Electronic Letters on Computer Vision and Image Analysis 5(4):24-46, 2006

A novel approach to sparse histogram image lossless compression using JPEG2000

Marco Aguzzi* and Maria Grazia Albanesi*

* *Dipartimento di Informatica e Sistemistica, Università degli Studi di Pavia, Via Ferrata 1, Pavia, Italy*

Received 8 June 2005; accepted 28 July 2006

Abstract

In this paper a novel approach to the compression of sparse histogram images is proposed. First, we define a sparsity index which gives hints on the relationship between the mathematical concept of matrix sparsity and the visual information of pixel distribution. We use this index to better understand the scope of our approach and its preferred field of applicability, and to evaluate the performance. We present two algorithms which modify one of the coding steps of the JPEG2000 standard for lossless image compression. A theoretical study of the gain referring to the standard is given. Experimental results on well standardized images of the literature confirm the expectations, especially for high sparse images.

Key Words: Image Compression, JPEG 2000, Sparse Histogram

1 Introduction

The JPEG2000 [1–8] started its standard formalization in 1997 and became an ISO standard in late 2000, confirming itself as the new reference point for researches in the field of still image compression. Among several innovations, the use of wavelets instead of DCT (Discrete Cosine Transform) (first appeared on [9], based on Fourier analysis, which was used by JPEG [10, 11] standard) allows multiresolution processing, preservation of spatial locality information and adaptivity on the image content. JPEG2000 obtains better results in terms of compression ratios, image quality and flexibility according to user demands. The JPEG2000 coding scheme is quite similar in philosophy to the EZW [12] and SPIHT [13] algorithms, even if it uses different data structures. Furthermore, the architectural design of the JPEG2000 allows several degrees of freedom aimed at tailoring the processing toward specific needs.

In literature, there are several proposal of approaches that modify only the coder (thus preserving the standard compatibility [14–26]) or both the coder and the decoder. Our approach belongs to the second group and the gain over the standard will be motivated from a theoretical point of view (by definition of a gain function, see Section 4.2) and by the experimental results. A very brief and preliminary version of our algorithms has been described in [27]; here we give a fully review of new experiments to validate our approach, and we add new considerations about comparison to JPEG 2000, PNG, and JPEG-LS. In fact we present concepts, theory and results about two novel algorithms which can be used to enhance performance (in terms of compression ratio) of

Correspondence to: <marco.aguzzi@unipv.it>

Recommended for acceptance by X. Otazu and J. Serra
ELCVIA ISSN:1577-5097
Published by Computer Vision Center / Universitat Autònoma de Barcelona, Barcelona, Spain

lossless compression of images [28–32] coded with JPEG2000 standard. In particular, the proposed algorithms modify the compression chain in the bit-plane encoding step, allowing adaptations particularly suited for sparse histogram images, for which the gain is at its best. For completeness sake, a brief introduction of JPEG2000 is given in Section 2; Section 3 gives the basis to understand the theory of the work through an original definition of histogram sparsity concept. Section 4 describes the two algorithms. Experimental results are fully reported in Section 4.1.1 and 4.1.3 for the first proposal, and in Section 4.2.2 for the second one. Conclusions (Section 5) and suggestions for further works end the paper.

2 JPEG2000 Overview

In order to fully understand the proposed algorithms for a modified JPEG2000 encoder, a brief introduction of the standard is presented: a block system level description of the compression chain is given (for further details the reader can refer to the corresponding literature [2, 4, 7]). In Figure 1 a black-box scheme of JPEG2000 compression chain is depicted, showing at which point our algorithms introduce modifications.

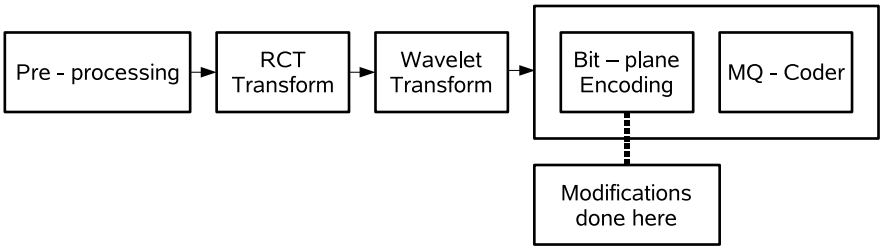


Figure 1: JPEG2000 compression chain at a system level: in the proposed algorithms, modifications occur in bit-plane encoding block.

2.1 Preprocessing

By referring to Figure 1, the first stage of JPEG2000 compression chain involves a resampling of pixel values to obtain symmetry around zero: if s_i is the i^{th} sample of an input image (of size $D = V * H$, where V and H are the vertical and horizontal dimensions, respectively), and bps is the bit per sample rate, this is done through relation in Equation 1:

$$\forall i \in [0, D-1] \; s_i^\star = s_i - 2^{bps-1} \tag{1}$$

so that $\forall i \in [0, D-1]$ the old samples s_i are in the range $[0, 2^{bps}-1]$ while the new ones $s_i^\star$ in $[-2^{bps-1}, 2^{bps-1}-1]$. This operation enhances the decorrelation among samples and helps in making statistical assumption on the sample distribution.

2.2 Wavelet Transform

The choice of the wavelet kernel [12, 33–37] is one of the points that makes JPEG2000 different from the standard JPEG. This kind of transform can produce highly decorrelated coefficients, although preserving spatial locality, which makes them the ideal input for an image compression algorithm. The advantages that JPEG2000 takes from wavelet transform are: a) the coefficients produced are resolution correlated, thus enabling the use of multiresolution analysis, and b) a smooth area of an image will produce small magnitude coefficients, while a sharp shaped one will give high magnitude ones; the second property allows an adaptive coding based on spatial locality.

The two wavelet transforms used in JPEG2000 coding chain are LeGall 5/3 and Daubechies 9/7 [38]. The characteristics of the former (good approximation property and the shortest biorthogonal filters available) lead

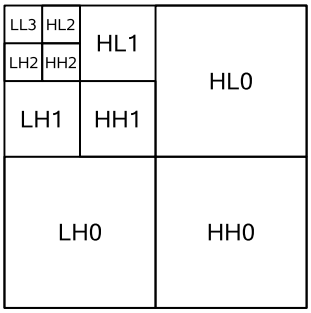


Figure 2: Wavelet decomposition of image I, organized in subbands.

it to be used for lossless compression, while the latter (good orthogonality, despite of a non optimal smoothness) is more suitable for the lossy one. In order to reduce computational load, instead of using the original Mallat multiresolution scheme [39] the wavelet decomposition is carried out using a lifting scheme [40]. Starting from the original image I, the coefficients produced by the multiresolution computation of the wavelet transform are organized in bands (conventionally referred to as LL, HL, LH and HH) and levels, depicted in Figure 2.

2.3 Coefficient coding

The coefficient coding is composed into several functional units, described in the current section (Figure 3). The first one, called Tier-1, processes wavelet coefficients and generates a bitstream; the second one, called

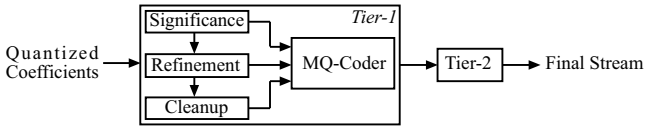


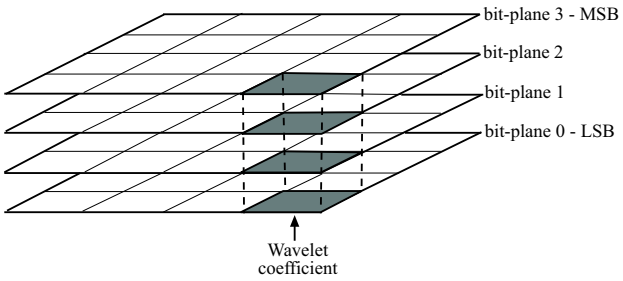
Figure 3: The coefficient coding block of JPEG2000 compression chain.

Tier-2, organizes the bitstream according to user specifications. In Tier-1, after the wavelet coefficients have been computed, the whole subsequent processing is organized into non overlapping code-blocks of (typically 64 by 64) coefficients; each code block is passed to the coding chain independently. This creates the input for *EBCOT* [41] algorithm. The main idea behind this algorithm is to code coefficients "by difference" between two sets: significant and non significant ones, the big effort is toward localizing the non significant (which occurs with a higher probability) area of blocks: in this way the position of significant coefficients is easily determined. This fundamental idea of EZW is implemented in JPEG2000 by bit-plane coding (Figure 4(a)) in which a particular scan order of coefficient inside each code block is applied (Figure 4(b)).

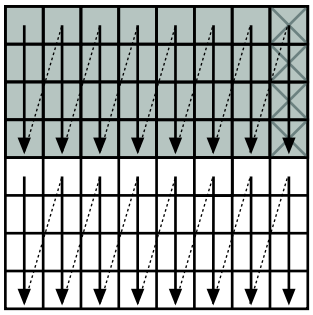
The implementation is slightly different from the original EZW because thresholding is abandoned and the significance of a coefficient is determined in bit-plane scanning according to the bare value of the bit and the relationship with neighborhood coefficients. The bit-plane scan unit is called *stripe*, which is four bits high and as wide as the code block (Figure 4(b)). For simplicity, we call a single column of a stripe a *quadruplet*: this term will be used further in the paper. The bit-plane coding is implemented in a sequence of three passes called *significance*, *refinement* and *cleanup*: each sequence is executed on every bit plane, starting from the MSB to the LSB.

2.3.1 Significance pass

In this pass each stripe is scanned, bit by bit, to decide if a coefficient is significant or not. For clarity sake, the symbols that will be used are presented in Table 1 (here, x and y identify the current position in the block). If



(a) A wavelet coefficient viewed over four different bit-planes.



(b) Two stripes are shown with scan order within a bit-plane in a code block. The first stripe has been shaded in dark gray, and a single quadruplet has been "crossed".

Figure 4: Organization and scan order of wavelet coefficients

Table 1: Symbol used in EBCOT summary.

Symbol	Meaning
$\sigma(x,y)$	The significance state matrix
$b(x,y)$	The current coefficient being processed
bp	The current bit-plane being processed
v	($b(x,y)$ and $1 << bp$) The current bit of coefficient $b(x,y)$ on bit-plane bp

the current coefficient $b(x,y)$ has not already been marked as significant ($\sigma(x,y) = 0$) and its neighborhood (Figure 5) contains at least one significant coefficient, the value v is conveyed to the MQ-coder (see Figure 3) and the corresponding $\sigma(x,y)$ is updated (one if v is equal to one, zero otherwise). If v is equal to one, the sign of the coefficient will be conveyed too. If the coefficient $b(x,y)$ has already been marked as significant ($\sigma(x,y) = 1$), the bit v will be processed in the refinement pass. If the current coefficient has not already been marked as significant ($\sigma(x,y) = 0$) and its neighborhood does not contain any significant coefficient, the cleanup pass will take care of it.

$b(x-1,y-1)$	$b(x,y-1)$	$b(x+1,y-1)$
$b(x-1,y)$	$\bullet$	$b(x+1,y)$
$b(x-1,y+1)$	$b(x,y+1)$	$b(x+1,y+1)$

Figure 5: The 8 connected neighborhood of a coefficient $b(x,y)$.

2.3.2 Refinement pass

This second pass is used to convey the magnitude of coefficients that have already been found significant ($\sigma(x,y) = 1$) in previous passes, conveying v.

2.3.3 Cleanup pass

This pass takes care of processing all the coefficients discarded by the two previous passes. It scans for every quadruplet: if all the four quadruplet coefficients have (σ=0) and their neighborhoods do not contain any significant coefficient and they do not become significant in the current bit-plane (all v are equal to zero) a bit zero is conveyed together with the corresponding context (this configuration has been called *non significant quadruplet*). If all the four quadruplet coefficients have $\sigma = 0$ and their neighborhoods do not contain any significant coefficient but at least one v value is equal to one, a string is conveyed to identify the position of the first $v = 1$ and the remaining coefficients are coded according to the standard significance pass policy. This particular configuration will be referred to as *half quadruplet*. If at least one of the four quadruplet coefficients have $\sigma = 1$, the whole quadruplet is coded as if it were on a significance pass.

When starting the chain on most significant bit plane, the significance and refinement pass have no effect on bitstream generation because the first actual σ matrix update is performed in the first cleanup pass.

2.4 MQ-coder

Referring to Figure 3, this part takes the values produced either by *significance*, *refinement* or the *cleanup* passes and generates the stream accordingly to a half context, half arithmetic coder. Instead of using a totally arithmetic coder [42], some previously defined probability context are used to empower the codeword production. The contexts have been defined for each pass according to probability model of the bitstream. For example, in the *significance* and *refinement* passes the choice of the context depends upon the corresponding band of the coefficient and the eight connected neighborhood configuration of the currently processed bit (a reference regarding the approach of MQ-coder can be found in [43]).

2.5 Tier-2 coder

Tier-2 coder is the second part of *EBCOT* algorithm. The bitstream produced by each code block from the previous step is now organized into layers: this is done with a sort of multiplexing and ordering the bitstreams associated to code blocks and bit-planes. A *layer* is a collection of some consecutive bit-plane coding passes from all code blocks in all subbands and in all components [1]. The quality of the image can be tuned in two ways: with the same levels of resolution but varying the number of layers, one can perceive an image in its original size with different level of degradation (typically in the form of blurring artifacts); with the same number of layers, but varying the levels of resolution, the image is perceived always at the same quality, but in different sizes, obviously smaller for less resolution levels.

The compressed data, now organized in layers, are then organized into packets, each of them composed by a header, containing important information about how the packet has been built, and a body. This kind of organization of the final codestream is done for a better control of the produced quality and for an easier parsing that will be done by a decoder.

3 Toward the proposed algorithms

The first of the two proposed algorithms makes modifications on the *significance* and *refinement* passes, while the second one works only on the *cleanup* pass; before explaining those in details, the concept of sparse histogram images is addressed in order to understand the approach of the work. The aim of this study is to find an intuitive relation between the term “sparse histogram” and its visual meaning; therefore, we propose a novel sparsity index definition. For clarity, the index is composed by two terms, which are defined as follows. Let us take the set T defined as:

$$T \equiv \{t \mid t \in [0, 2^{bps} - 1]\} \tag{2}$$

where bps is the bit per sample of the image and $\sharp T$ will denote the cardinality of T. In this work, 256 gray levels images are taken into account, so bps will be normally set to 8. We define the function $H(t)$ as the histogram of a given image, that is, for each image tone t, $H(t)$ is equal to the number of occurrences of the tone t in the image. Let's define a threshold th as the mean value of a normalized version of $H(t)$ (the normalization of $H(t)$ is done on the maximum value of $H(t)$, so that for every image the histogram values are in the range $[0, 1]$), which visually is a discriminant between an image having some gray tones more prevalent than other, or having all the gray tones playing more or less the same role. Basing upon the histogram function, we define a sparsity index as the sum of two terms:

$$\mathcal{I} = A + B \tag{3}$$

The computation of A and B is explained in the following paragraphs.

3.1 The sparsity index: first term computation

The first term of the sparsity index is defined as follows: from Equation 2, taking into account the set

$$T' \equiv \{t \mid H(t) < th\} \tag{4}$$

we define A as

$$A = \frac{\sharp T'}{\sharp T} \tag{5}$$

that is, the ratio between the number of image tones t that have a histogram value below that threshold and the total number of image tones.

3.2 The sparsity index: second term computation

Starting from $H(t)$, we define m_1 (Figure 6(a)) as

$$m_1 = \max(H(t)) \tag{6}$$

and $\overline{t}$ as

$$\overline{t} \quad | \quad H(\overline{t}) = m_1 \tag{7}$$

From 6 and 7, we modify $H(t)$ to produce $H_\star$ defined as:

$$H_\star = \begin{cases} H(t) & \forall t \neq \overline{t} \\ 0 & \text{if } t = \overline{t} \end{cases} \tag{8}$$

The same operations are performed on $H_\star$, so (Figure 6(b)):

$$m_2 = \max(H_\star(t)) \tag{9}$$

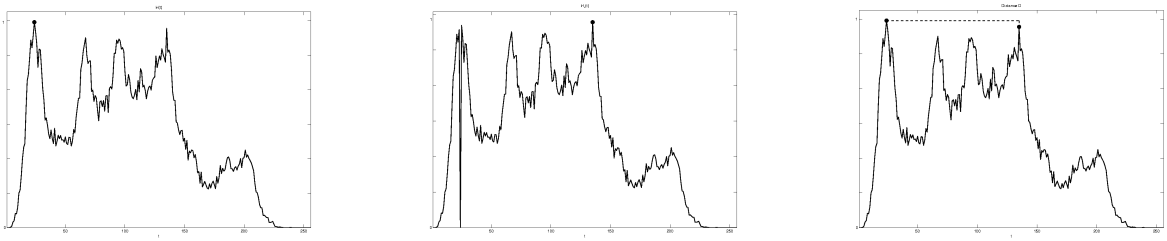
and $\overline{\overline{t}}$

$$\overline{\overline{t}} \quad | \quad H_\star(\overline{\overline{t}}) = m_2 \tag{10}$$

If there is an ambiguity in the choice of $\overline{t}$ and $\overline{\overline{t}}$, we consider the values of $\overline{t}$ and $\overline{\overline{t}}$ that minimize the distance $|\overline{t} - \overline{\overline{t}}|$ (Figure 6(c)). So, the second term (B) is defined as follows:

$$B = \frac{|\overline{t} - \overline{\overline{t}}|}{\sharp T - 1} \tag{11}$$

In order to better highlight the sparsity index boundaries, two synthetic images have been created: in image 7(a) the number of pixels is equal to the number of gray scale tones, in image 7(b) only the two most distant tones (e.g. black and white) have been used.



(a) The image histogram $H(t)$ with m_1 highlighted by a rounded dot.

(b) The image histogram $H_\star(t)$ with m_2 highlighted by a rounded dot.

(c) The image histogram and the distance $|\bar{t} - \bar{\bar{t}}|$ indicated by the dashed line.

Figure 6: The steps of the computation of the second term B as in Equation 11.

(a) "Smooth"

(b) "Hard"

Figure 7: Test images for sparsity index boundaries evaluation. (a) sparsity index is equal to 0 and (b) is equal to 2.

When the sparsity index is calculated for the first image (7(a)), called "Smooth", its histogram is a constant function of value 1 over all the image tones: so we have

$$th = 1 \text{ (the only histogram value)} \tag{12}$$

and, from Equation 5

$$A = \frac{0}{\sharp T} = 0 \text{ (no tone is below the threshold)} \tag{13}$$

For B term, we have (see Equation 11):

$$B = \frac{1}{\sharp T - 1} \text{ (each maximum is one-tone far from its neighbor)} \tag{14}$$

therefore, from Eqs. 3, 13, and 14 we have

$$\mathcal{I} = 0 + \frac{1}{\sharp T - 1} = 0.003922 \tag{15}$$

Taking into account the second image (7(b)), called "Hard", its histogram consists of only two peaks at the edge of the graph ($t = 0$ and $t = \sharp T - 1$), both of value $\sharp T/2$ (this height of the peaks is valid only in this case for a "Hard" image measuring 16 by 16 pixels; in the most general case, for a V*H image, the peak height is (V*H)/2). For image "Hard" we have (using the normalized version of $H(t)$, the two $\sharp T/2$ peaks normalize to 1):

$$th = (1 + 0 + \ldots + 0 + 1)\,\frac{1}{\sharp T} = \frac{2}{\sharp T} \tag{16}$$

and (Equation 5):

$$A = \frac{\sharp T - 2}{\sharp T} \tag{17}$$

(all the tones but the two at the very edge of the histogram are below the threshold, which is small but strictly positive.) For B term, we have (Equation 11):

$$B = \frac{|0 - (\sharp T - 1)|}{\sharp T - 1} \text{ (the two tones are at the maximum distance)} \tag{18}$$

therefore, from Eqs. 3, 17, and 18 we have

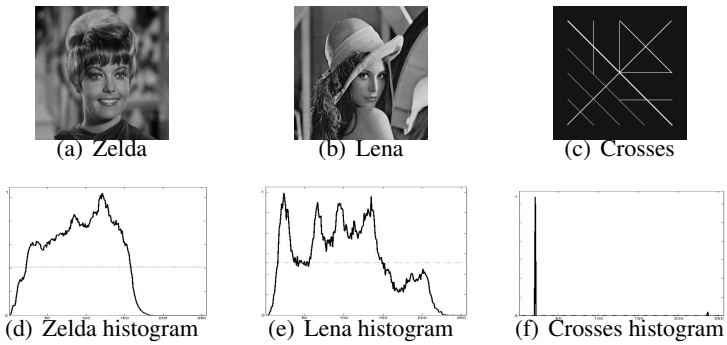
$$\mathcal{I} = \frac{\sharp T - 2}{\sharp T} + 1 = 1.992188 \tag{19}$$

In Table 2 is shown a list of images from literature in increasing order, according to the sparsity index. In

Table 2: Test images in increasing order according to the sparsity index (Equation 3).

Name	zelda	mandrill	bride	bird	peppers	camera	barb	mehead	mountain	boat	lena	frog	squares	montage	slope	fractalators	text	library-1	circles	crosses	fpf
Id	21	13	4	2	17	5	1	14	16	3	11	10	19	15	18	9	20	12	6	7	8
Unique Colors	187	226	256	145	230	247	221	256	110	224	230	102	4	251	248	6	2	221	4	2	2
$\mathcal{I}$	0.47	0.52	0.55	0.61	0.65	0.66	0.66	0.72	0.72	0.75	0.96	0.97	1.18	1.61	1.67	1.69	1.69	1.69	1.83	1.84	1.84

order to give the reader a visual feedback for sparsity index $\mathcal{I}$ we present (Figure 8), three examples from the tested images, showing also, with a solid line, the corresponding threshold th: the first one is *zelda* (Figure 8(a)), the least sparse ($\mathcal{I}$=0.473), with its histogram, then an intermediate case with *lena* (Figure 8(b)) ($\mathcal{I}$=0.955), and finally, the most sparse case with *crosses* (Figure 8(c)) ($\mathcal{I}$=1.835).



(a) Zelda (b) Lena (c) Crosses

(d) Zelda histogram (e) Lena histogram (f) Crosses histogram

Figure 8: Reading the figures columnwise, for each column the couple picture-histogram is presented.

4 The two proposals

4.1 First algorithm: Stripe lengthening

The first algorithm proposed in this paper addresses the first part of *Tier-1* encoder, the *significance* and the *refinement* passes: recalling that bit coefficients are encoded in groups of four by four (the quadruplets), the aim is to reduce changing context overhead by making quadruplets (and, consequently, stripes) longer; therefore, we call it "stripe lengthening". We expect that, by lengthening the stripe, the number of context changes is reduced; this is particularly true for images with a higher sparseness, because their characteristics are likely to change less often than other kind of images. For this reason, we expect better result for images with a higher index $\mathcal{I}$. The lengthening is the same in *significance* and *refinement* pass (while the *cleanup* pass is performed with the original length 4): in fact, it would not be coherent to group coefficients in two different ways in the

same encoding process. The reason why we choose to leave the cleanup pass with its original stripe length of 4 bits is that we prefer to have the two modifications distinguished; moreover, it does not make sense using longer stripe in the cleanup pass when the second modification is active: the longer the stripe, the shorter the quadruplet series.

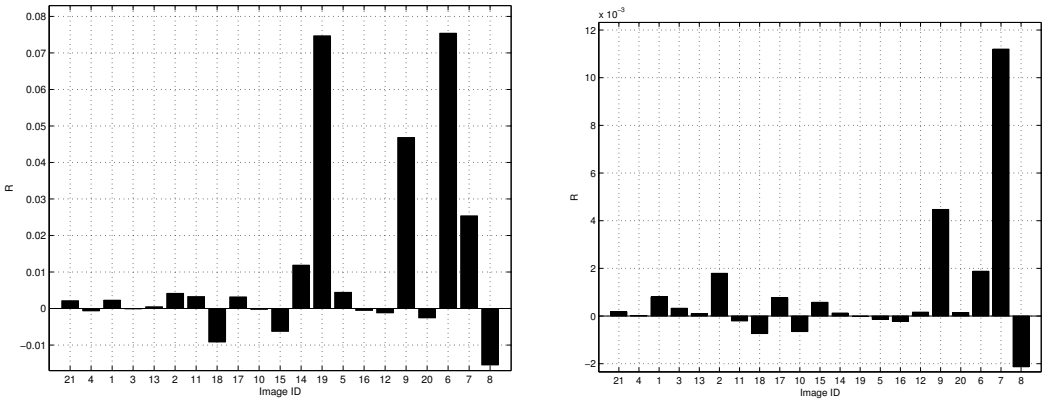
In order to perform the modification of the algorithm to the encoding process, two variables (pointers) have been used to keep track of the current quadruplets and to identify the beginning of the next one and they are properly initialized according to the new stripe length. The original JPEG2000 quadruplet length was fixed to 4 (hence the name). In our algorithm, we test the encoder over various quadruplet lengths; the range varies from 5 to the block height, but in the experimental results here reported only significant values are considered, namely 8, 16 and 32. We have implemented the algorithm in a way that it is fully compatible with the source code Jasper [44] in which the piece of code regarding each quadruplet bit was repeated four times. As we have parameterized the length, we embodied such code in a counter based loop which cycles until the quadruplet length counter becomes zero. The implementation of the stripe lengthening requires a corresponding modification of the decoder structure, in order to couple correctly coder and decoder. Various combinations of resolution levels and stripe lengths have been evaluated; first we present the results of the experiments for single combinations (*stripe length-resolution level* - Figure 9(a)-11(b)), then three overall graphs will be shown in order to give a general overview of the combinations.

4.1.1 Single combination experiments

The bar graphs (see Figure 9(a) and 9(b)) have on the X-axis an image identifier (for space reason it could not be possible to put the whole image name (see Table 2)), and on the Y-axis a ratio R defined as

$$R = 1 - \frac{P}{O} \tag{20}$$

where P is the encoded file size obtained using the encoder modified by this proposal and O is the size regarding the original JPEG2000 encoder. The formula in Equation 20 has been adopted in order to give an immediate visual feedback: if the bar is above the ordinate of value 0, the modified encoder works better (the file is smaller) than the original one, the contrary otherwise. In Figure 9 graphs relative to 8 bit long stripes with 2 and 6 levels of resolution are presented.



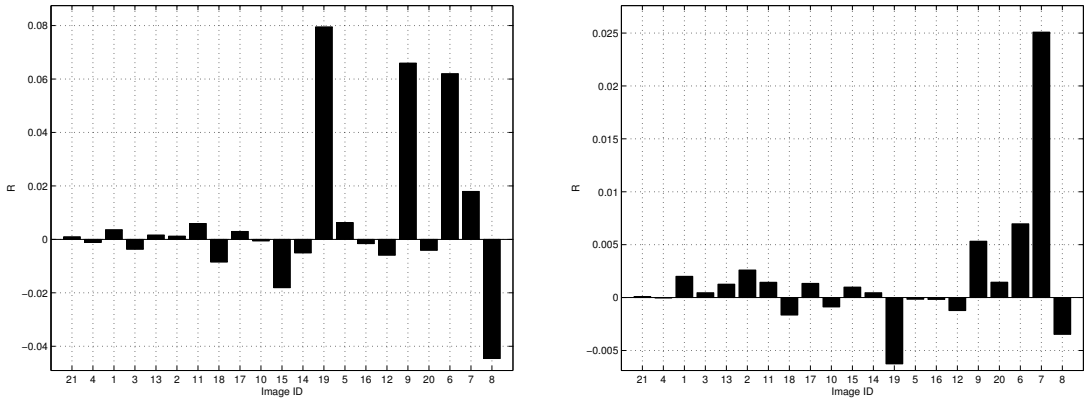
(a) Two levels of resolution. Best gain about 8% (b) Six levels of resolution. Best gain about 1.1%

Figure 9: Single combination graphs for stripe length equal to 8 bit for each image, the R value (Equation 20) shows the gain of our coder compared to JPEG2000. Images are ordered increasingly according to sparsity.

The most remarkable results (between 7% and 8% gain) are obtained at 2 levels of resolution, while, using 6 levels, the situation becomes worse (around 1.1%) and just few images (always the sparse histogram ones) can perform well. The performance decay when the number of wavelet decomposition levels increase because the

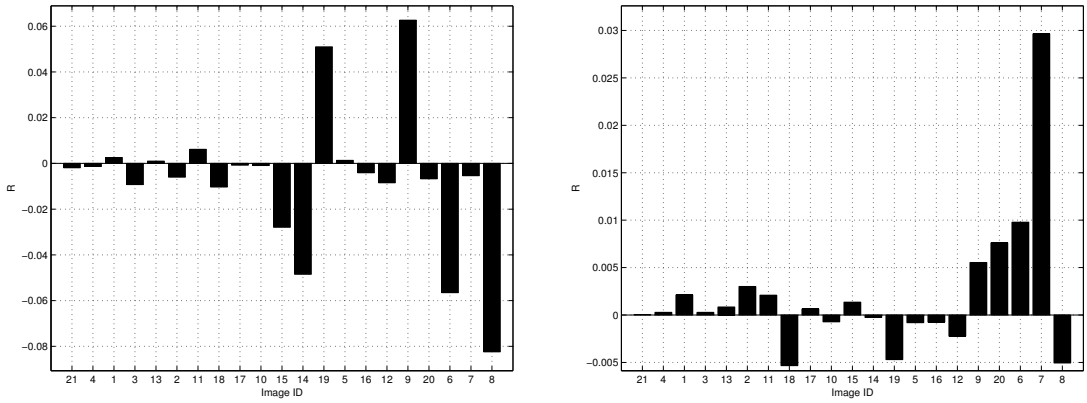
code blocks will be smaller, as dictated by the wavelet transform. Therefore, stripe lengthening will work in a smaller number of areas (if the stripe length is greater than the code block, the stripe is truncated to the code block dimension) and its application will have less relevance.

In the subsequent Figures (10 and 11) plots are as follows: for stripes length of 16 and 32 bits two graphs, regarding 2 and 6 levels of resolutions, are depicted. Therefore, the algorithm introduces enhancement at lower levels of resolutions, where JPEG2000 has a lower compression ratio.



(a) Two levels of resolution. Best Gain about 8% (b) Six levels of resolution. Best Gain about 2.5%

Figure 10: Single combination graphs for stripe lengths equal to 16 bit: for each image, the R value (Equation 20) shows the gain of our coder compared to JPEG2000. Images are ordered increasingly according to sparsity.



(a) Two levels of resolution. Best Gain about 6% (b) Six levels of resolution. Best Gain about 3%

Figure 11: Single combination graphs for stripe length equal to 32 bit: for each image, the R value (Equation 20) shows the gain of our coder compared to JPEG2000.

4.1.2 Overall graphs

As an overall example, the values of R obtained at the various levels of resolution are depicted in Figures 12-14. Besides, the mean value of R, computed over 3 - 6 levels of resolution, is reported in the graph. In the graph of Figure 13 the best results are obtained; from the next one (Figure 14) there is no more reason to further increase the stripe length because the ratio become significantly less than zero, meaning that the original coder is performing better than the modified one. In Figures 15(a) and 15(b) overall mean graphs are proposed. In the first one the mean is over the stripes, while in the second the mean is over the resolution levels.

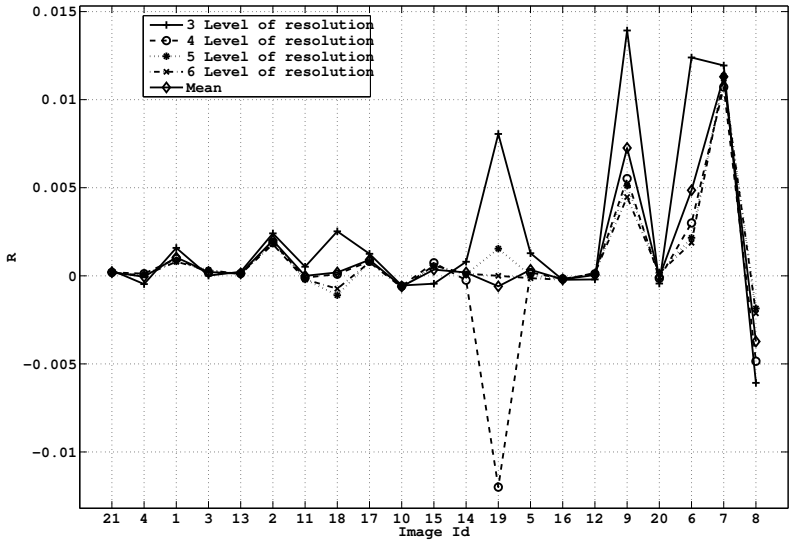


Figure 12: Overall graph for stripe length equal to 8 bit: for each image, the R value (Equation 20) shows the gain of our coder compared to JPEG2000. The best gain is in term of 8% of the file size coded with standard JPEG2000. The images are sorted by their sparsity value.

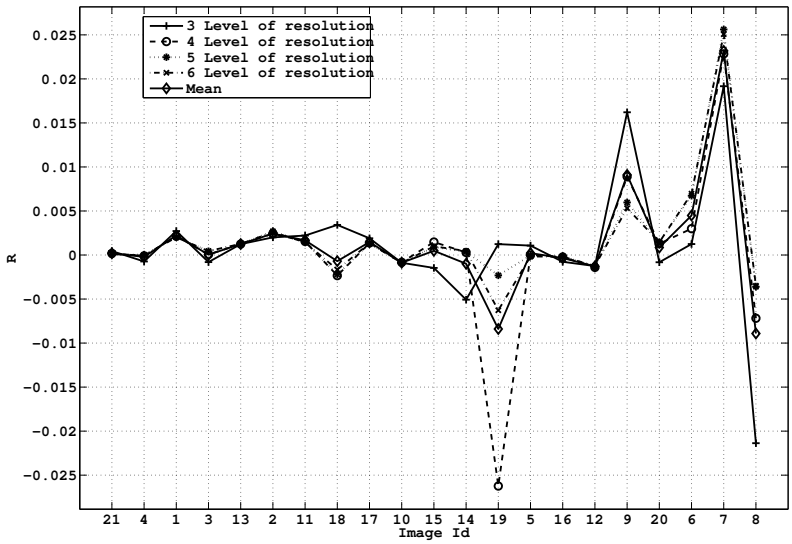


Figure 13: Overall graph for stripe length equal to 16 bit: for each image, the R value (Equation 20) shows the gain of our coder compared to JPEG2000. The best gain is in term of 8% of the file size coded with standard JPEG2000. The images are sorted by their sparsity value.

From the several experiments one thing is clearly noticeable: almost all of the image files with a high sparsity index are significantly smaller when processed by our algorithm at a low level of resolution: the behavior does not follow the trend of JPEG2000 encoder, which obtains better results with more resolution levels. It is otherwise important to notice that in each graph the best performances are always given by images with a high degree of sparseness, in accordance to what we supposed at the beginning of the work. Another approach has

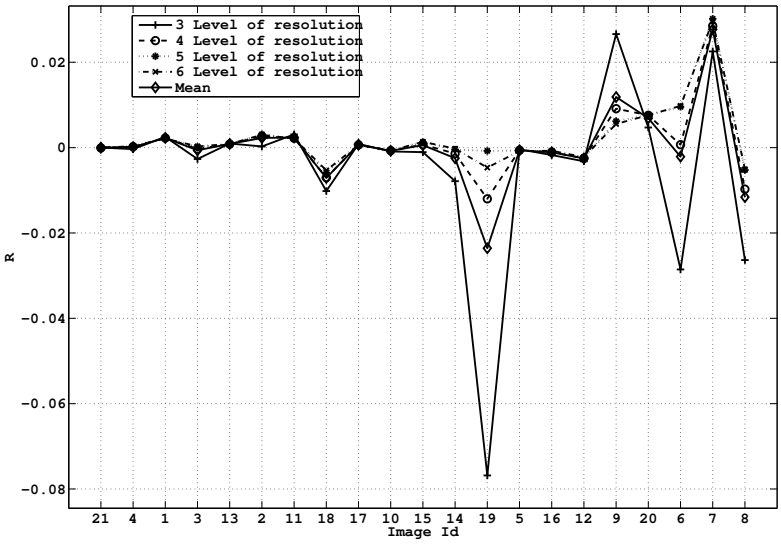


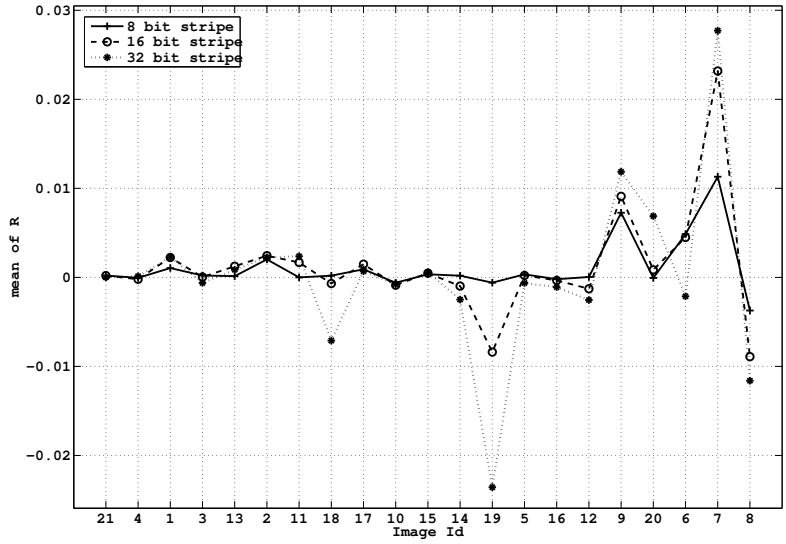
Figure 14: Overall graph for stripes height equal to 32 bit: for each image, the R value (Equation 20) shows the gain of our coder compared to JPEG2000. The best gain is in term of 6% of the file size coded with standard JPEG2000. The images are sorted by their sparsity value.

been followed in the second proposed algorithm, which is almost resolution - independent, but still preserving the biased behavior toward sparse histogram images. Observing that there is a better margin for improvement when the resolution level is low but standard JPEG2000 level of resolution is 5, to justify the use of less levels of resolution and the stripe lengthening, we compare the execution times of standard algorithm with no stripe lengthening and 5 levels of resolution to the use of 8 bit stripe lengthening and 2 levels of resolution. As Figure 16(a) shows, despite the fact that this comparison is not so favorable for the modified algorithm in terms of compression, execution times are always lower or at most equal to the standard algorithm. If we compare the size, our modification is better for sparse histogram images (Figure 16(b).)

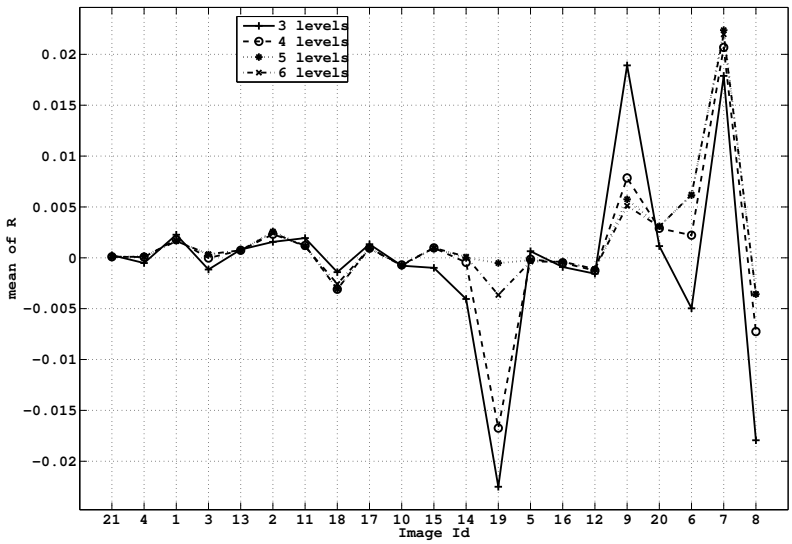
4.1.3 Comparison to other compression algorithms

Several articles in literature [6, 45] show that JPEG2000 now is one of the most promising and well performing format for compression of still images; even if some formats perform better in some cases, JPEG2000 characteristics and versatility justify our choice of having JPEG2000 as our main comparison term. Moreover, as we consider lossless compression only, we consider also JPEG-LS [46] as the "optimum" comparison term. However, we want to give an idea of possible comparisons to other approaches by taking also into consideration another compression algorithm: the choice has been directed to PNG [47] format, because it is one of the most recent image format that has achieved popularity on the Web. In this section, a comparison table (Table 3) among the obtained compression ratios of Standard JPEG2000, stripe lengthening, JPEG-LS, and the PNG format is given. From the experimental results (summarized in Table 3), we point out that the performance of our algorithm, when applied to sparse histogram images (from *text* to *fpf*), are between standard JPEG2000 and JPEG-LS. It is interesting to study the effect of multiresolution and stripe length choice over the compression ratio; so, in order to find the best stripe length and level of resolution, we define an improving factor F (which is a function of the stripe length and the level of resolution) computed over the entire set of 21 images:

$$F(stripe, level) = \sum_{i=1}^{21} \frac{size_{standard} - size_{modified}^{stripe,level}}{size_{uncompressed}} \tag{21}$$



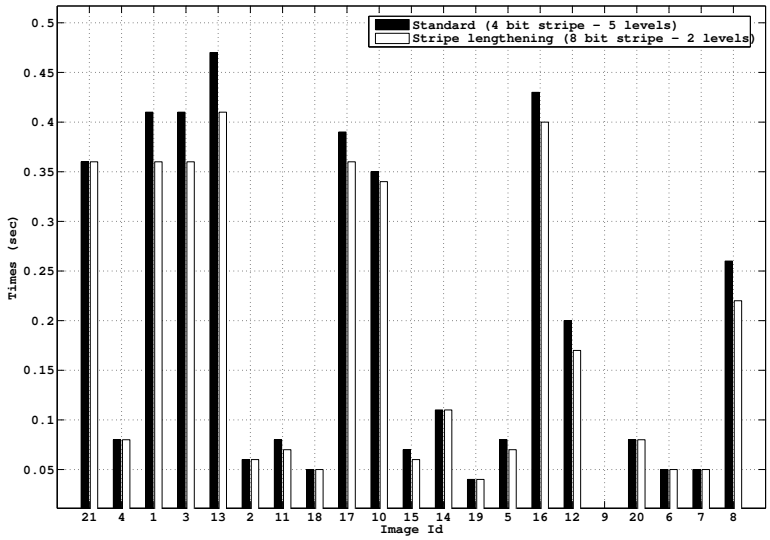
(a) Mean over stripe length.



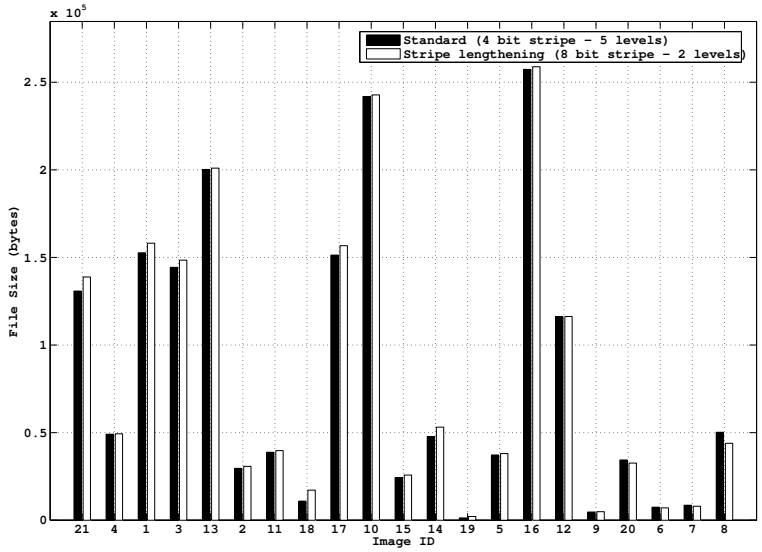
(b) Mean over resolution levels.

Figure 15: On top, the mean of R value is computed over different stripe lengths of 8, 16, and 32 pixels with 3-6 Level of resolution; on bottom, the mean is computed over 3 - 6 Resolution Level with a different stripe lengths of 8, 16, and 32 pixels.

From the computation of all the reasonable values of F, we obtained the best result for 32 bit stripe length and 5 levels of resolution. We underline that the factor F is a objective measure only, and does not take into account whatever image characteristic. F computation mediates between highly positive and highly negative results in each image; therefore, there is no contradiction between this result and the ones presented in Figures 12-14.



(a) Time comparison. The computation time is always lower (at most equal) than the one obtained with the standard encoder, and this behavior is more prominent in the high sparsity region.



(b) Size comparison. Our algorithm works better on images with a high sparsity index.

Figure 16: Comparison between a 5 level standard JPEG2000 against a 2 level (stripe 8) modified algorithm.

4.2 Second algorithm: SuperRLC

4.2.1 Theoretical considerations

The second proposal focuses on the last part of Tier-1 coder, which is the *cleanup* pass. Recalling the concepts exposed in section 2.3.3, pointing at the fact that each completely non significant quadruplet is conveyed with one bit, a possible improvement is the following: instead of coding quadruplets with one bit each, we use a binary word for every set of consecutive non significant quadruplets. The binary word length is fixed throughout

Table 3: Comparison of the compression ratios obtained from lossless JPEG2000, stripe lengthening (8 bit stripe and 2 levels of resolution), JPEG-LS, and PNG.

Image	JPEG2000	8 bit stripe - 2 levels	PNG	JPEG-LS
zelda	2.00	1.89	1.88	2.00
bridge	1.34	1.33	1.35	1.38
barb	1.72	1.66	1.51	1.69
boat	1.82	1.77	1.73	1.88
mandrill	1.31	1.30	1.28	1.33
bird	2.22	2.13	2.02	2.31
lena	1.69	1.65	1.59	1.75
slope	6.02	3.82	5.60	5.09
peppers	1.73	1.67	1.65	1.78
frog	1.28	1.27	1.33	1.32
montage	2.70	2.54	2.72	2.94
mehead	3.78	3.39	4.02	4.89
squares	50.52	31.14	99.52	103.77
camera	1.76	1.72	1.71	1.86
mountain	1.19	1.19	1.21	1.25
library-1	1.40	1.40	1.56	1.57
fractalators	3.80	3.65	11.15	6.03
text	1.91	2.01	27.81	4.91
circles	8.84	9.30	35.49	52.46
crosses	7.67	8.21	32.48	20.77
fpf	7.50	8.57	43.18	64.10

the whole encoding process. The gain is higher as the number of consecutive non significant quadruplets increases. At the end of the *cleanup* pass, it is possible to identify on the current bit-plane different series of non significant quadruplet, each of them has a different length.

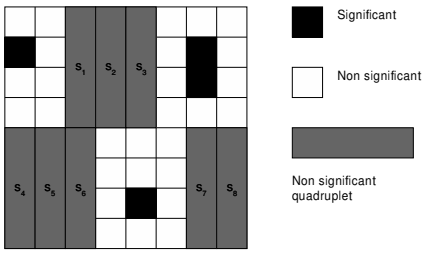


Figure 17: Different groups of quadruplets. From this block it can be figured out eight single quadruplets, s_1 through s_8. Here we have S_3 with $\sharp S_3$ equal to 2, and S_2 with $\sharp S_2$ equal to 1.

Before attempting a non trivial modification to the coder, it is important to see how this is going to affect its behavior (a good introduction on statistic has been taken from [48]). In this explanation the concept of non significant consecutive quadruplets will be addressed many times, so in the following we refer it to as *nscq*. A pictorial view of what is explained below is given in Figure 17. Let us define a set S_j containing all *nscq*s with the same length j, with j varying in the range $[1, j_{MAX}]$, where j_{MAX} is the longest group of *nscq* that could

be found. The length of a single *nscq* will be referred to as $l(nscq)$. Another index that has to be defined is k_{MAX}, which is the maximum number of *nscq* that can be sent to the MQ-coder in only one binary word, as explained later. So j_{MAX} depends on the current coefficients of the bit-plane in the running encoding process, while k_{MAX} (we have $k_{MAX} \leq j_{MAX}$) depends on the actual length of the binary word: referring to the length as W we have $k_{MAX} = 2^W - 1$. If the length of the current *nscq* is less than k_{MAX}, the *nscq* is entirely coded, otherwise it is treated as two shorter *nscq*. This process affects the values of $\sharp S$ (we use $\sharp S$ referring to $\sharp S_j \forall j$). The following equations describe the process in detail referring to how $\sharp S$ is modified. Using the remainder function

$$r(x, y) = \left(x - \left\lfloor \frac{x}{y} \right\rfloor y \right) \tag{22}$$

and setting p and q as

$$\begin{aligned} p &= r(l(nscq), k_{MAX}) \\ q &= \left\lfloor \frac{l(nscq)}{k_{MAX}} \right\rfloor \end{aligned} \tag{23}$$

the computation of $\sharp S$ comes to as described by Algorithm 1 (for clarity, $\sharp S$ values are expressed in percentage). That means that when a *nscq* with its length greater than k_{MAX} is found, the encoder performs in this way: it conveys q series k_{MAX} long and one series p long.

Algorithm 1 Cardinality computation. p and q are computed in Equation 23.

if $l(nscq) \leq k_{MAX}$ **then**
  $\sharp S_{l(nscq)} = \sharp S_{l(nscq)} + 1$
else
  $\sharp S_{k_{MAX}} = \sharp S_{k_{MAX}} + q$
  $\sharp S_p = \sharp S_p + 1$
end if

Applying the $\sharp S$ computation to Lena image, Figure 18 presents the results of the histogram of the $\sharp S$ computation. The explanation of a high value in the case $j = 31$ is quite straightforward: the last bin of the histogram counts all not significant sequences composed by 31 quadruplets. Hence, all sequences composed by more than 31 (in this case) quadruplets will score a point for the bin in the r position, and q point for the bin in the 31st position. Due to this reason, the last bin takes a value higher than the others.

Figure 20(a) shows the same histogram computation for image Crosses.

The $\sharp S$ refers to a particular choice of k_{MAX}, depending on W. Its computation ends the implementation of modified cleanup pass. However, as we are interested into an evaluation of the actual gain, (referring to the standard JPEG2000 cleanup pass) we can study the behavior of $\sharp S$ varying the upper limit of $l(nscq)$. Therefore, we have recalculated $\sharp S$ values for k varying from 1 to k_{MAX}. To do this a new $\sharp S$ (named $\sharp S'$) is computed in this way:

$$\forall k \leq k_{MAX} \; \sharp S'_k = \sum_{i=1}^{k_{MAX}} \sharp S_z \text{ where } z = \max(k_{MAX}, k \cdot i) \tag{24}$$

So now $\sharp S'$ contains the probability distribution as if j_{MAX} (that is, the maximum of $l(nscq)$) could assume all the values in the range 1,... k_{MAX}. Knowing from Equation 24 the entire probability distribution, it is interesting to give an estimation of the acquired gain versus the original coder. We defined a gain function G as

$$G(k, k_{MAX}) = \frac{k}{1 + \log_2(k_{MAX} + 1)} \tag{25}$$

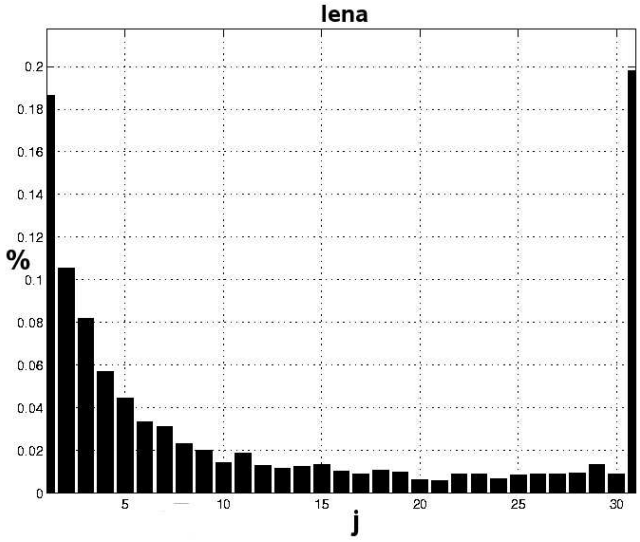


Figure 18: Cardinality computation for *nscq*s (Image Lena).

in which the numerator represents the bits used by the original coder and the denominator the bits used by the modified one. Function G (Equation 25) is used to weight the quadruplets probability distribution leading to (assuming that k_{MAX} is fixed):

$$\sharp S'_\star(k) = G(k)\sharp S'_k - 1 \tag{26}$$

In Figure 19 is presented the final graph that shows $\sharp S'_\star$ (Equation 26). The meaning of this function $\sharp S'_\star$ is an evaluation of $\sharp S'(k)$ weighted by the gain function G, thus giving an estimation of the gain toward JPEG2000 standard coder for each admissible value of consecutive quadruplet series. We point out that in Equation 26, the term -1 introduces a shift; therefore if values of $\sharp S'_\star$ (Figure 19) are greater than zero it means a positive gain. Apart from an initial irregular shape, the function in Figure 19 shows a stable gain for a maximum $l(nscq)$

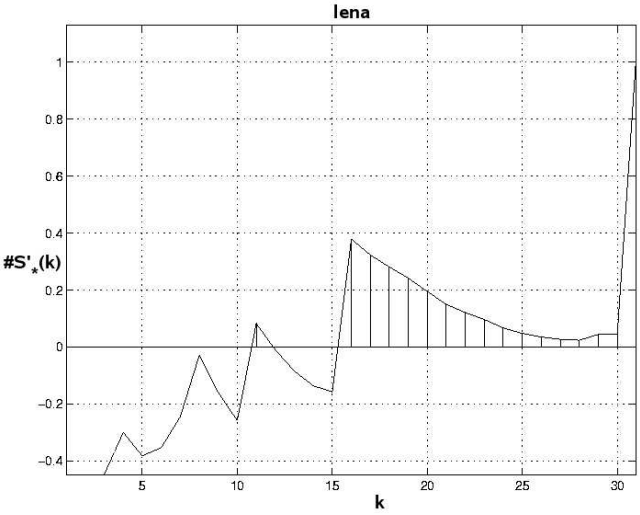


Figure 19: The weighted histogram $\sharp S'_\star(k)$ (Equation 26) shows the gain in terms of bit number with respect to the original coder JPEG2000 as a function of quadruplet series length (Image Lena).

greater than 16. From this point on, our algorithm performs better. This Figure refers to Lena, but the same behavior is confirmed (or even better) for the other images: as an example, in Figure 20(b) the function in

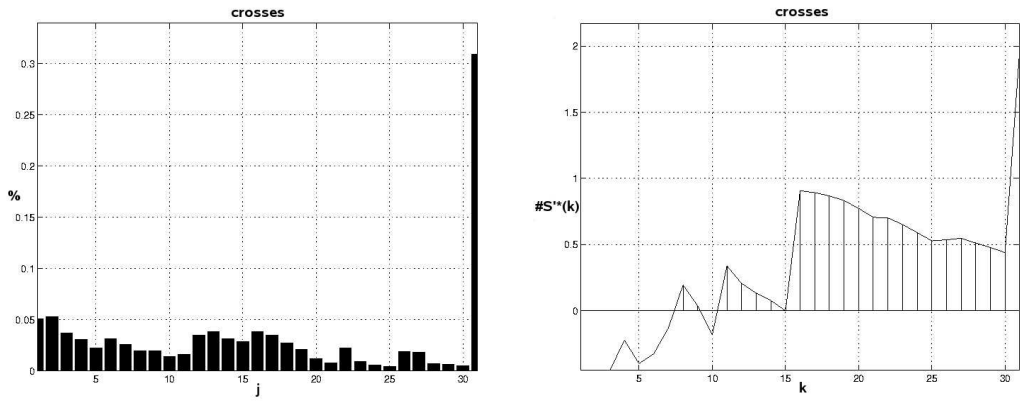
Equation 26 is plotted for image "crosses". For completeness, we have computed the gain for all the other test images. The final measure of how much using the *SuperRLC* is convenient is given by:

$$M = \sum_{i=1}^{k_{MAX}} \sharp S'_{\star_i} \tag{27}$$

Equation 27 computes in fact the integral function of the weighted histogram (Equation 26), measuring the positive gain area. M values are reported in Table 4. It is important to remind that this is not an overall gain, but it is just set up for bits sent by the cleanup pass to the final stage of Tier-1, the *MQ-coder*.

4.2.2 Comparisons with other images and the sparsity index

"Lena" is a well - known standard image for compression evaluation; however we present here the results about other interesting cases: as regards *Crosses* image, which has a sparsity index of 1.835 against the 0.955 of Lena, the tracking of $\sharp S$ during the encoding process is as depicted in Figure 20(a). The distribution follows



(a) Cardinality computation for *nscqs*.

(b) The weighted $\sharp S'_{\star}(k)$ (Equation 26) shows the gain in terms of bit number respect to the original coder JPEG2000 as a function of quadruplet series length

Figure 20: Cardinality computation and weighted $\sharp S'_{\star}(k)$ for image Crosses.

the sparsity: while in the *lena* case a smaller $\sharp S$ was more probable, this time the plot shows a clear need of a longer quadruplets series length ($\sharp S$). The graphic, depicted in Figure 20(b), overtakes more strongly the gain equal to one, and the measure given by Equation 27 is equal to 10.2347, against -0.3694 for Lena. In Table 4 all integral values associated to each image have been reported. Referring to Table 2 for sparsity indices, the values of M grow following, with few exceptions, the sparsity order. More generally, for all the images, there is always a value $\overline{k}$ for which our algorithm outperforms JPEG2000 $\forall k > \overline{k}$.

Table 4: Images and their M value (Equation 27)

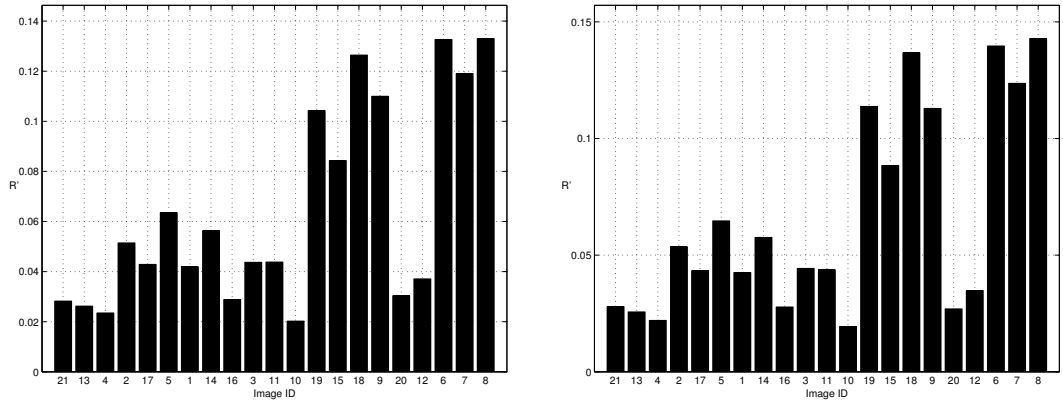
Name	crosses	circles	squares	fractalators	slope	fpf	montage	camera	mehead	bird	barb	lena	boat	peppers	zelda	library-1	mandrill	mountain	text	bridge	frog
Image ID	7	6	19	9	18	8	15	5	14	2	1	11	3	17	21	12	13	16	20	4	10
M	10.23	9.45	8.78	7.78	6.72	5.08	2.63	0.93	0.51	0.49	0.18	-0.37	-0.84	-0.89	-2.32	-2.89	-3.28	-3.32	-3.84	-4.05	-5.08

4.2.3 Overall comparison to JPEG2000

In this part, the computation of the ratio between all the bits (from significance, refinement, and cleanup) sent by original JPEG2000 and SuperRLC coder is plotted. As an example, two cases for W value equal to 5 and 6, respectively are shown. For each graph (Figure 21(a) and 21(b)), the X-axis will report the usual image identifier (refer to Table 2) and the Y-axis will report the quantity R' defined as:

$$R' = \frac{|BP - BO|}{D} \tag{28}$$

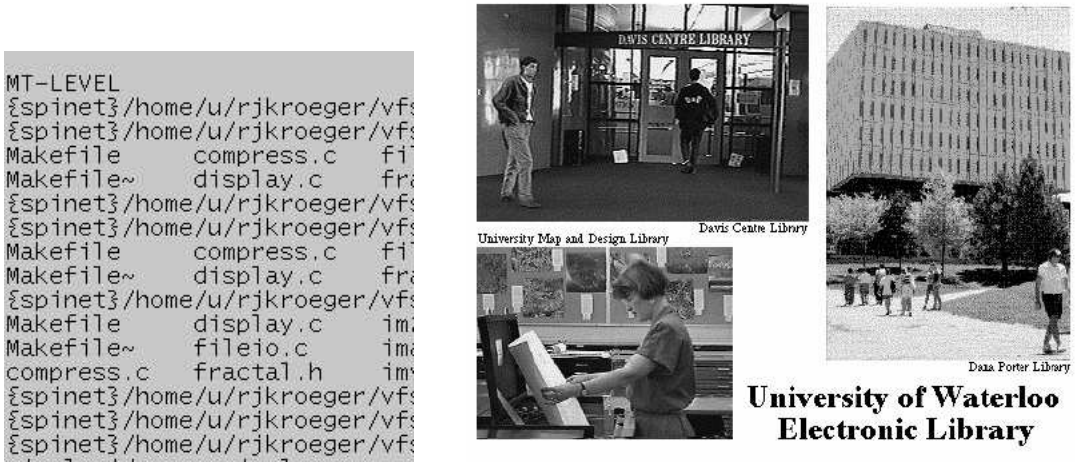
where BP are the bits sent by the Tier-1 encoder to the *MQ-coder* in the RLC method, BO refers to the JPEG2000 original encoder and D is the size of the original uncompressed image file expressed in bit. As it can be seen, the gain goes from a minimum of 2% to a maximum of 15%.



(a) code word of 5 bits (W=5). (b) code word of 6 bits (W=6).

Figure 21: Ratio between the bits sent by the original and modified encoder.

For these values of W, the experiments show always a positive gain, which is more impressive for sparse histogram images. The only really relevant exceptions are for images 12 and 20 (*text* and *library-1*, respectively), which do not follow the behavior dictated by their sparseness; as it could be clearly seen (Figure 22(a) and 22(b)), these two images have nothing in common with the other images with sparse histogram, that is, their



(a) The *text* image. (b) The *library-1* image.

Figure 22: Figures for which the behavior does not follow the general trend.

structure is so significantly different (a text document and a compound image) which justifies their anomalous behavior. We underline the fact that also image 8 is a text image, but it performs quite well. Moreover, the simple fact that an image represents a text is not discriminant. The reason is that image 8 is really a sparse image with a constant white background and a foreground (the text) which is spatially distributed in a very sparse way. We mean that in the case of image 8, "sparsity" is not only on grey level distribution, but also on the geometrical disposition inside the image. This does not hold for the second text image (12), where the background is not constant and the text is an automated list. By carefully observing the plots of Figure 21, a further comment about image 10 (frog) is mandatory. Image 10, despite its index $\mathcal{I}$, has a value of R' (Equation 28) significantly smaller than image 11 and 19, and we explain this fact by observing that the "frog" image is a highly textured one, because the subject (the animal) is so close to the camera that the spatial disposition of pixels is more similar to a texture than a tipical object on a background.

We prefer to consider R' (Equation 28) as a performance index, rather that comparing the bare file sizes at the end of the entire compress chain, because: a) we are interested into exploring the positive effects of our modifications on the significance, refinement, and cleanup group (where they effectively occur); b) we want to exclude any possible effects of the arithmetic coder, which follows in the chain.

4.2.4 Technical details

The project has been developed on a Celeron 1.5Ghz, 512Mb RAM, using Microsoft Visual C [49] for JPEG2000 codec and Matlab [50] for presenting the results in a convenient way. The codec is Jasper version from 1.400 to 1.600, of which a guide can be found in Jasper User Manual [44] and implements a JPEG2000 coder fully reviewed by the creator of Jasper M.D. Adams [5].

5 Conclusions

In this paper, we have presented two algorithms for modifying all the three passes of JPEG2000 Tier-1 coder, that is significance, refinement, and cleanup. Theoretical studies have been reported in order to classify the images according to the concept of sparsity and to compute the saving of conveyed bits. The best results are for high sparsity images: assuming the spatial preservation of wavelet coefficients, it has come to mind that longer stripe would have grouped better the (scarce) significant zones in this kind of images. The first algorithm, stripe lengthening, shows significant gain at low levels of resolution on the overall file size respect to the standard. The second algorithm, *SuperRLC*, has the advantage of being independent on the levels of resolution; experiments confirm that it gives a relevant gain on bits conveyed to the MQ-coder respect to the standard JPEG2000. Future works will regard a modification of the arithmetic coder in order to adapt the probability context to the statistics of the new conveyed symbols. We would like to use the same approach as [51] to extend the arithmetic coder contexts in order to evaluate if it is possible to capture the efficiency of the new bitstream generated by our algorithm.

References

[1] M. Rabbani and D. Santa Cruz, "The JPEG2000 still-image compression standard," in *Proceedings of IEEE International Conference on Image Processing*, Thessaloniki, Greece, Oct. 2001, course given at conference.

[2] A. N. Skodras, C. A. Christopoulos, and T. Ebrahimi, "JPEG2000: The upcoming still image compression standard," *Pattern Recognition Letters*, vol. 22, pp. 1337–1345, 2001.

[3] M. W. Marcellin, M. J. Gormish, A. Bilgin, and M. P. Boliek, "An overview of JPEG-2000," in *Proc. IEEE Data Compression Conference (DCC'00)*, Snowbird, UT, Mar. 2000, pp. 523–541.

[4] D. Taubman, E. Ordentlich, M. Weinberg, and G. Seroussi, "Embedded block coding in JPEG 2000," *Signal Processing: Image Communication*, vol. 17, pp. 49–72, 2002.

[5] M. D. Adams, "The JPEG2000 still image compression standard," ISO/IEC JTC 1/SC 29/WG 1 N 2412, Tech. Rep., Dec. 2002.

[6] M. Charrier, D. Santa Cruz, and M. Larsson, "JPEG 2000, the next millenium compression standard for still images," in *Proc. of the IEEE International Conference on Multimedia Computing and Systems (ICMCS'99)*, Florence, Italy, June 1999, pp. 131–132.

[7] ISO/IEC, *15444-1:2000 Information technology - JPEG2000 - image coding system - Part 1: Core coding system*, International Standard ISO/IEC 15444-1 Std., 2000.

[8] D. S. Taubman and M. W. Marcellin, *JPEG2000: Image compression fundamentals, standards, and practice*. Kluwer International Series In Engineering and Computer Science, 2002.

[9] N. Ahmed, T. Natarajan, and K. R. Rao, "Discrete cosine transform," *IEEE Trans. Computers*, vol. C-23, pp. 90–93, Jan. 1974.

[10] G. K. Wallace, "The JPEG still picture compression standard," *Communications of the ACM*, vol. 34, no. 4, pp. 30–44, 1991.

[11] ISO/ITU, "ISO 10918 information technology – digital compression and coding of continuous-tone still images: Requirements and guidelines," 1994.

[12] J. M. Shapiro, "Embedded image coding using zerotree of wavelet coefficients," *IEEE Trans. Signal Processing*, vol. 41, pp. 3445–3462, Dec. 1993.

[13] A. Said and W. A. Pearlman, "A new fast and efficient image codec based on set partitioning in hierarchical trees," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 6, pp. 243–250, June 1996.

[14] E. Ardizzone, M. La Cascia, and F. Testa, "A new algorithm for bit rate allocation in JPEG 2000 tile encoding," in *Proc. IEEE International Conference on Image Analysis and Processing (ICIAP'03)*, Mantua, Italy, Sept. 2003, pp. 529–534.

[15] T.-H. Chang, L.-L. Chen, C.-J. Lian, H.-H. Chen, and L.-G. Chen, "Computation reduction technique for lossy JPEG 2000 encoding through ebcot tier-2 feedback processing," in *Proc. IEEE International Conference on Image Processing (ICIP'02)*, Rochester, NY, June 2002, pp. 85–88.

[16] S. Battiato, A. Buemi, G. Impoco, and M. Mancuso, "Content - dependent optimization of JPEG 2000 compressed images," in *Proc. IEEE International Conference on Consumer Electronics (ICCE'02)*, Los Angeles, CA, June 2002, pp. 46–47.

[17] K. C. B. Tan and T. Arslan, "An embedded extension algorithm for the lifting based discrete wavelet transform in JPEG 2000," in *Proc. IEEE Internation Conference on Acoustic, Speech, and Signal Processing (ICASSP'02)*, Orlando, FL, May 2002, pp. 3513–3516.

[18] M. Kurosaki, K. Munadi, and H. Kiya, "Error concealment using layer structure for JPEG 2000 images," in *Proc. IEEE Asia-Pacific Conference on Circuit and Systems (APCCS'02)*, Oct. 2002, pp. 529–534.

[19] L. Aztori, A. Corona, and D. Giusto, "Error recovery in JPEG 2000 image transmission," in *Proc. IEEE International Conference on Acoustic, Speech, and Signal Processing (ICASSP'01)*, Salt Lake City, UT, May 2001, pp. 364–367.

[20] Y. M. Yeung, O. C. Au, and A. Chang, "Successive bit - plane allocation technique for JPEG2000 image coding," in *Proc. IEEE International Conference on Acoustic Speech*, Hong Kong, China, Apr. 2003, pp. 261–264.

[21] G. Pastuszak, "A novel architecture of arithmetic coder in JPEG2000 based on parallel symbol encoding," in *Proceedings of the IEEE International Conference on Parallel Computing in Electrical Engineering*, Sept. 2004, pp. 303–308.

[22] T. Tillo and G. Olmo, "A novel multiple description coding scheme compatible with the JPEG2000 decoder," *IEEE Signal Processing Letters*, vol. 11, no. 11, pp. 908–911, Nov. 2004.

[23] W. Du, J. Sun, and Q. Ni, "Fast and efficient rate control approach for JPEG2000," *IEEE Transactions on Consumer Electronics*, vol. 50, no. 4, pp. 1218 – 1221, Nov. 2004.

[24] K. Varma and A. Bell, "Improving JPEG2000's perceptual performance with weights based on both contrast sensitivity and standard deviation," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 3, May 2004, pp. 17–21.

[25] T. Kim, H. M. Kim, P.-S. Tsai, and T. Acharya, "Memory efficient progressive rate-distortion algorithm for JPEG 2000," *IEEE Transaction on Circuits and Systems for Video Technology*, vol. 15, no. 1, pp. 181–187, Jan. 2005.

[26] K. Vikram, V. Vasudevan, and S. Srinivasan, "Rate-distortion estimation for fast JPEG2000 compression at low bit-rates," *IEE Electronics Letters*, vol. 41, no. 1, pp. 16–18, Jan. 2005.

[27] M. Aguzzi, "Working with JPEG 2000: two proposals for sparse histogram images," in *Proc. IEEE International Conference on Image Analysis and Processing (ICIAP'03)*, Mantua, Italy, Sept. 2003, pp. 408–411.

[28] P. J. Ferreira and A. J. Pinho, "Why does histogram packing improve lossless compression rates?" *IEEE Signal Processing Letters*, vol. 9, no. 8, pp. 259–261, Aug. 2002.

[29] A. J. Pinho, "An online preprocessing technique for improving the lossless compression of images with sparse histograms," *IEEE Signal Processing Letters*, vol. 9, no. 1, pp. 5–7, Jan. 2002.

[30] ——, "On the impact of histogram sparseness on some lossless image compression techniques," in *Proceedings of the IEEE International Conference on Image Processing*, 2001, pp. 442–445.

[31] ——, "A comparison of methods for improving the lossless compression of images with sparse histogram," in *Proceedings of IEEE International Conference on Image Processing*, 2002, pp. 673–676.

[32] ——, "An online preprocessing technique for improving the lossless compression of images with sparse histograms," *IEEE Signal Processing Letters*, vol. 9, pp. 5–7, 2002.

[33] C. K. Chui, A. K. Chan, and C. S. Liu, *An Introduction To Wavelets*. San Diego, CA: Academic Press, 1991.

[34] M. Vetterli and J. Kovačević, *Wavelets And Subband Coding*. Prentice Hall, 1995.

[35] G. Strang and T. Nguyen, *Wavelets and filters banks*. Wellsley, MA: Wellsley-Cambridge Press, 1997.

[36] I. Daubechies, *Ten lectures on wavelets*, ser. CBMS Lecture. SIAM, 1992, no. 61.

[37] P. N. Topiwala, *Wavelet Image and Video Compression*. Norwell, MA: Kluwer Academic Publisher, 1998.

[38] M. Unser and T. Blu, "Mathematical properties of the JPEG2000 wavelet filters," *IEEE Trans. Image Processing*, vol. 12, pp. 1080–1090, Sept. 2003.

[39] S. G. Mallat, "A theory for multiresolution signal decomposition: the wavelet representation," *IEEE Transaction on Pattern Analysis and Machine Intelligence*, vol. 11, no. 7, pp. 674–693, July 1989.

[40] K. Andra, C. Chakrabarti, and T. Acharya, "Efficient implementation of a set of lifting based wavelet filters," in *Proc. IEEE International Conference on Acoustic, Speech, and Signal Processing (ICASSP'01)*, Salt Lake City, UT, May 2001, pp. 1101–1104.

[41] D. Taubman, "High performance scalable image compression with EBCOT," *IEEE Trans. Image Processing*, vol. 9, pp. 1158–1170, July 2000.

[42] I. H. Witten, R. M. Neal, and J. G. Cleary, "Arithmetic coding for data compression," *Communication of the ACM*, vol. 30, no. 6, pp. 520–540, June 1987.

[43] M. J. Slattery and J. L. Mitchell, "The qx-coder," *IBM Journal of Research and Development, Data compression technology in* ASIC *cores*, vol. 42, no. 6, 1998.

[44] M. D. Adams, *Jasper software reference manual*, Oct. 2002. [Online]. Available: http://www.ece.uvic.ca/ mdadams/jasper/jasper.pdf

[45] D. Santa-Cruz and T. Ebrahimi, "An analitical study of JPEG2000 functionalities," in *Proceedings of the IEEE International Conference on Image Processing*, vol. 2, Sept. 2000, pp. 49–52.

[46] M. Weinberger, G. Seroussi, and G. Sapiro, "The loco-i lossless image compression algorithm: Principles and standardization into jpeg-ls," HP Labs, Tech. Rep., 2000.

[47] ISO/IEC, "15948:2004 information technology - computer graphics and image processing - portable network graphics (png): Functional specification," ISO/IEC, Tech. Rep., 2004.

[48] S. Haykin, *An Introduction to Analog and Digital Communications*. John Wiley & Sons, 1991.

[49] B. W. Kerninghan and D. M. Ritchie, *The* C *Programming Language (*ANSI C*)*. Prentice Hall, 1988.

[50] The Mathworks Staff, *Using Matlab*, The Mathworks, Ed. Natick, MA: The Mathworks, Inc., 2000.

[51] Z. Liu and L. Karam, "Mutual information-based analysis of JPEG2000 contexts," *IEEE Transaction on Image Processing*, vol. 14, no. 4, pp. 411–422, Apr. 2005.