and a rate-distortion optimization analysis that establishes the deviations on the distortion produced when coding 2SDQ indices. The implementation of the 2SDQ in current codecs requires simple modifications that neither affect the codec's architecture nor its throughput. In this work, this is demonstrated integrating the 2SDQ in the core coding system of JPEG2000.

The main insight behind 2SDQ is to exploit the low density of coefficients with large magnitudes found in wavelet subbands, which are quantized rougher than the remaining coefficients. Conceptually, this can be seen as the introduction of more information at the most significant bitplanes of the quantized image, which in general contain mostly zeroes. The main advantage of 2SDQ is that decreases the number of magnitude bits (or bitplanes) coded without penalizing coding performance. This may be of utility to reduce the computational load of the codec, to simplify the scanning order of the bitplane coding engine, to transmit high quality images using few bitplanes, to code high bit-depth images, or to enhance the coding capabilities of devices with constrained resources.

Acknowledgment

The author thanks the associate editor and the anonymous reviewers for their comments, which helped to improve the quality of this manuscript.

Appendix A

A compander is a signal processing technique that defines a function through which an input signal is compressed before quantization and transmission, and then expanded again after

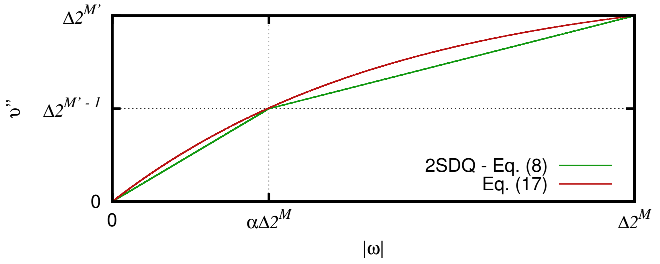


Fig. 10: Illustration of two companders. The horizontal axis is the input signal, whereas the vertical axis is the signal after companding.

dequantization. The 2SDQ scheme defined in this work can also be seen as a compander that employs the piecewise linear functions expressed in (8) and (9) to respectively compress and expand the signal. Fig. 10 illustrates the companding function embodied by the 2SDQ scheme.

In general, companders employ logarithmic or exponential functions to transform the signal. An alternative to (8), (9) might be the logarithmic function

$$\upsilon'' = \Delta 2^{M'_x} \left(1 - \frac{\phi^{1-|\omega|/\Delta 2^{M_x}} - 1}{\phi - 1} \right) , \qquad (17)$$

where ϕ represents its shape. Fig. 10 illustrates this function with $\phi = 6.25$. Logarithmic functions are not employed in this work because their are more computationally complex than the operations required in (8) and (9), and because our experience indicates that the use of such functions does not improve performance significantly.

References

[1] J. M. Shapiro, “Embedded image coding using zerotrees of wavelet coefficients,” *IEEE Trans. Image Process.*, vol. 41, no. 12, pp. 3445–3462, Dec. 1993.

[2] A. Said and W. A. Pearlman, “A new, fast, and efficient image codec based on set partitioning in hierarchical trees,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, no. 3, pp. 243–250, Jun. 1996.

[3] D. S. Taubman and M. W. Marcellin, *JPEG2000 Image compression fundamentals, standards and practice*. Norwell, Massachusetts 02061 USA: Kluwer Academic Publishers, 2002.

[4] W.-Y. Chan, S. Gupta, and A. Gersho, “Enhanced multistage vector quantization by joint codebook design,” *IEEE Trans. Commun.*, vol. 40, no. 11, pp. 1693–1697, Nov. 1992.

[5] H. Jafarkhani and N. Farvardin, “A scalable wavelet image coding scheme using multi-stage pruned tree-structured vector quantization,” in *Proc. IEEE International Conference on Image Processing*, vol. 3, Oct. 1995, pp. 81–84.

[6] C. F. Barnes, S. A. Rizvi, and N. M. Nasrabadi, “Advances in residual vector quantization: A review,” *IEEE Trans. Image Process.*, vol. 5, no. 2, pp. 226–262, Feb. 1996.

[7] E. A. B. da Silva, D. G. Sampson, and M. Ghanbari, “A successive approximation vector quantizer for wavelet transform image coding,” *IEEE Trans. Image Process.*, vol. 5, no. 2, pp. 299–310, Feb. 1996.

[8] K. Bao and X.-G. Xia, “Image compression using a new discrete multiwavelet transform and a new embedded vector quantization,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 10, no. 6, pp. 833–842, Sep. 2000.

[9] D. Mukherjee and S. K. Mitra, “Successive refinement lattice vector quantization,” *IEEE Trans. Image Process.*, vol. 11, no. 12, pp. 1337–1348, Dec. 2002.

[10] ——, “Vector SPIHT for embedded wavelet video and image coding,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 3, pp. 231–246, Mar. 2003.

[11] P. W. Wong, “Progressively adaptive scalar quantization,” in *Proc. IEEE International Conference on Image Processing*, vol. 1, Oct. 1996, pp. 357–360.

[12] Z. Xiong, K. Ramchandran, and M. T. Orchard, “Space-frequency quantization for wavelet image coding,” *IEEE Trans. Image Process.*, vol. 6, no. 5, pp. 677–693, May 1997.

[13] A. Ortega and M. Vetterli, “Adaptive scalar quantization without side information,” *IEEE Trans. Image Process.*, vol. 6, no. 5, pp. 665–676, May 1997.

[14] G. J. Sullivan, “On embedded scalar quantization,” in *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 4, May 2004, pp. 605–608.

[15] M. W. Marcellin and T. R. Fischer, “Trellis coded quantization of memoryless and Gauss-Markov sources,” *IEEE Trans. Commun.*, vol. 38, no. 1, pp. 82–93, Jan. 1990.

[16] A. Aksu and M.Salehi, “Multistage trellis coded quantisation (MS-TCQ) design and performance,” *IEE Proceedings- Communications*, vol. 144, no. 2, pp. 61–64, Apr. 1997.

[17] H. Brunk and N. Farvardin, “Embedded trellis coded quantization,” in *Proc. IEEE Data Compression Conference*, Apr. 1998, pp. 93–102.

[18] H. Jafarkhani and V. Tarokh, “Successively refinable trellis coded quantization,” in *Proc. IEEE Data Compression Conference*, Apr. 1998, pp. 83–92.

[19] A. Bilgin, P. J. Sementilli, and M. W. Marcellin, “Progressive image coding using trellis coded quantization,” *IEEE Trans. Image Process.*, vol. 8, no. 11, pp. 1638–1643, Nov. 1999.

[20] P. Seigneurbieux and Z. Xiong, “Progressive trellis-coded space-frequency quantization for wavelet image coding,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 12, no. 7, pp. 587–591, Jul. 2002.

[21] S. Steger and T. Richter, “Universal refinable trellis coded quantization,” in *Proc. IEEE Data Compression Conference*, Mar. 2009, pp. 312–321.

[22] M. D. Gaubatz and S. S. Hemami, “Ordering for embedded coding of wavelet image data based on arbitrary scalar quantization schemes,” *IEEE Trans. Image Process.*, vol. 16, no. 4, pp. 982–996, Apr. 2007.

[23] H. Gish and J. N. Pierce, “Asymptotically efficient quantizing,” *IEEE Trans. Inf. Theory*, vol. 14, no. 5, pp. 676–683, 1968.

[24] N. Farvardin and J. W. Modestino, “Optimum quantizer performance for a class of non-gaussian memoryless sources,” *IEEE Trans. Inf. Theory*, vol. 30, no. 3, pp. 485–497, May 1984.

[25] G. J. Sullivan, “Efficient scalar quantization of Exponential and Laplacian random variables,” *IEEE Trans. Inf. Theory*, vol. 42, no. 5, pp. 1365–1374, Sep. 1996.

[26] M. W. Marcellin, M. A. Lepley, A. Bilgin, T. J. Flohr, T. T. Chinen, and J. H. Kasner, “An overview of quantization in JPEG 2000,” *ELSEVIER Signal Processing: Image Communication*, vol. 17, no. 1, pp. 73–84, Jan. 2002.

[27] F. Auli-Llinas, “General embedded quantization for wavelet-based lossy image coding,” *IEEE Trans. Signal Process.*, vol. 61, no. 6, pp. 1561–1574, Mar. 2013.

[28] ——, “Low complexity embedded quantization scheme compatible with bitplane image coding,” in *Proc. IEEE Data Compression Conference*, Mar. 2013, pp. 271–280.

[29] F. Auli-Llinas and M. W. Marcellin, “Scanning order strategies for bitplane image coding,” *IEEE Trans. Image Process.*, vol. 21, no. 4, pp. 1920–1933, Apr. 2012.

[30] Y.-Z. Zhang, C. Xu, W.-T. Wang, and L.-B. Chen, “Performance analysis and architecture design for parallel EBCOT encoder of JPEG2000,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 17, no. 10, pp. 1336–1347, Oct. 2007.

[31] K. Sarawadekar and S. Banerjee, “An efficient pass-parallel architecture for embedded block coder in JPEG 2000,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 21, no. 6, pp. 825–836, Jun. 2011.

[32] F. Auli-Llinas, “Stationary probability model for bitplane image coding through local average of wavelet coefficients,” *IEEE Trans. Image Process.*, vol. 20, no. 8, pp. 2153–2165, Aug. 2011.

[33] W. R. Bennett, “Spectra of quantized signals,” *Bell System Technical Journal*, vol. 27, no. 3, pp. 446–472, Jul. 1948.

[34] J. Z. Sun and V. K. Goyal, “Scalar quantization for relative error,” in *Proc. IEEE Data Compression Conference*, Mar. 2011, pp. 293–302.

[35] M. Petkovic, Z. Peric, and A. Mosic, “Optimisation of variable-length code for data compression of memoryless Laplacian source,” *IET Communications*, vol. 5, no. 7, pp. 906–913, Apr. 2011.

[36] F. Auli-Llinas, “Model-based JPEG2000 rate control methods,” Ph.D. dissertation, Universitat Autònoma de Barcelona, Barcelona, Spain, Dec. 2006. [Online]. Available: http://www.deic.uab.cat/~francesc

[37] F. Auli-Llinas and J. Serra-Sagrista, “JPEG2000 quality scalability without quality layers,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 18, no. 7, pp. 923–936, Jul. 2008.

[38] H. Everett, “Generalized Lagrange multiplier method for solving problems of optimum allocation of resources,” *Oper. Res.*, vol. 11, pp. 399–417, 1963.

[39] F. Auli-Llinas and M. W. Marcellin, “Distortion estimators for bitplane image coding,” *IEEE Trans. Image Process.*, vol. 18, no. 8, pp. 1772–1781, Aug. 2009.

[40] Institut Cartografic de Catalunya. (2013, May) ICC. Barcelona 08038 (Spain). [Online]. Available: http://www.icc.cat

[41] Corporacio Parc Tauli. (2013, May) UDIAT centre diagnostic. Sabadell 08208 (Spain). [Online]. Available: http://www.parctauli.es/webcspt/udiat

[42] F. Auli-Llinas. (2013) BOI codec. [Online]. Available: http://www.deic.uab.cat/~francesc/software/boi

[43] J. Bartrina-Rapesta, J. Serra-Sagrista, and F. Auli-Llinas, “JPEG2000 ROI coding through component priority for digital mammography,” *ELSEVIER Computer Vision and Image Understanding*, vol. 115, no. 1, pp. 59–68, Jan. 2011.

Francesc Aulí-Llinàs (S’2006-M’2008) is a Ramón y Cajal fellow in the Department of Information and Communications Engineering, Universitat Autònoma de Barcelona (Spain). He received the B.Sc., B.E., M.Sc., and Ph.D. degrees in Computer Science from the Universitat Autònoma de Barcelona in 2000, 2002, 2004, and 2006, respectively. Since 2002 he has been consecutively funded with doctoral and postdoctoral fellowships in competitive calls. From 2007 to 2009 he carried out two research stages of one year each with the group of David Taubman, at the University of New South Wales (Australia), and with the group of Michael Marcellin, at the University of Arizona (USA). He develops and maintains BOI, a free-source JPEG2000 implementation. In 2000 and 2002 he received two awards of Bachelor given to the first students of the promotion. In 2006 he was awarded with a free software mention from the Catalan Government for the development of BOI. In 2013, he was awarded with a distinguished R-Letter given by the IEEE Communications Society for a paper co-authored with Michael Marcellin. He is reviewer for various magazines and symposiums and has authored numerous papers in the top journals and conferences of his field. His research interests include a wide range of image coding topics, including highly scalable image and video coding systems, rate-distortion optimization techniques, parallel architectures for image and video coding, distortion estimation, embedded quantization, and interactive image and video transmission, among others.