Chapter 10

How does digital cinema compress images?

In spite of the strong temporal redundancy of video, in the Digital Cinema industry each image from a movie is compressed separately[1]

A. Descampe(•), C. De Vleeschouwer(°), L. Jacques(°+) and F. Marqués(*)

(•) intoPIX S.A., Belgium
(°) Université catholique de Louvain, Belgium
(+) Ecole Polytechnique Fédérale de Lausanne, Switzerland.
(*) Universitat Politècnica de Catalunya, Spain.

The development of digital technologies has drastically modified the requirements and constraints that a good image representation format should meet. Originally, the requirements were to achieve good compression efficiency while keeping the computational complexity low. This has lead in 1992 to the standardization of the JPEG format, which is still widely used today (see Chapter 8). Over the years however, many things have evolved: more computing power is available and the development of Internet has required image representation formats to be more flexible and network-oriented, to enable efficient access to images through heterogeneous devices.

[1] This work has been partially supported by was supported by the Belgian National Funds for Scientific Research (FRS-FNRS)

In this context, the JPEG committee worked on an advanced and versatil image compression algorithm, called *JPEG2000* (Rabbani and Joshi 2002, Skodras et al 2001, Taubman and Marcellin 2002). It became an International Standard from the ISO[2] in 2001. Since then, it has been adopted on a growing number of professional markets that require both high quality images and intrinsic *scalability*, i.e. intrinsic ability to seamlessly adapt to the user needs and available resources[3]. Among those markets, we should cite Digital Cinema (DC)[4], which has adopted JPEG2000 as its official standard (DCI 2005), or the *contribution*[5] segment of broadcast solutions (Symes 2006). Other potential applications are medical imaging (Tzannes and Ebrahimi 2003, Foos et al. 2003), remote sensing (Zhang and Wang 2005), and audio-visual archives (Janosky and Witthus 2004).

Fig. 10.1 illustrates the high compression efficiency (Santa-Cruz et al. 2002) by comparing a JPEG and a JPEG2000 image at two different compression ratios. However, the feature that makes JPEG2000 really unique is its scalability. From a functional point of view, image scaling can be done at four different levels (see Fig. 10.2):

1 The *resolution*: the wavelet transform (further described below) reorganizes the information in so-called *resolution levels*, each of them incrementally refining the spatial resolution of the image. Starting from the original full resolution, each successive level transforms its input image into a four times smaller[6] image plus details coefficients. Independent encoding of low resolution image and details coefficients enables access at multiple resolutions.

2 The *bit-depth*: data is entropy-encoded on a "per bit-plane" basis. This means that most significant bits of all wavelet coefficients are encoded before less significant ones. By grouping encoded bits of equal significance, we obtain *quality layers*. The first quality layer gives a

[2] International Organization for Standardization

[3] In a more general sense, scalability is defined as "the ability of a computer application or product (hardware or software) to continue to function well when it (or its context) is changed in size or volume in order to meet a user need" (from http://www.whatis.com).

[4] It might sound surprising that a still image codec is used, instead of an MPEG video standard (see Chapter 9). However, extensive tests have revealed that the advantage of exploiting temporal redundancy between successive frames is significantly reduced for the high resolution and quality levels required by Digital Cinema (Smith and Villasenor 2004, Fossel et al. 2003, Marpe et al. 2003).

[5] The contribution, in broadcast, is the transfer of high quality versions of the distributed media between different broadcast providers.

[6] Each level divides the image width and height by 2.

coarse version of the image (only the most significant bits of each pixel are used), which is further refined by subsequent quality layers.

3 The *spatial location*: any spatial area inside an image can easily be extracted from a JPEG2000 codestream without having to process other parts of this image.

4 The *colour component*: when an image is made of several components (like in colour images or, more generally, multi-modal images), each component is coded independently and can therefore be extracted separately.

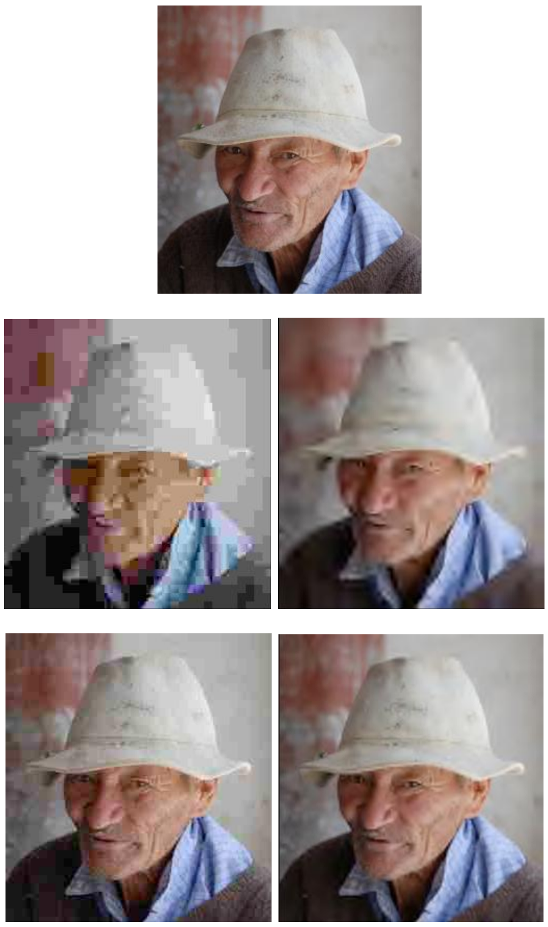


Fig. 10.1 Subjective comparison between JPEG (left) and JPEG2000 (right) compression efficiency. First row: original image. Second row: compression ratio of 170. Third row: compression ratio of 65.

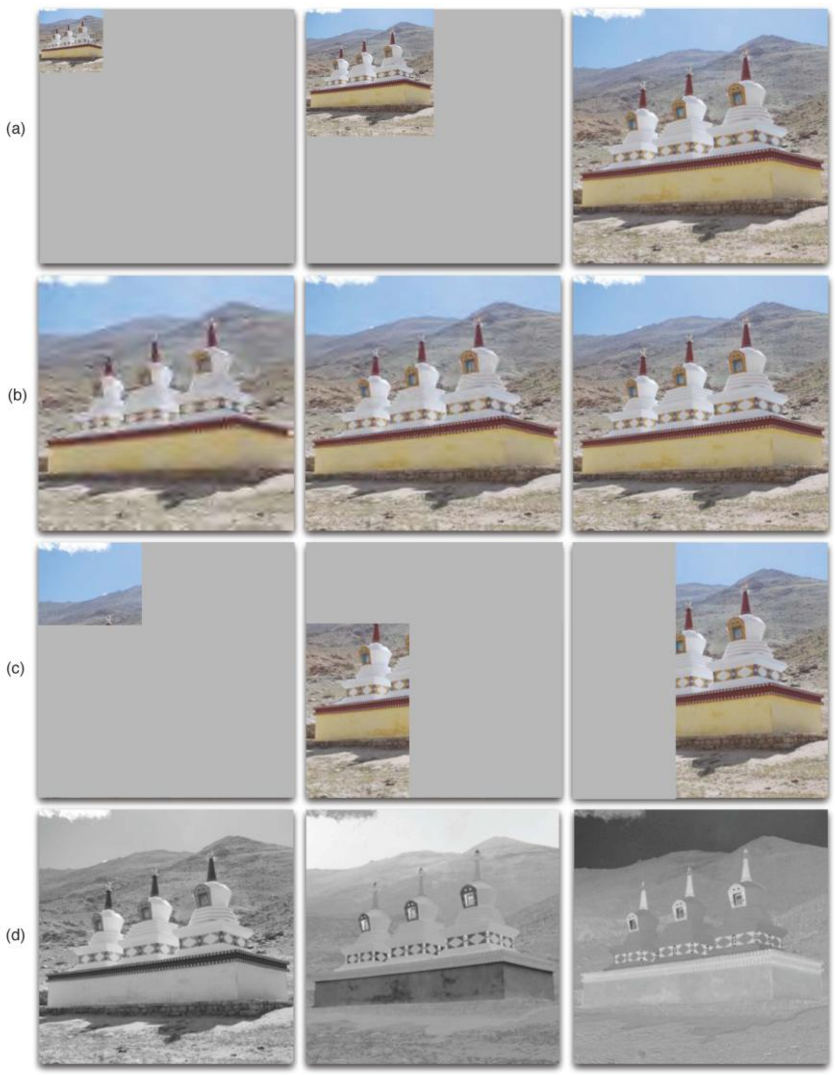


Fig. 10.2 Scalability in JPEG2000: Starting from a JPEG2000 codestream, image information can be extracted in several different ways: (a) by resolution, (b) by quality layer, (c) by spatial location, (d) by component (Y, Cb and Cr components, represented with a grayscale color map).

Based on the above image representation mechanisms, a JPEG2000 codestream organizes the image data in a hierarchical set of packets, each one containing the information related to a given quality layer from a given resolution, in a given spatial location of one of the image components. Thereby, one can easily extract the exact portion of information that corresponds to his/her needs (in terms of image area, resolution, etc) and available resources (bandwidth, display, etc).

The rest of the Chapter further explains and demonstrates the fundamental mechanisms that support such a versatile scalability. Specifically, the wavelet transform, the bit-plane encoder, and the grouping of bit-planes into quality layers are respectively considered in Section 10.1, Section 10.2 and Section 10.3. MATLAB proof-of-concept experiments are proposed in Section 0. For completeness, Section 10.5 concludes with a survey of the complete pipeline implemented to generate a JPEG2000 codestream, based on the concepts introduced along the Chapter.

10.1 Background – Introduction to wavelet and multiresolution transforms

This Section introduces some basic concepts about wavelet analysis of 1-D and 2-D signals. Essentially, wavelets aim at changing the representation of a signal (i.e. the association of a time or a position with a certain value) so as to reorganize the information contained in the signal and reveal some properties that did not appear clearly in the initial representation.

Before the birth of wavelet analysis, most of signal processing was performed using global tools such as global signal statistics, Fourier transform/series, global Discrete Cosine Transform (DCT), etc. These decompositions are global in the sense that they do not provide any information about the local signal structure. The DCT of a signal for instance points us "how many" frequencies are present inside a temporal sequence, but we do not know when each one was produced : *there is no way to produce a music partition with the Fourier reading of an orchestral symphony, you can just count the number of particular notes produced during the whole concert.*

To address this weakness, some attempts were early made to "artificially" localize these techniques by computing them within several limited time intervals or support areas. However, this solution has its drawbacks. The image block artefact illustrated in Fig. 10.1 for the JPEG compression is for instance due to an image representation that is split across a set of independent and non-overlapping block-based DCT.

In contrast, this windowing process is incorporated naturally within a time-frequency 1-D signal representation known as the *Wavelet Transform* (WT). As explained above, the WT gives birth to *multiresolution descriptions* of any signal and can easily be generalized to image representation.

10.1.1 Think globally, act locally

Let us start by dealing with 1-D continuous signal and see later how to practically manipulate signals digitally recorded as a discrete sequence of values. Our signal is a continuous function *s*(*t*) representing the recording of some physical process at every time $t \in \mathbb{R}$. For convenience, *s* is not displaying weird behaviours and has, for instance, a finite energy; that is:

$$s \in L^2(\mathbb{R}) = \left\{ u : \|u\|^2 \triangleq \int_{\mathbb{R}} |u(t)|^2 \, dt < \infty \right\} \tag{10.1}$$

In the purpose of analyzing the content of *s*, we may first compute its *approximation* $A_0 s$ in the set V_0 of signals that are constant on each interval $I_{0,n} = [n, n+1) \in \mathbb{R}$ for $n \in \mathbb{Z}$. To compute such an approximation, it is convenient to define

$$\varphi(t) = \begin{cases} 1 & \text{if } 0 \le t < 1 \\ 0 & \text{elsewhere} \end{cases} \tag{10.2}$$

The function φ, named the Haar *scaling function*, is fundamental for describing elements of V_0. The set $\Phi_0 = \left\{ \varphi_{0,n}(t) = \varphi(t-n) : n \in \mathbb{Z} \right\}$, made of translated copies of φ, is a basis for V_0, i.e. any function of V_0 can be described as a linear combination of the φ_0 elements. This basis is actually orthonormal according to the usual scalar product $\langle u, v \rangle = \int_{\mathbb{R}} u(t) v(t) dt$, i.e. $\langle \varphi_{0,n}, \varphi_{0,m} \rangle = \delta_{nm}$ with δ_{nm} the Kronecker's symbol, which is equal to 1 if $n = m$ and to 0 otherwise.

Thanks to φ, the average of the signal *s* in interval $I_{0,n} = [n, n+1)$ is simply computed by

$$a_0(n) \triangleq \langle \varphi_{0,n}, s \rangle = \int_n^{n+1} s(t) dt \tag{10.3}$$

Thus, $a_0(n)\varphi_{0,n}(t)$ is nothing but the approximation of *s* in $I_{0,n}$ by a constant function of height $a_0(n)$. For the whole time line, the approximation reads:

$$A_0 s \triangleq \sum_{n \in \mathbb{Z}} a_0(n) \varphi_{0,n}(t) \tag{10.4}$$

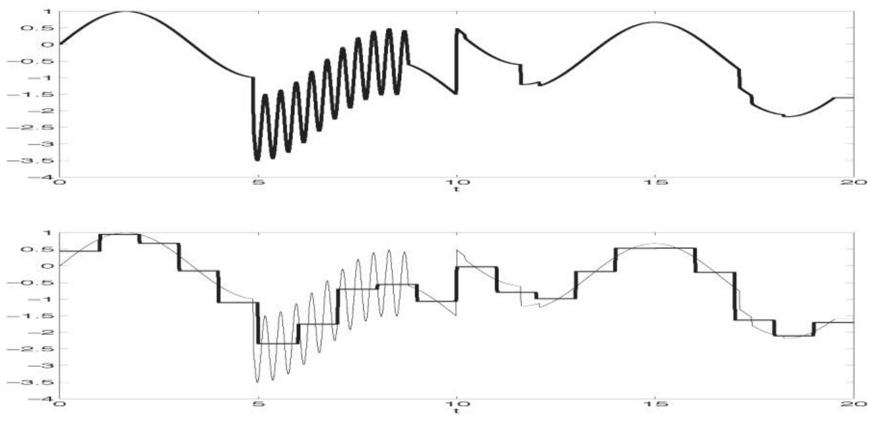


Fig. 10.3 1st row: Original signal. 2nd row: Approximation $A_0 s$ of this signal in V_0

A simple approximation illustration is drawn in Fig. 10.3 for a toy signal s made of smooth, transient and oscillatory parts on the time interval [0,20]. As expected, $A_0 s \in V_0$ approximates the initial s but we can notice that smoothed parts of s are better rendered than the transient ones. The very oscillating part between t = 5 and t = 9 is for instance completely smoothed while its sloppy trend is preserved.

10.1.2 Approximate... but details matter

The numerical example of the last Section suggests that we could tune the level of resolution of the piecewise constant function set on which s is projected to approximate with better success some parts of a given signal.

In other words, we want to work now with the general space V_j of functions constant on intervals $I_{j,n} = \left[2^{-j} n, 2^{-j}(n+1)\right)$. Indices j and n are named the *resolution* and the *position* parameters respectively. The higher j, the better the approximation of s in V_j.

These spaces V_j are hierarchically organized, i.e. each V_j is included into V_{j+1}. Moreover, $\Phi_j = \left\{\varphi_{j,n}(t) = 2^{j/2}\varphi(2^j t - n) : n \in \mathbb{Z}\right\}$ is the new orthonormal[7] basis for V_j. Approximating s by $A_j s \in V_j$ is straightforward with:

[7] The multiplicative constant $2^{j/2}$ in the definition of $\varphi_{j,n}$ guarantees the unit normalization at all resolutions, i.e. $\|\varphi_{j,n}\| = 1$

$$A_j s \triangleq \sum_{n\in\mathbb{Z}} a_j(n)\varphi_{j,n}(t), \qquad a_j(n) = \langle \varphi_{j,n}(t), s\rangle \tag{10.5}$$

Note that $a_j(n)\varphi_{j,n}(t)$ is a constant function over interval $I_{j,n}$ of height equal to the average value of $s(t)$ over $I_{j,n}$. Therefore, for infinite resolution j, i.e. when $j \to \infty$, V_j tends to the space $L^2(\mathbb{R})$ itself since these averages tend to the actual values of $s(t)$.

We may notice that the density of basis function $\varphi_{j,n}$ per unit of time changes with the resolution. Indeed, at a given resolution j, $\varphi_{j,n}$ and $\varphi_{j,n+1}$ are separated by a distance 2^{-j}. We will see later that this observation leads to the *downsampling*[8] (or *upsampling*[9]) operations considered in Section 10.1.3.

In this general framework, asking now if the resolution j is sufficient to represent with high quality a local signal behaviour is related to determining the information that is lost when switching from one resolution $j + 1$ to the coarser one j.

Therefore, we would like to compute the *details* that have to be added to an approximation $A_j s$ to get the approximation at level $(j + 1)$. Trivially, we have

$$A_{j+1}s = A_j s + \left(A_{j+1}s - A_j s\right) = A_j s + D_j s \tag{10.6}$$

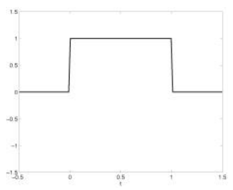
where $D_j s$ are those details. We may remark that $D_j s \in V_{j+1}$ but the V_{j+1} space is actually too big to describe well any possible $D_j s$. In fact $D_j s$ belongs to a new space of functions W_j which is the *detail space* of resolution j. It is also referred as *subband* of resolution j in the literature, and can be generated by the orthogonal basis $\Psi_j = \left\{\psi_{j,n}(t) = 2^{j/2}\psi(2^j t - n) : n \in \mathbb{Z}\right\}$, based on the Haar *wavelet*:

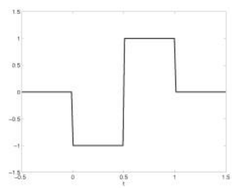
$$\psi(t) = \begin{cases} -1 & \text{if } 0 \le t < 1/2 \\ 1 & \text{if } 1/2 \le t < 1 \\ 0 & \text{elsewhere} \end{cases} \tag{10.7}$$

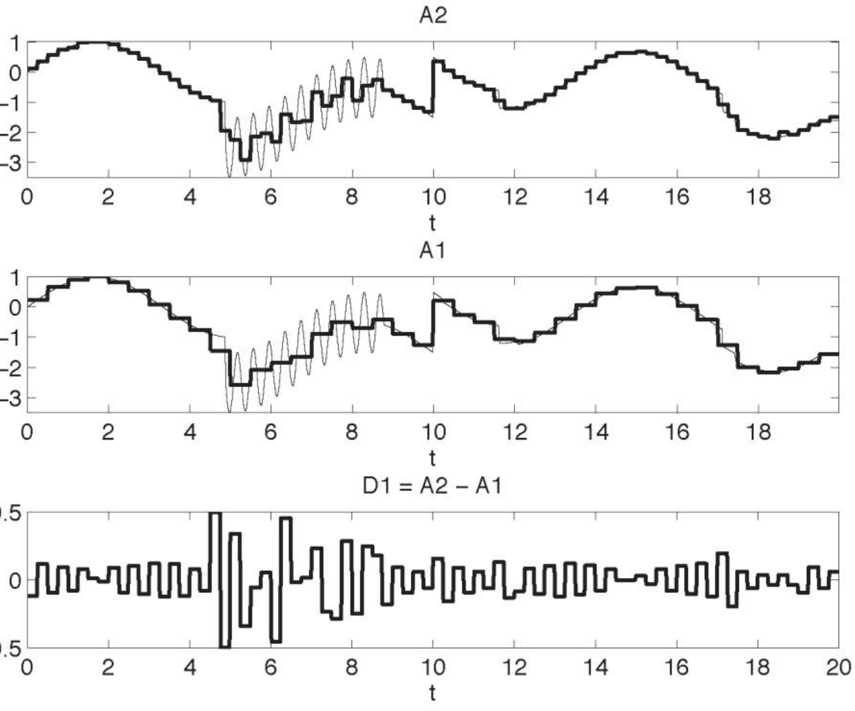
[8] That is deleting every other position n

[9] That is inserting zeros between every position n

The wavelet function ψ has a vanishing average, i.e. $\int_{\mathbb{R}} \psi(t)dt = \langle \psi, 1 \rangle = 0$, and is also orthogonal to φ, i.e. $\langle \psi, \varphi \rangle = 0$. Both functions φ and ψ are represented in Fig. 10.4.a and Fig. 10.4.b respectively.


(a)


(b)



(c)

Fig. 10.4 (a) Haar scaling function. (b) Haar wavelet. (c) Example of detail coefficients from two resolution levels. The signal analyzed is the same as in Fig. 10.3, which is drawn in gray on the two top figures

The piecewise constant shape of ψ is implicitly determined by Eq. (10.7) and by the definition of φ. In short, for $t \in I_{j,n}$, we know that

$A_j s(t) = a_j(n)\varphi_{j,n}(t)$. The height of this constant function, i.e. $2^{j/2}a_j(n)$, is also the average of $s(t)$ over interval $I_{j,n}$. By construction, this interval merges two adjacent intervals from the higher level of approximation, i.e. $I_{j,n} = I_{j+1,2n} \cup I_{j+1,2n+1}$. As a consequence, an approximation at level j is exactly the mean value of the two corresponding approximations computed at level (j+1), and the plateau value of $A_j\, s$ on $I_{j,n}$ is exactly at mid-height of the two constant pieces of:

$$A_{j+1}s(t) = a_{j+1}(2n)\varphi_{j+1,2n}(t) + a_{j+1}(2n+1)\varphi_{j+1,2n+1}(t) \qquad (10.8)$$

Therefore, the detail $D_js = A_{j+1}s - A_js$ has thus the shape of ψ over $I_{j,n}$.

In Fig. 10.4.c, we give a plot of D_1s on the previous numerical signal (Fig. 10.3). The two plots on top represent A_2s (A2) and A_1s (A1). The last on the bottom is $D_1s = A_2s - A_1s$ (D2). In the D_1s plot, we may observe that detail coefficients with high amplitude, i.e. absolute value, are not very numerous. They are mainly concentrated around the transient parts of the original signal (mainly in interval [5, 10]). This localization effect is a manifestation of the zero average of ψ. Whenever $s(t)$ is almost constant on a certain interval $U \in \mathbb{R}$, if $\psi_{j,n}$ is well concentrated on U, the corresponding wavelet coefficient vanishes.

Without entering into too many details, let us mention that the Haar basis is just one example, actually the simplest, of orthonormal basis leading to a Wavelet Transform definition. All the previous definitions, i.e. scaling and wavelet functions, approximation and detail spaces V_j and W_j, can be extended in a general formalism named *Multiresolution Analysis* (Mallat 1999, Daubechies 1992). In particular, the wavelet and the scaling functions can be designed to reach certain regularity properties (vanishing moments, compact support, etc), to address the specific requirements of the underlying signal processing or analysis application.

10.1.3 Wavelet Transform: Definition and Computation

Now that we know how to switch from one resolution to another, we can iterate the decomposition of Eq. (10.6) from resolution level J as follows[10]

[10] As it will be shown in Section 10.1.4, typically the highest resolution level for discrete signals is the original signal itself.

$$\begin{aligned} s(t) &= \lim_{j\to\infty} A_j s(t) = A_J s(t) + \sum_{j=J}^{\infty} D_j s(t) \\ &= \sum_n a_J(n)\varphi_{J,n}(t) + \sum_{j=J}^{\infty} d_j(n)\psi_{j,n}(t) \end{aligned} \tag{10.9}$$

The (*Haar*) *wavelet transform* of a signal *s* is the collection of all the detail (or wavelet) coefficients $d_j(n) = \langle \psi_{j,n}, s \rangle$ plus the approximation coefficients $a_J(n)$. Eq. (10.9) is referred to the inverse WT, taking coefficients and rebuilding the original signal $s(t)$.

The WT implies a new representation of the signal: a temporal or spatial description of *s* has been replaced by a 2-D resolution/position description $\{d_j(n), a_J(n)\}$. Moreover, the locality of the functions φ and ψ involves that these coefficients are actually a local measurement of *s*. As a mathematical microscope, coefficients $d_j(n)$ are thus probing locally the content of the signal *s* with a lens ψ of magnification 2^j, i.e. with a size 2^{-j}, and at position $2^{-j}n$.

About the practical computation of the WT, up to now it was suggested that scalar products, i.e. complete integrations on $\mathbb{R}$ (or Riemann sums for numeric simulations), between *s* and basis elements $\psi_{j,n}$ or $\varphi_{J,n}$ must be computed. There is however a recursive technique to follow based on the idea that φ and ψ respect each a *scaling equation*, i.e. there exists filters *h* and *g* such that

$$\varphi_{j-1,0}(t) = \sum\nolimits_n h(n)\varphi_{j,n}(t) \tag{10.10}$$

$$\psi_{j-1,0}(t) = \sum\nolimits_n g(n)\psi_{j,n}(t) \tag{10.11}$$

The sequence $h(n) = \langle \varphi_{j,n}, \varphi_{j-1,0} \rangle$ and $g(n) = \langle \psi_{j,n}, \psi_{j-1,0} \rangle$ are named the *conjugate mirror filters* (or CMF) of φ and ψ. Interestingly, it may be noticed that the values of *h* and *g* are actually independent of the resolution *j*. In other words, the link existing between the scaling functions and the wavelets between two consecutive resolutions is always the same. It can be shown also that *h* and *g* stay the same if we translate in time the functions on the left of Eq. (10.10) and Eq. (10.11), i.e. if we develop in the same way $\varphi_{j-1,m}$ and $\psi_{j-1,m}$ for a given integer *m*. In consequence, as shown below, the knowledge of the sequences *h* and *g* dramatically simplifies the computation of the WT.

For the Haar basis, they are quite simple to calculate; the only non-zero elements are $h(0) = h(1) = 1/\sqrt{2}$ and $g(0) = -g(1) = 1/\sqrt{2}$. Note that for general (φ, ψ) i.e. outside of the Haar system, we have always $g(n) = (-1)^{1-n} h(1-n)$ (Daubechies 1992).

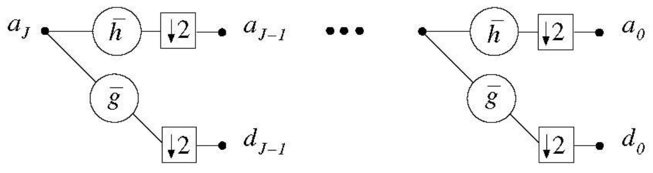
Thanks to these scaling relations, it is easy to prove (Mallat 1999) that

$$a_{j-1}(p) = \sum_n h(n-2p) a_j(n) = \left(a_j * \bar{h}\right)(2p) \tag{10.12}$$

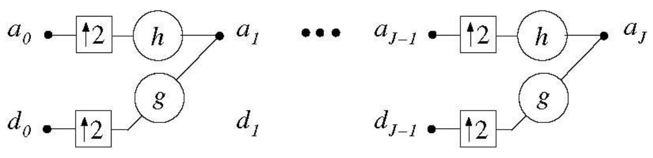
$$d_{j-1}(p) = \sum_n g(n-2p) a_j(n) = \left(a_j * \bar{g}\right)(2p) \tag{10.13}$$

with $\bar{u}(n) = u(-n)$ and * stands for the discrete *convolution* between two sequences. Notice the presence of a factor 2 in the argument of the convolution. As formally described in Section 10.1.2, this represents actually a *downsampling* by a factor 2 of the density of positions when passing from resolution *j* to resolution *j-1*.

Therefore, from an "engineer" point of view, a wavelet transform can be seen as the recursive application of a filter bank, i.e. the combination of the discrete filters *h* and *g*, thereby resulting in a low resolution version of the signal plus a set of detail subbands.



(a) Decomposition



(b) Reconstruction

Fig. 10.5 (a) Decomposition and (b) Reconstruction scheme for approximation and detail coefficients computations. Symbols $\uparrow 2$ and $\downarrow 2$ represent up-sampling, and down-sampling operations respectively. Circles mean convolution with filter name inside.

In the converse sense, approximation coefficients can be rebuilt from approximation and detail coefficients at a coarser level with

$$a_{j+1}(p) = \left(\breve{a}_j * h\right)(p) + \left(\breve{d}_j * g\right)(p) \tag{10.14}$$

where, for any sequence u, $\breve{u}(2n) = u(n)$ and $\breve{u}(2n+1) = 0$, i.e. the operator inserts zero between sequence elements (*upsampling* operator).

In short, a_j and d_j are computed from a_{j+1}, and a_{j+1} can be rebuilt from a_j and d_j, in each case knowing the CMF filters h and g. This hierarchic computing (summarized schematically in Fig. 10.5) drastically simplifies the computation of the WT compared to an approach in which each coefficient should be computed based on the convolution of the original signal with rescaled filters.

Biorthogonality

Let us include a remark on the concept of *biorthogonality*. The mentioned orthogonality between φ and ψ may be lifted leading for instance to the definition of *biorthogonal* systems. This requires the addition of two *dual* functions $\tilde{\varphi}$ and $\tilde{\psi}$. The set $\{\varphi, \psi, \tilde{\varphi}, \tilde{\psi}\}$ is more flexible and easier to design than an orthogonal system $\{\varphi, \psi\}$.

In the induced biorthogonal WT, the direct functions φ and ψ are used to compute the approximation and the detail coefficients a_j and d_j, while the dual functions are related to the reconstruction process.

Table 10.1 Examples of CMF filters. Filters g and $\tilde{g}$ are not represented for Daubechies filters since $g(n) = (-1)^{1-n}\tilde{h}(1-n)$ and $\tilde{g}(n) = (-1)^{1-n}h(1-n)$.

n	Haar $\tilde{h}[n]$	Haar $h[n]$
0	$2^{-1/2}$	$-2^{-1/2}$
1	$2^{-1/2}$	$2^{-1/2}$

n	Daubechies 7/9 $\tilde{h}[n]$	Daubechies 7/9 $h[n]$
0	0.78848561640637	0.85269867900889
±1	0.41809227322204	0.37740285561283
±2	-0.04068941760920	-0.11062440441844
±3	-0.06453888262876	-0.02384946501956
±4		-0.03782845554969

On the filter side, we work also with 4 filters $\{h, g, \tilde{h}, \tilde{g}\}$. The first two, the direct filters h and g, are used in the decomposition process, while the *dual filters* $\tilde{h}$ and $\tilde{g}$ are involved in the reconstruction. For instance, JPEG2000 uses either the Daubechies 9/7 filters, i.e. h and $\tilde{h}$ have 9 and 7 non-zero elements respectively (see Table 10.1), or the Daubechies 5/3 filters (named also LeGall 5/3).

10.1.4 WT and Discrete Signals: DWT

In the previous Sections, we have seen that, thanks to a very simple basis, a continuous signal can be decomposed in several resolutions of details. In practice, signals are provided as a discrete (and finite) sequence of numbers, corresponding to a *sampling* of the original signal s, i.e. with values $s_d(n)$.

To extend the above theory from continuous to discrete signals, we simply assume that s_d corresponds to the approximation coefficients[11] $a_J(n) = \langle \varphi_{J,n}, s \rangle$ of some hypothetical continuous s at resolution J. Thus, pyramidal rules of Section 10.1.3 can be directly applied onto $s_d(n) = a_J(n)$. This is what defines the discrete WT, or DWT.

Taking into account the downsampling arising in the computations of Eq. (10.12), the whole DWT of a discrete sequence s_d of $N = 2^J$ elements[12] at a resolution $0 \leq J_0 < J$, is thus composed of $2^{J-1} + 2^{J-2} + \ldots + 2^{J_0} = 2^J - 2^{J_0}$ wavelet coefficients d_j ($J_0 \leq j < J$), and 2^{J_0} approximation coefficients a_{J_0}. Therefore, the DWT provides exactly the same number of coefficients, i.e. $N = 2^J$, as the number of samples in s_d. No *redundancy* has been introduced in the transform.

For the inverse DWT (iDWT), the rule is thus simply to apply Eq. (10.13) to recover $a_J(n) = s_d(n)$ from the DWT coefficients. About the computation complexity of the DWT and the iDWT, thanks both to the downsampling and to the pyramidal computations, all can be performed in not more than $O(N)$ operations, i.e. the total number of multiplications-additions involved increases linearly with the number N of signal samples.

[11] See (Mallat 1999) for further details on that subject.

[12] To be exact, assuming s_d limited to N values induces particular boundary treatments in the DWT like making the sequence periodic or using different wavelets to compute WT coefficients involving boundaries. The interested reader may refer to (Daubechies 1992) or (Mallat 1999).

10.1.5 WT and DWT for Images: 1+1 = 2

In previous Sections, we were concerned only about wavelet transforms of 1-D signal. How can we extend it to the manipulations of images, i.e. 2-D functions? The recipe that we are going to describe below is quite simple: 2-D DWT can be obtained by applying 1-D DWT successively to each dimension of the image.

As previously commented, a wavelet transform can be seen as the recursive application of a filter bank, resulting in a low resolution version of the signal plus a set of detail subbands. When dealing with images, the computation of the wavelet transform can be performed by applying the filter bank successively in the vertical and horizontal directions, resulting in the computation process shown in Fig. 10.6.

Here, we follow the JPEG2000 terminology, and refer to *L* to represent a low-pass convolution by *h* (computing a signal approximation), and to *H* to denote a high-pass convolution by *g* (computing signal details). Step by step, in the notations of Section 10.1.3, the initial image *I* = `LL0` is first decomposed into two vertically down-sampled convolutions

$$\mathtt{L1}(\mathbf{n}) = \left(\mathtt{LL0} *_y \bar{h}\right)(n_x, 2n_y) \quad \text{and} \quad \mathtt{H1}(\mathbf{n}) = \left(\mathtt{LL0} *_y \bar{g}\right)(n_x, 2n_y) \qquad (10.15)$$

where $\mathbf{n} = (n_x, n_y)$ is the 2-D index, $*_u$ stands for the 1-D convolution (as in Eq. (10.12) performed in direction $u \in \{\text{“x”}, \text{“y”}\}$, i.e. horizontal or vertical.

By downsampling effect, `L1` and `H1` are thus two rectangular images as depicted in the second stage in Fig. 10.6. Then, the first level of a WT decomposition is eventually reached by applying horizontal convolutions to the two previous outputs so that, for instance, $\mathtt{LL1}(\mathbf{n}) = \left(\mathtt{L1} *_x \bar{h}\right)(2n_x, n_y)$ (third stage in Fig. 10.6). Finally, any other level of decomposition *n* + 1 is obtained iteratively by working on the last approximation `LLn` (fourth stage in Fig. 10.6). Notice that when *n* increases, resolution *j* decreases.

As subsampling operations arise in both horizontal and vertical directions, each subband *n* contains four times fewer coefficients than at resolution *n* + 1, and the *N*x*N* pixel values are transformed in *N*x*N* coefficients.

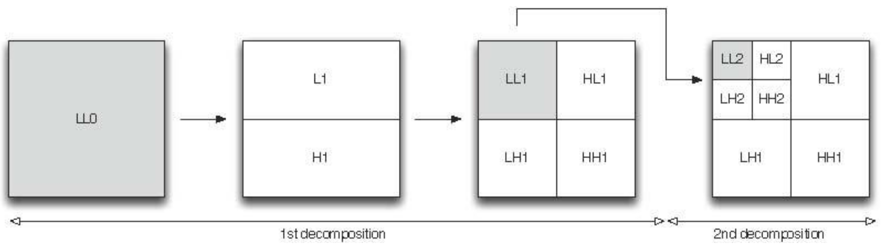


Fig. 10.6 2-levels Discrete Wavelet Transform: Continuing with the JPEG2000 terminology, note that the index of the different bands in the JPEG2000 context is increasing when the resolution decreases. This is highlighted here since JPEG2000 indexing is inverting the notation used at the previous Sections.

It is important to understand here that no compression has been done up to this point. The image information has just been reorganized and decorrelated (see Chapter 8) so as to concentrate the image energy in the upper left corner. By doing so, the image has been "prepared" for compression: most high-frequency subbands coefficients are indeed close to zero (see grey points in Fig. 10.7) and can be therefore efficiently compressed.

10.2 Background – Context-based modeling of wavelet coefficients bit-planes

This Section defines how the wavelet coefficients corresponding to the multiresolution representation of an image, as presented in Section 10.1, can be entropy coded, both efficiently and in a way that supports random spatial access and progressive decoding capabilities.

10.2.1 Spatial and bit-depth scalability

First, let us see how spatial and quality scalability is obtained. In short, each subband is split into several rectangular entities, named *codeblocks*, which are compressed independently to preserve random spatial access. To offer the capability to decode the image at multiple quality levels, the coefficients in a codeblock are bit-plane encoded, which means that a coefficient is primarily defined by its most significant bit, and progressively refined by adding bits in decreasing order of significance. The decomposition of the image into planes of bits is illustrated in Fig. 10.7.

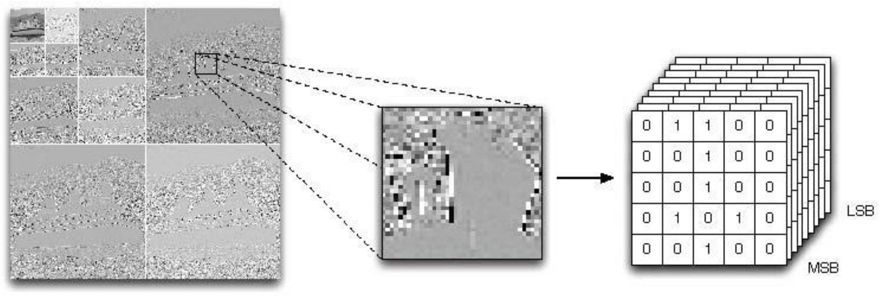


Fig. 10.7 Encoding of a codeblock on a per bit-plane basis. Codeblocks are usually 32x32 or 64x64 blocks of wavelet coefficients (here, only 25 coefficients have been represented for visual convenience).

10.2.2 Efficient entropy coding

Let us now describe how coefficient bit-planes can be efficiently entropy coded. Entropy coding has been introduced in Chapter 8, and consists in compacting the sequence of messages generated by a random source, based on the knowledge of the statistics of the source. When dealing with bit-planes, efficient compression thus relies on accurate estimation of the probability distribution associated to the sequence of binary symbols encountered while scanning the bit-planes, typically in a most-to-least significant and raster scan order (from left to right and from top to bottom). For improved estimation, a *context-based image modeller* has been introduced by JPEG2000 designers. Context-based means that the probability of a binary symbol is estimated based on its neighbourhood, also named its *context*.

In a sense, the context-based modeller partitions the aggregated image source into a set of sources characterized by distinct contexts. As we know that the aggregation of two sources (with known probability distributions) into a single source (modelled by an aggregated distribution) leads to an entropy increase, we conclude that the context-based approach increases coding efficiency, as long as the statistics associated to each context can be accurately estimated. The benefit is only significant when the statistics of the distinct sources are sufficiently different.

Formally, let $b_{i,j}$ denote the binary random variable associated to the j^{th} bit of the i^{th} bit-plane, and define $C_{i,j}$ to be the random state variable associated to the context of $b_{i,j}$, with possible realizations $\mathbf{c}$ of $C_{i,j}$ belonging to

Γ. Using $P(.)$ to denote the probability distribution of a random variable, the entropy of $b_{i,j}$ is

$$H\left(b_{i,j}\right) = -\sum_{k=0}^{1} P\left(b_{i,j} = k\right) \log_2\left(P\left(b_{i,j} = k\right)\right) \tag{10.16}$$

and we define the context-based entropy of $b_{i,j}$ as follows:

$$H_C\left(b_{i,j}\right) = -\sum_{\mathbf{c}=\Gamma} P\left(C_{i,j} = \mathbf{c}\right) \sum_{k=0}^{1} P\left(b_{i,j} = k \mid C_{i,j} = \mathbf{c}\right) \log_2\left(P\left(b_{i,j} = k \mid C_{i,j} = \mathbf{c}\right)\right) \tag{10.17}$$

In practice, the probability distributions $P(b_{i,j})$ and $P(b_{i,j} \mid C_{i,j} = \mathbf{c})$ are estimated based on histogram computations, i.e. probabilities are approximated by frequencies of occurrence. Those estimated values can be computed either based on the signal to encode or based on a predefined and representative set of images. In the first case, the frequencies of occurrence have to be transmitted as side information, while in the latter case, they are a priori known by both the coder and decoder.

In JPEG2000, the context is computed based on state variables related to surrounding coefficients, and to the processed coefficient itself. The most important state variable is the *significance status* of a coefficient. Initially, all coefficients are labelled as insignificant and bit-planes are processed from the most to the least significant one. A coefficient is said to switch from insignificant to significant at the most significant bit-plane for which a bit equal to ‘1’ is found for this coefficient. Once significant, the coefficient keeps this status for all the remaining less significant bit-planes. Other variables affecting the context are the kind of subband (LL, HL, LH or HH), the sign of the coefficient, and its *first refinement status*[13].

Intuitively, it is easy to understand that such a context improves the predictability of encoded binary values. Indeed, in a given bit-plane, if a non-significant coefficient is surrounded with significant ones, it is more likely to become significant (i.e. get a '1' bit) than if it is surrounded with non-significant coefficients. We will therefore use a higher probability of getting a '1' when encoding bits that belong to coefficients that are in this situation.

Based on the above arguments, we understand that context-based modelling is likely to significantly decrease the entropy of the binary source

[13] This variable is always equal to '0', except at the bit-plane immediately following the bit-plane where the coefficient became significant, where it is set to '1'

associated to bit-plane scanning. To turn such entropy gain into actual bit budget reduction, it is important to implement an efficient entropy coder. In JPEG2000, this is done by a *MQ-coder*, which is a derivative of the arithmetic Q-coder (Mitchell and Pennebaker 1988). According to the provided context, the coder chooses a probability for the bit to encode, among predetermined probability values supplied by the JPEG2000 Standard and stored in a look-up table[14]. Using this probability, it encodes the bit and progressively generates code-words.

10.3 Background - Rate-distortion optimal bit allocation across wavelet codeblocks

This Section describes how the single and complete codestream generated by the entropy coder can be adapted to meet a given bit-budget constraint, while preserving image quality. Such adaptation is typically required when storage or transmission resources get scarce.

In Section 10.2, we have explained that the coefficients of an image codeblock are encoded on a bit-plane by bit-plane basis. Hence, bit-budget reduction can simply be obtained by dropping the bitstream segments associated to the least significant codeblock bit-planes. Conversely to entropy coding, which does not cause any loss of information, such dropping mechanisms obviously affects image quality. Hence, a fundamental problem consists in deciding for each codeblock about the number of bit-planes to drop, so as to minimize the distortion on the reconstructed image while meeting the given bit-budget (storage or rate) constraint.

The problem of *rate-distortion* (RD) *optimal allocation* of a bit budget across a set of image blocks characterized by a discrete set of RD trade-offs[15] has been extensively studied in the literature (Shoham and Gersho 1988, Ortega et al. 1994, Ortega 1996). Under strict bit-budget constraints, the problem is hard, and its resolution relies on heuristic methods or dynamic programming approaches (Ortega et al. 1994). In contrast, when some relaxation of the rate constraint is allowed, Lagrangian optimization and convex-hull approximation can be considered to split the global optimization problem in a set of simple block-based local decision problems (Shoham and Gersho 1988, Ortega et al. 1994, Ortega 1996). This ap-

[14] For improved coding efficiency, JPEG2000 dynamically updates the probability distribution associated to each context along the coding process. In this way, the context-based modeller adapts to the image content and to the evolution of the probability distribution across the bit-planes

[15] In the JPEG2000 context, for each codeblock.

proach is described in details in the rest of this Section. In short the convex-hull approximation consists in restricting the eligible transmission options for each block to the RD points sustaining the lower convex hull of the available RD trade-offs of the block[16]. Global optimization at the image level is then obtained by allocating the available bit-budget among the individual codeblock convex-hulls, in decreasing order of distortion reduction per unit of rate.

10.3.1 Problem definition

We assume that *N* known input codeblocks have to be encoded using a given set *Q* of *M* admissible quantizers, such that the choice of the quantizer *j* for a codeblock *i* induces a distortion d_{ij} for a cost in bits equal to b_{ij}.

The objective is then to find the allocation $\mathbf{x} \in Q^N$, which assigns a quantizer $x(i)$ to codeblock *i*, such that the total distortion is minimized for a given rate constraint.

In our case, the index *j* of the quantizer refers to the number of encoded bit-planes, $0 \leq j \leq M$. We also assume an additive distortion metric, for which the contribution provided by multiple codeblocks to the entire image distortion is equal to the sum of the distortion computed for each individual codeblock.

In practice, the distortion metrics are computed based on the square error (SE) of wavelet coefficients, so as to approximate the reconstructed image square error (Taubman 2000). Formally, let $c_b(n)$ and $\hat{c}_b(n)$ respectively denote the two-dimensional sequences of original and approximated subband samples in codeblock *b*. The distortion d_{ij} associated to the approximation of the i^{th} codeblock by its *j* first bit-planes is then defined by

$$\mathrm{d}_{ij} = \sum_{n \in i} w_{sb}^2 \left(\hat{c}_i^j(n) - c_i(n) \right)^2 \tag{10.18}$$

where $c_i(n)$ and $\hat{c}_i^j(n)$ respectively denote the original and the quantized n^{th} coefficient of codeblock *i*, while w_{sb} denotes the L2-norm of the wavelet basis functions for the subband *sb* to which codeblock *b* belongs (Taubman 2000).

Formally, the rate-distortion optimal bit allocation problem is then formulated as follows:

[16] The convex hull or convex envelope of a set of points X in a vector space is the (boundary of the) minimal convex set containing X

Optimal rate-constrained bit allocation: For a given target bit-budget B_T, find $\mathbf{x}^*$ such that:

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} \sum_{i=1}^{N} \mathrm{d}_{ix(i)} \tag{10.19}$$

subject to

$$\sum_{i=1}^{N} \mathrm{b}_{ix(i)} < B_T \tag{10.20}$$

10.3.2 Lagrangian formulation and approximated solution

Strictly speaking, the above formulation is known in the literature as a Knapsack problem, which can be solved at high computational cost using dynamic programming (Kellerer 2004, Wolsey 1998). Hopefully, in most communication applications, the bit-budget constraint is somewhat elastic. Buffers absorb momentary rate fluctuations, so that the bits that are saved (overspent) on the current image just slightly increment (decrement) the budget allocated to subsequent images, without really impairing the global performance of the communication.

Hence, we are interested in finding a solution to Eq. (10.19), subject to a constraint B' that is reasonably close to B_T. This slight difference dramatically simplifies the RD optimal bit allocation problem, because it allows the application of the Lagrange-multiplier method. We now state the main and fundamental theorem associated with Lagrangian optimization, because it sustains our subsequent developments.

Theorem: For any $\lambda \geq 0$, the solution $\mathbf{x}_\lambda^*$ to the unconstrained problem:

$$\mathbf{x}_\lambda^* = \arg\min_{\mathbf{x}} \left(\sum_{i=1}^{N} \mathrm{d}_{ix(i)} + \lambda \sum_{i=1}^{N} \mathrm{b}_{ix(i)} \right) \tag{10.21}$$

is also the solution to the constrained problem of Eq. (10.17) when the constraint:

$$B_T = \sum_{i=1}^{N} \mathrm{b}_{ix_\lambda^*(i)} \tag{10.22}$$

Proof: Let us define $D(\mathbf{x}) = \sum_{i=1}^{N} \mathrm{d}_{ix(i)}$ and $B(\mathbf{x}) = \sum_{i=1}^{N} \mathrm{b}_{ix(i)}$. By definition of $\mathbf{x}_\lambda$*, we have $D(\mathbf{x}_\lambda^*) + \lambda B(\mathbf{x}_\lambda^*) \leq D(\mathbf{x}) + \lambda\, B(\mathbf{x})$ for all $\mathbf{x} \in Q^N$. Equivalently, we have $D(\mathbf{x}_\lambda^*)$ - $D(\mathbf{x}) \leq \lambda\, B(\mathbf{x})$ - $\lambda B(\mathbf{x}_\lambda^*)$, for all $\mathbf{x} \in Q^N$. Hence, because $\lambda \geq 0$, for all $\mathbf{x} \in Q^N$ such that $B(\mathbf{x}) \leq B(\mathbf{x}_\lambda^*)$, we have $D(\mathbf{x}_\lambda^*)$ - $D(\mathbf{x}) \leq 0$. That is, $\mathbf{x}_\lambda$* is also the solution to the constrained problem when $B_T = B(\mathbf{x}_\lambda^*)$. □

This theorem says that to every nonnegative λ, there is a corresponding constrained problem whose solution is identical to that of the unconstrained problem. As we sweep λ from zero to infinity, sets of solutions $\mathbf{x}_\lambda$* and constraints $B(\mathbf{x}_\lambda^*)$ are created. Our purpose is then to find the solution which corresponds to the constraint that is close to the target bit-budget B_T.

We now explain how to solve the unconstrained problem. For a given λ, the solution to Eq. (10.21) is obtained by minimizing each term of the sum separately. Hence, for each codeblock *i*,

$$x(i)_\lambda^* = \arg\min_j \left(\mathrm{d}_{ij} + \lambda \mathrm{b}_{ij}\right) \tag{10.23}$$

Minimizing Eq. (10.23) intuitively corresponds to finding the operating point of the i^{th} codeblock that is "first hit" by a line of absolute slope λ (see the examples in Fig. 10.8). The convex-hull RD points are defined as the $(\mathrm{d}_{ij}, \mathrm{b}_{ij})$ pairs that sustain the lower convex-hull of the discrete set of operating points of the i^{th} codeblock.

For simplicity, we re-label the $M_H(i) \leq M$ convex-hull points, and denote $(\mathrm{d}_{ik}^H, \mathrm{b}_{ik}^H)$, $k \leq M_H(i)$ to be their rate-distortion coordinates. When sweeping the λ value from infinity to zero, the solution to Eq. (10.21) goes through the convex-hull points from left to right. Specifically, if we define $S_i(k) = \left(\mathrm{d}_{ik}^H - \mathrm{d}_{i(k+1)}^H\right) / \left(\mathrm{b}_{i(k+1)}^H - \mathrm{b}_{ik}^H\right)$ to be the slope of the convex-hull after the k^{th} point, the k^{th} point is optimal when $S_i(k\text{-}1) > \lambda > S_i(k)$, i.e. as long as the parameter λ lies between the slopes of the convex-hull on both sides of the k^{th} point.

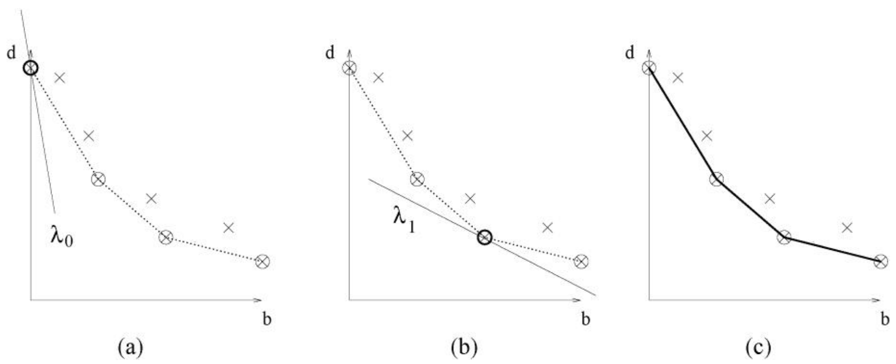


Fig. 10.8 Examples of Lagrangian-based bit allocation. In all graphs, the crosses depict possible operating points for a given codeblock. Circled crosses correspond to RD convex-hull points, which provide the set of solutions to the unconstrained bit allocation problem. (a) and (b) depict the 'first hit' solution for two distinct values of λ, while (c) plots the lower convex-hull.

At the image level, RD optimality is achieved by ensuring that each codeblock selects its operating point based on the same rate-distortion trade-off, as determined by the λ parameter. Since the λ slope is the same for every block, this algorithm is also referred to as a *constant slope optimization*. As illustrated in Section 10.3.3 and further described in (Ortega and Ramchandran 1998), the intuitive explanation of the algorithm is simple. By considering operating points at constant slope, all blocks are made to operate at the same marginal return for an extra bit in rate-distortion trade-off. By marginal return, we mean the incremental reduction of distortion obtained in return for an extra bit. Hence, the same marginal return for all blocks means that, the distortion reduction resulting from one extra bit for any given block is equal to the distortion increase incurred in using one less bit for another block (to maintain the same overall budget). For this reason, there is no allocation that is more efficient for that particular budget.

Now that we have solved the unconstrained problem, we explain how to find the solution whose constraint is close to the targer budget B_T. While reducing the value of λ, the optimal solution to Eq. (10.21) progressively moves along the convex-hull for each codeblock (e.g. going from λ_0 to λ_1 in Fig. 10.8), ending up in decoding more and more bit-planes. The process naturally covers the entire set of solutions to the unconstrained problem, in increasing order of bit consumption and image reconstruction quality. Under a budget constraint B_T, we are interested in the solution that maximizes the quality while keeping the bit-budget below the constraint.

Hence, given a bit-budget and the set of accessible convex-hull RD points for each codeblock, overall RD optimality is achieved at the image level by decoding the bit-planes corresponding to the convex-hull RD points selected in decreasing order of benefit per unit of rate, up to exhaustion of the transmission budget (Shoham and Gersho 1988). The approach is described in a JPEG2000 context in (Taubman and Rosenbaum 2003).

10.3.3 Lagrangian optimization: a non-image based example

To further illustrate the generality and intuition of Lagrangian optimization, we rephrase (Ortega and Ramchandran 1998) and consider an example that is outside the scope of image coding. This should hopefully highlight the general applicability of those solutions to resource allocation problems. The example is described as follows.

Nora is a student dealing with 2 questions during a 2-hour exam. Both questions worth 50% of the grade for the course, and we assume Nora is able to project her expected performance on each question based on how much time she devotes to them, as depicted in Fig. 10.9.

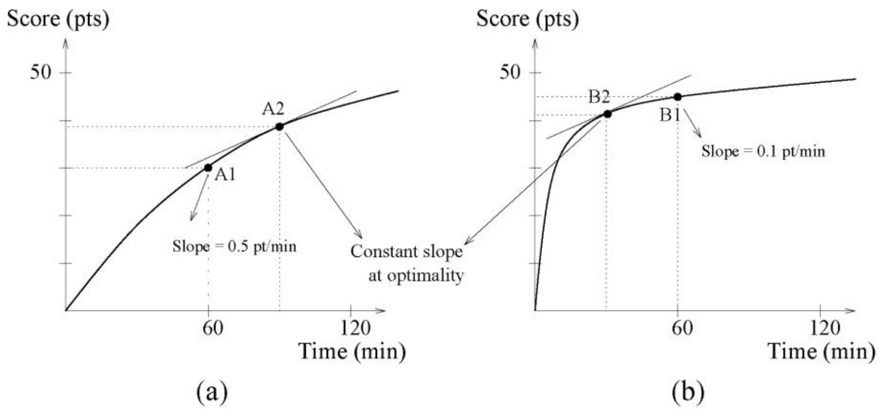


Fig. 10.9 Illustration of Lagrangian optimization. Graphs (a) and (b) depict the score-time trade-offs corresponding to 1st and 2nd questions of Nora's exam, respectively. Optimal overall score is obtained when the allocation of an additional time unit provides the same score increment for both questions.

Since the examination time is limited, Nora has to budget her time carefully. One option could be to devote half of the time to each question. This would amount to operating points A_1 and B_1 in Fig. 10.9, which results in an expected score of 75 (30 + 45). Can Nora make better use of her time? The answer lies in the slopes of the (Time, Score) trade-off curves that characterize both questions. Operating point A_1 has a slope of about 0.5 point/minute, while operating point B_1 has a slope of only 0.1 point/minute. Clearly, Nora could improve her score by diverting some time from the second to the first question. Actually, she should keep on stealing time from the second question as long as it provides a larger return for the first question than the corresponding loss incurred on the second question. At the optimum, the same marginal return (i.e.: score increment) would be obtained from any additional time spent on any question. This is exactly the operating points A_2 and B_2 in Fig. 10.9, which live on the same slope of the trade-off characteristics, and correspond to a complete allocation of the 2 hours budget, for an optimal score equal to 80 (38 + 42).

Here, it is worth emphasizing that the above reasoning relies on the convex nature of the time/score characteristics (see Section 10.3.2). We now consider a slightly different example to better capture the intuition lying behind the convex-hull constraint. In this example, one of the two questions is composed of a hierarchy of 4 dependent sub-questions. By dependent, we mean that a sub-question can only be answered if the answers to previous sub-questions are known. Fig. 10.10 depicts Nora's expected performance to such a hierarchical question, as a function of the time devoted to it. Since the points resulting from a correct sub-answer might not be in-line with the amount of time required to solve the corresponding sub-problem, the time/score trade-off might be non-convex. In our example, answers to sub-question A, B, C and D respectively worth 10, 10, 15 and 15 points, but are expected to require 10, 30, 20 and 60 minutes to be answered correctly and completely by Nora.

Solving the unconstrained time allocation problem $t^{*}= \text{argmax}_t(s + \lambda t)$ when the slope parameter λ sweeps from infinity to zero ends up in scanning the convex-hull (depicted with dots in Fig. 10.10) of the time/score characteristic. Hence, the Lagrangian framework will prevent considering non convex hull operating points as eligible solutions to the resource allocation problem. In Fig. 10.10, all operating points lying on the time/score characteristic between A and C become ineligible solutions to the unconstrained optimization problem, whatever the slope parameter is. Intuitively, those non-convex points correspond to cases for which time has been spent with relatively little return compared to what will be obtained by a longer effort. Those non-convex operating points should thus also be omitted by the iterative solution described in Section 10.3.2. This is the reason why it is crucial to first compute the trade-off characteristic convex-hull before running the iterative solution proposed in Section 10.3.2.

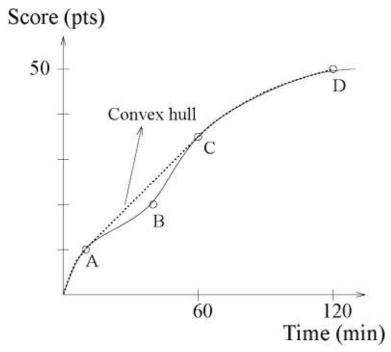


Fig. 10.10 Illustration of convex-hull approximation. Operating points lying between A and C on the time/score characteristic do not belong to the time/score convex-hull, and are thus ineligible solutions for the unconstrained allocation problem.

10.4 MATLAB proof of concept

This Section demonstrates the theoretical concepts previously introduced in this Chapter. A number of experiments are first presented to illustrate the multiresolution and decorrelative properties of the wavelet transform. In a second part, an image is selected and processed through a simplified JPEG2000 scheme.

All along this Section, compressed image quality are estimated based on mean squared error (MSE) or peak signal to noise ratio (PSNR) metrics. The MSE between an $N \times M$ image I and its approximation $\tilde{I}$ is equal to

$$MSE(I,\tilde{I}) = \frac{1}{MN}\sum_{m=1}^{M}\sum_{n=1}^{N}\left(I(m,n)-\tilde{I}(m,n)\right)^2 \qquad (10.24)$$

In turn, the PSNR is measured in decibels (dB), and is defined by

$$PSNR(I,\tilde{I}) \triangleq 10\log_{10}\left(\frac{Dynamic}{MSE(I,\tilde{I})}\right) \qquad (10.25)$$

where *Dynamic* denotes the dynamic range of the image and is equal to 255 for an 8 bits/pixel image.

10.4.1 Experiments with the Wavelet Transform

Four numerical experiments are presented to make the reader more familiar with the practical aspects of the 1-D and 2-D DWT implementations in MATLAB. The third and fourth experiments go slightly further this exploration by studying a very simple image compression procedure in the wavelet domain.

Experiment 1: Computing the Haar DWT

Let us present an example of DWT computation using the MATLAB Wavelet Toolbox. The following sequence of MATLAB commands performs the Haar DWT of our favorite 1-D signal (see Fig. 10.3) from resolution $J = 10$ to resolution $J_0 = 5$. Detail and approximation coefficients are represented with the `waveshow()` command, and are depicted in Fig. 10.11. Localization of important coefficients close to transient signal parts is now obvious.

```
%Loading the signal, N=1024=2^10
load 1d-sig;
% Performing the DWT,
J = 10; J0 = 5;
[W,L] = wavedec(sig, J-J0, 'haar');
% Showing it (waveshow.m is part of the CDROM)
figure; waveshow(sig,W,L)
```

Using the detail and approximation coefficients, we can also create the hierarchy of different approximation signals from resolution J0 = 5 to J = 10, which is the original signal itself. These signals are represented with the `appshow` () command, and are depicted in Fig. 10.12.

```
%% Showing the ladder of approximations
%% (appshow.m is part of the CDROM)
figure; appshow(sig,W,L, 'haar')
```

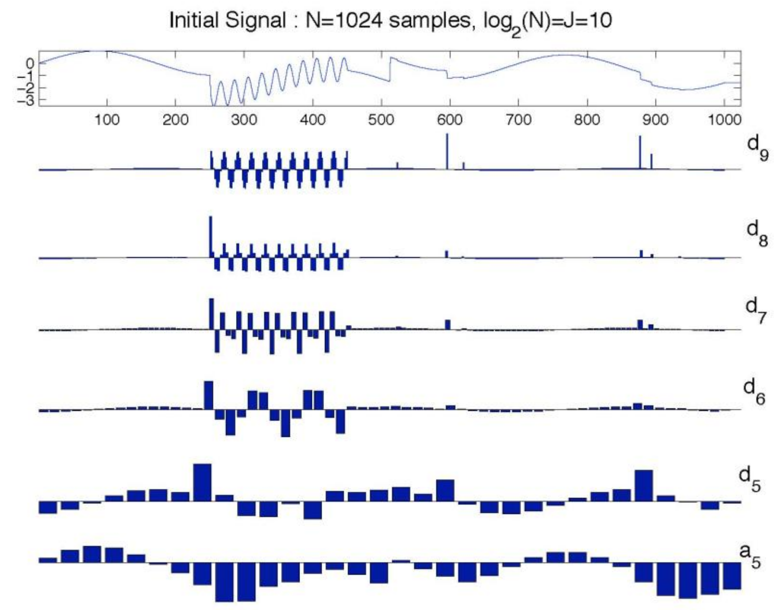


Fig. 10.11 Haar wavelet transform of the 1-D signal presented in Fig. 10.3.

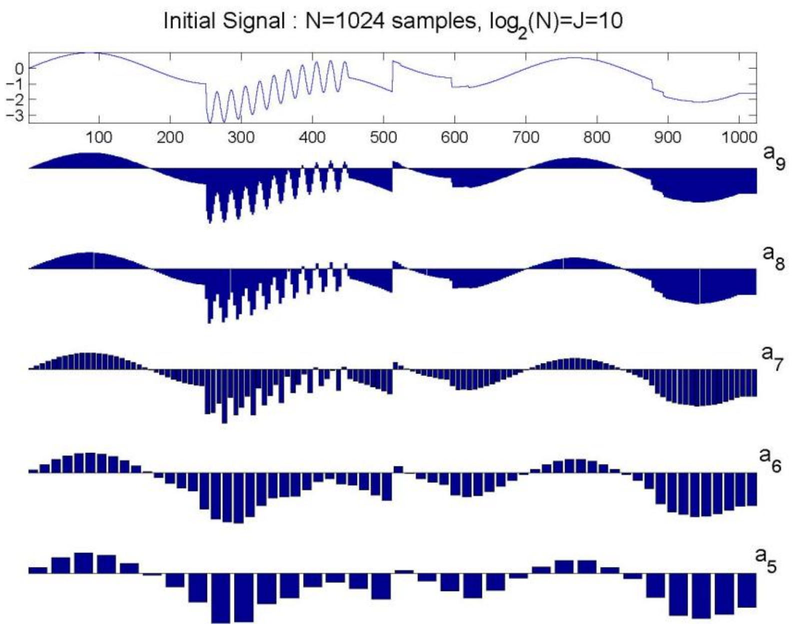


Fig. 10.12 Hierarchy of approximations obtained with the Haar wavelet transform of the 1-D signal presented in Fig. 10.3.

Experiment 2: Performing 2D-DWT

The following routine computes the Daubechies 9/7 2-D DWT of the input images (see Fig. 10.13), and displays the wavelet coefficients in the same way as explained in Fig. 10.6. In `waveshow2()`, horizontal, vertical and diagonal coefficients are shifted around 128 and normalized by their maximal absolute value at each resolution. The concept is illustrated in Fig. 10.14. Note that for visualization purposes, the pixel values of the various subbands have been normalized (i.e.: coefficients with zero value are represented with a 128 value). We observe that the most significant coefficients appear only in the transient parts, i.e. close to the edges or in the textures of the image (e.g. the grass in the bottom of Fig. 10.13.a).

```
% Loading the image : The Cameraman picture
% img is a 256x256 size array
im = double(imread('cameraman.tif'));
figure; imagesc(im); colormap(gray); axis equal tight;
set(gca,'clim',[7 253])
% 2D DWT Computations with Daubechies 9/7 (== 'bior4.4')
% W contains the DWT coefficients in a column shape.
J = log2(256); J0 = 2; wname = 'bior4.4';
[W,S] = wavedec2(im, J-J0, wname);
% The DWT array
figure, waveshow2(W,S,'haar');
```

Experiment 3: On the compression road.

We now focus on the “sparseness” of the wavelet coefficients for the representation of *natural* images. The relevance of the WT regarding compression holds in the following concept: *very few coefficients concentrate the essential of the image information*. This can already be perceived in Fig. 10.14. Conversely to initial pixel values, detail coefficients with high amplitude are not very numerous and are well localized on image edges.

The following code is the continuation of Experiment 2. Here we compress images by keeping only the 10% highest energy wavelet coefficients. This is achieved by sorting the wavelet values by decreasing order of magnitude and recording the amplitude of the K^{th} largest element (with K the closest integer to $N^2/10$). Then, all other wavelet coefficients with non zero magnitude are set to 0, i.e. their information is lost (see Fig. 10.15).

```
% Sorting wavelet coefficient amplitude
sW = sort(abs(W(:)), 'descend');

% Number of elements to keep
% Compression of 90% !
K = round(256*256/10);
T = sW(K);

% Thresholding values of W lesser than T
% i.e. we keep the K strongest
nW = W;
```

```
nW(abs(nW) < T) = 0;

% Rebuilding the compressed image
Timg = waverec2(nW, S, wname);
figure; imagesc(Timg); colormap(gray); axis equal tight;
set(gca,'clim',[7 253])
```

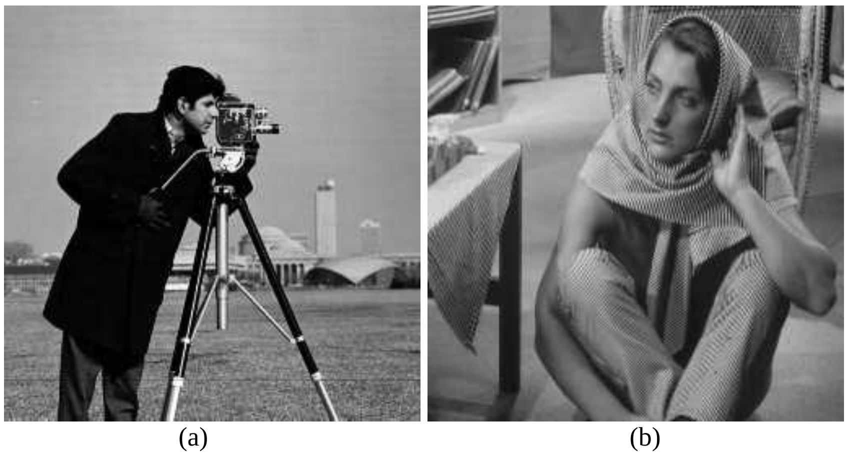


Fig. 10.13 (a) Original *Cameraman* image (256x256 pixels) and (b) Original *Barbara* image (512x512 pixels).

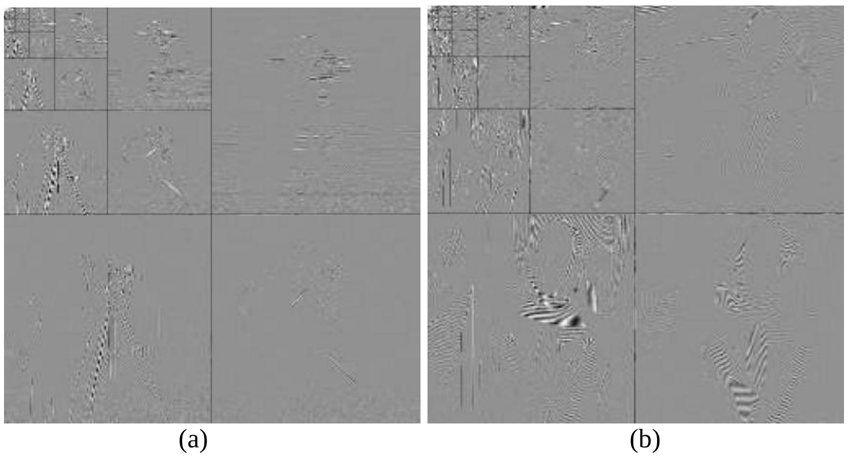


Fig. 10.14 2-D DWT of (a) *Cameraman* and (b) *Barbara* using Daubechies 9/7.

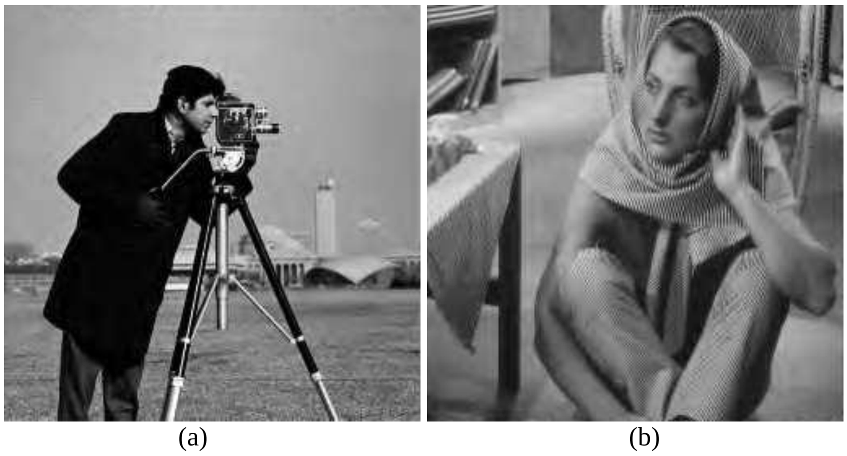


Fig. 10.15 Reconstruction of the previous images using the 10% of DWT coefficients with highest energy: (a) *Cameraman* (PSNR = 32.87 dB) and (b) *Barbara* (PSNR = 31.75 dB)

Experiment 4: Quantifying Compression Quality

Let us now quantify a bit further the quality reached by the compression scheme of Experiment 3 in function of both the number of DWT coefficients kept during the thresholding and the type of wavelet filters used.

In Fig. 10.16 the quality curve obtained for different percentages of wavelet coefficients is drawn (from 5% to 30%) for the *Cameraman* image of Fig. 10.13. We can clearly see that the Daubechies 9/7 filter provides the best quality. However, quality is not the only criterion which makes a filter better than another. Daubechies (or Legall) 5/3 filters can be expressed by rational numbers and, therefore, are used for *lossless compression* in JPEG2000.

```
nbpix = 256*256;

% Fixing the percentage of pixels to keep in the compression
% between 5% and 30%
K = round(((5:5:30)/100) * nbpix);
nbK = length(K);

% Cameraman Image decomposed on J levels
im = double(imread('cameraman.tif'));
J = log2(256); J0 = 0;

% Quality metrics between two images : MSE and PSNR.
% Assuming an 8 bits original image
MSE = @(X,Y) norm(X(:) - Y(:), 2)^2 / nbpix;
PSNR = @(X,Y) 10*log10( (256-1)^2 / MSE(X,Y) );

wname1 = 'bior4.4'; %% Daubechies 9/7
```

```
wname2 = 'bior3.3'; %% Daubechies/Legall 5/3

[W1,S1] = wavedec2(im, J-J0, wname1);
[W2,S2] = wavedec2(im, J-J0, wname2);

sW1 = sort(abs(W1(:)), 'descend');
sW2 = sort(abs(W2(:)), 'descend');

for k = 1:nbK,
   % Setting all the DWT coefficients smaller than the Kth
magnitude to
   % zero.
   % For DB97
   T1 = sW1(K(k));
   nW1 = W1;
   nW1(abs(nW1) < T1) = 0;
   Timg1 = waverec2(nW1, S1, wname1);

   % For DB53
   T2 = sW2(K(k));
   nW2 = W2;
   nW2(abs(nW2) < T2) = 0;
   Timg2 = waverec2(nW2, S2, wname2);

   % Recording quality
   curve_97(k) = PSNR(im, Timg1);
   curve_53(k) = PSNR(im, Timg2);
end
```

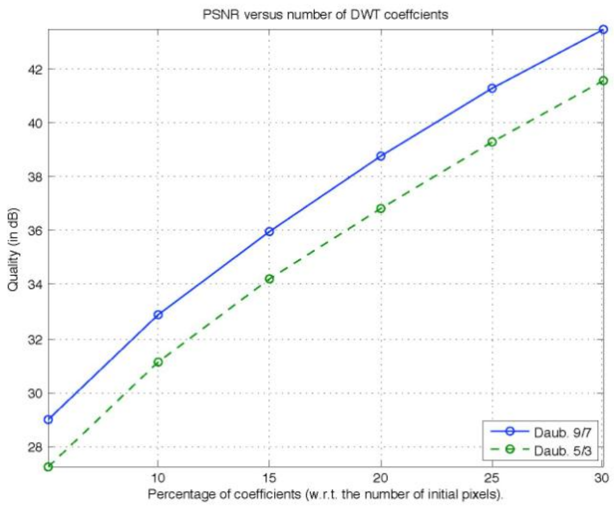


Fig. 10.16 Quality curve of compressed images (for the *Cameraman* image) for different percentage of DWT coefficients and for different filters.

10.4.2 A simplified JPEG2000 scheme

In this Section, we provide a proof of concept of the JPEG2000 image compression standard. We start by transforming the input image using 2D-DWT. We then consider context-based entropy coding of the wavelet coefficient bit-planes, and conclude by performing a rate-distortion optimal allocation.

All along this code, a structure `img` is defined that will contain the required settings to process the image. These settings include the path to the image to be processed (`img.path`), the number of wavelet decompositions to apply (`img.nwdec`), the kind of wavelet filters to use (`img.wfilt`), etc.

First of all, we load a greyscale image and shift the coefficients from an unsigned to a signed representation.

```
X = imread(img.path);
X = double(X);
img.bdepth = ceil(log2(max(X(:)+1)));
[img.h img.w] = size(X);
X = X - pow2(img.bdepth-1); % DC-level shifting
```

Discrete Wavelet Transform

The JPEG2000 standard defines 2 different filters, namely the 5/3 and the 9/7 transforms. The former is used in JPEG2000 when lossless coding is required. The latter, with a slightly higher decorrelating power, is used for lossy coding.

The norms of the synthesis filters[17] are denoted by `wnorm_53` and `wnorm_97`. They will be used to approximate square errors in the pixel domain based on the ones in the wavelet domain. According to the parameter `wfilt` defining the wavelet transform to be used, we apply the 5/3 or 9/7 transform. The function `wavedec2` from the Wavelet Toolbox is used.

```
if img.wfilt==0
    [C,S] = wavedec2(X,img.nwdec,lo_53_D,hi_53_D);
elseif img.wfilt==1
    [C,S] = wavedec2(X,img.nwdec,lo_97_D,hi_97_D);
else
    error('wavelet filter not recognized');
end
```

[17] The norm of a filter corresponds to the sum of the squared coefficients of the filter. It can be seen as the average amplitude change that will occur when filtering the signal.

Context-based modeling of coefficients bit-planes

After the DWT, the JPEG2000 algorithm performs bit-plane context-based entropy coding of each subband. This Section computes the incremental bit-budget and distortion reduction resulting from the addition of bit-planes to refine DWT coefficients. Thereby, it provides the inputs required by the rate-distortion optimal bit allocation mechanisms envisioned in the next Section. Moreover, this Section illustrates the advantage of context-based modelling by comparing two measures of the entropy associated with the binary representation of the wavelet coefficients. In the first case, the binary symbols are assumed to be generated by an i.i.d. sequence of random variables, and the probability distribution of each binary random variable is estimated based on the frequency of occurrence of 1's and 0's symbols. In the second case, we use the conditional probabilities associated with each binary symbol, knowing its context. The reduction of entropy between the first and second case corresponds to the benefit obtained from the chosen context model.

It should be noted that to avoid a too long Section, we do not actually implement the entropy coder. Only the performance of such coder is evaluated, through estimation of the source entropy. This estimation enables to compute the output rate that would be obtained for a given distortion level.

Before being entropy-coded, wavelet coefficients are quantized and mapped on a certain amount of bits. To do so, we first separate the sign and magnitude of the wavelet coefficients. They will indeed be encoded separately.

```
Csign = sign(C);
Cmagn = abs(C);
```

Then, coefficients are quantized: this quantization is different from the one that will implicitly occur later by dropping some of the least significant bit-planes, done by dividing them by a pre-defined quantization stepsize. In our case, the quantization stepsize chosen follows the rule used in the OpenJPEG library[18] and depends on the kind of subband and on the norm of the synthesis filter. The following code corresponds to the quantization stepsize for the LL subband.

```
if img.wfilt==0
    img.res(1).sb(1).qstep = 1;
elseif img.wfilt==1
    img.res(1).sb(1).qstep = 1/wnorm_97(1,img.nwdec+1);
end
```

[18] Open-source JPEG2000 codec from the TELE lab, UCL, Belgium: http://www.openjpeg.org

Eventually, quantized coefficients are mapped onto a certain amount of bits (fixed-point representation). In this simple experiment, we simply keep the integer part of the coefficients that we represent using 16 bits. For the LL subband, this is done through the following code (`Am` corresponds to the magnitude of the wavelet coefficients from the LL subband):

```
img.res(1).sb(1).coeff =
uint16(floor(Am/img.res(1).sb(1).qstep));
```

The code for the other subbands is similar and is therefore not reproduced here. As we can see, all quantized coefficients are stored in the `IMG` structure, resolution per resolution, and subband per subband.

In JPEG2000, wavelet coefficients are processed bit-plane by bit-plane and not coefficient by coefficient. Now that the coefficients are quantized and mapped onto a fixed number of bits, we can truly observe the scalability offered by such bit-plane representation. To illustrate this, we choose a subband, let's say the HL subband of last resolution, and display its *k* most significant bit-planes for *k* =1,.., *K* where *K* is the number of significant bit-planes for this subband:

```
sbcoeff = img.res(end).sb(1).coeff;
sbsign = img.res(end).sb(1).sign;
K = ceil(log2(max(double(sbcoeff(:)+1))));
```

It should be noted that *K* is not necessarily equal to the maximum number of bit-planes (16 in our experiment). Indeed, in most subbands and especially in those corresponding to the high frequencies, 16 bits will be far too much to represent the coefficients values. Consequently, many of the most significant bit-planes will often remain to zero. In practice, when encoding a subband, rather than encoding these all-zero bit-planes, we skip them until the first ‘1’ bit is encountered. The number of skipped bit-planes will then simply be stored in a header of the compressed bitstream.

The truncation of the coefficients is done by applying successively a “AND” mask and an “OR” mask to the wavelet coefficients. The first one sets the required number of least significant bits to ‘0’. The second one moves the truncated coefficient value to the middle of the truncation step by setting the most significant truncated bit to ‘1’.

```
mask_AND = bitshift(uint16(65535),nbp_discard);
mask_OR = bitshift(uint16(1),nbp_discard-1);
…
m_trunc = bitor(bitand(sbcoeff,mask_AND),mask_OR);
```

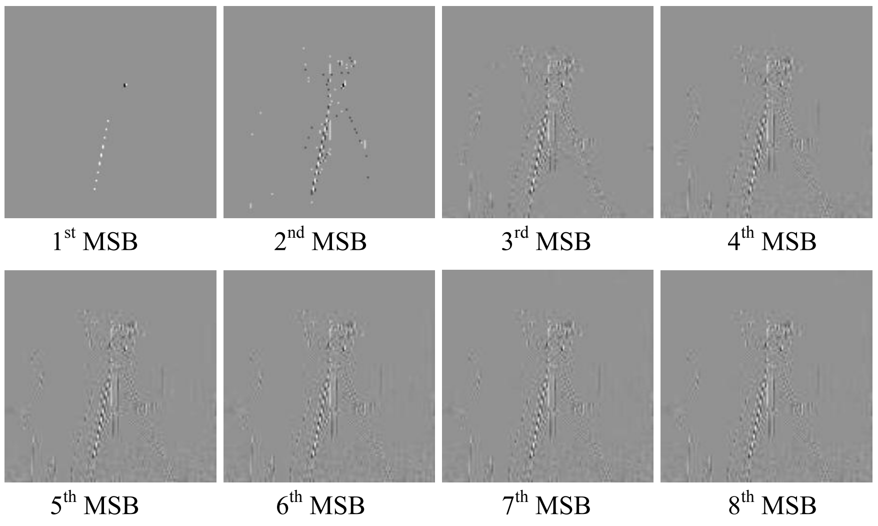


Fig. 10.17 Progressive refinement of wavelet coefficients through inclusion of an increasing number of bit-planes (MSB: most significant bit).

As we see in Fig. 10.17, wavelet coefficients are progressively refined, as the bit-planes (from the most to the least significant one) are included in the coefficient estimation.

We now compute the probability distribution of the binary representation of our image. This distribution will then be exploited to compute the related entropy.

As explained above, we will compare two kinds of distribution. The first one is based on the frequency of occurrence of ‘1’ and ‘0’ symbols over the whole image. In the second case, the conditional probabilities of binary random variables are estimated, knowing their context. The relevance of such an approach has been intuitively justified in Section 10.2.2.

Practically, the context of a bit corresponds to a set of state variables related to (i) the coefficient to whom the bit belongs, and (ii) its neighbouring coefficients. In this experiment, two state variables were used.

1. The *significant status* of a coefficient. Let us remind that a coefficient is said to become significant in a given bit-plane if a '1' bit is encountered for the first time for this coefficient (all other more significant bits were '0's).
2. The *first refinement status* of a coefficient. Among already significant coefficients, we will distinguish those that became significant in the previous (more significant) bit-plane.

In this proof-of-concept, we have considered twelve different contexts. They are presented in the preamble of function `get_context` and are actually a subset of the contexts used in the JPEG2000 standard (which uses 19 different contexts). Nine out of the twelve contexts are used to code not yet significant coefficients, while the three last ones are used for already significant coefficients. The sign of each coefficient is introduced "as is" in the codestream and is not entropy-coded.

Let us first initialize the number of contexts and the Context Distribution Table (CDT). This vector stores the probability of having a '1' for each context. The last element of the vector is used to store the global probability of getting a '1' on the whole image.

```
global nctxt;
nctxt = 12;
CDT = zeros(nctxt+1,1);
```

As explained in Section 10.2.1, before being entropy-coded, subbands are divided in small entities called codeblocks. Each codeblock will then be entropy-coded separately, starting from the most significant bit-plane to the least significant one.

In the code, several `for` loops are embedded so that each subband from each resolution level is processed and divided in such codeblocks.

```
for resno=1:numel(img.res)
    for sbno=1:numel(img.res(resno).sb)
        ...
        for y0=1:img.cbh:size(coeff,1)
            for x0=1:img.cbw:size(coeff,2)
```

Then, each codeblock is analyzed. First, as explained above, the number of all-zero most significant bit-planes is computed (`NBPS` field of each codeblock).

```
cb.nbps = ceil(log2(max(double(cb.coeff(:)+1))));
```

Now, each codeblock is analyzed through the `analyze_cb` function.

```
[cb.ctxt,cb.disto] = analyze_cb(cb);
```

This function will process each bit-plane starting from the first most significant non-zero bit-plane and return two variables:

(1) a matrix `ctxt` that gives two values for each context and each bit-plane : (i) the total number of bits and (ii) the number of '1' bits,

(2) a vector `disto` that computes, for each bit-plane `BP`, the square error between the original codeblock and the codeblock whose least significant bit-planes are truncated, starting from and including bit-plane BP.

This `disto` vector will be used to decide where to truncate the codeblock when performing the R-D allocation. However, it has to be adapted to reflect the square error in the pixel domain. To do so, the values of the `disto` vector (that correspond to the square errors between original and truncated wavelet coefficients) are multiplied by two factors. The first one is the squared stepsize that was used for the quantization that took place after the wavelet transform. By doing so, we actually perform an inverse quantization to get the square errors back in the dynamic range obtained just after the wavelet transform. Then, these “dequantized” square errors are multiplied by the norm of the corresponding synthesis filter. As explained at the beginning of Section 10.4.2, this last operation gives the corresponding square errors in the pixel domain, which are the ones of interest as we want precisely to approximate the distortion reduction in this domain.

```
if img.wfilt==0
    cb.disto=cb.disto.*((img.res(resno).sb(sbno).qstep)^2*...
               (wnorm_53(cb.sbtype,img.nwdec+2-resno))^2);
elseif img.wfilt==1
    cb.disto=cb.disto.*((img.res(resno).sb(sbno).qstep)^2*...
               (wnorm_97(cb.sbtype,img.nwdec+2-resno))^2);
end
```

Once matrix `ctxt` has been computed for each codeblock, we can easily compute a Context Distribution Table (`CDT`) that will be used for entropy coding. The `CDT` stores for each of the 12 contexts the probability to get a '1' bit according to the processed image. A 13th value stores the global probability to get a '1' bit, independently from the neighbourhood of the coefficient. These values are presented in Fig. 10.18.

The `CDT` is then saved so that it can be re-used when encoding other images.

```
save(CDTpath,'CDT');
```

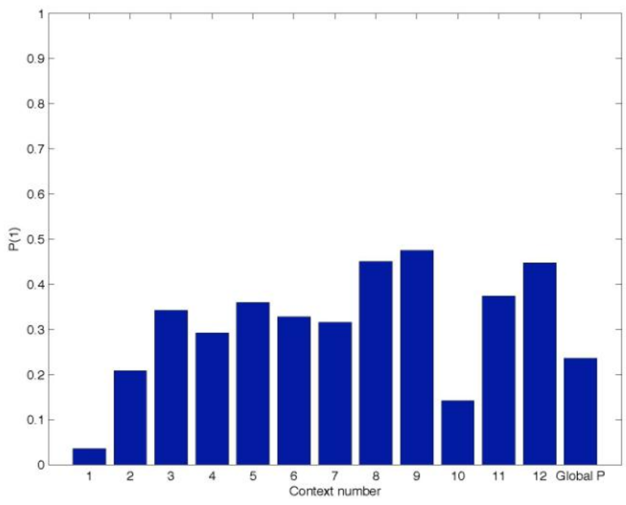


Fig. 10.18 Comparison of conditional probability of getting a '1', knowing the context of the bit, with the global probability, independent from the context.

Once the `CDT` has been computed, we can use it to estimate the entropy of each codeblock. If `img.useownCDT = 0`, we do not use the CDT computed on the processed image but the one specified in `img.CDT2use`. It should be noted that using a context distribution computed on an image whose content is very different from the one of the processed image does not change the performances drastically. This can be explained by the fact that the context mainly characterizes local image features, which reduces their dependency on the global appearance and statistics of the image. Moreover, in our proof-of-concept example, the conditional distributions are computed on all bit-planes of all subbands, which further reduces the dependency of those distributions on the global image features. In JPEG2000 however, the distribution is re-initialized for each codeblock and is dynamically adapted while encoding the bit-planes. In this case, the entropy coder uses a more adapted distribution and is therefore more efficient.

Practically, for each subband, we progressively fill in a matrix `RD`: `img.res(resno).sb(sbno).RD(cbno,bpno,i)` where `i` =1 for the rate values and `i` = 2 for the distortion values. Each pair of values (`i` =1,2) in the matrix RD gives therefore the amount of bits that are needed to encode a given bit-plane from a given codeblock in the processed subband (R), and the distortion reduction that this bit-plane brings when decoding the image (D). As all codeblocks do not have the same number of significant bit-planes in a subband, dimension 2 of matrix RD (counting the bit-planes) is taken equal to the maximum number of significant bit-planes among the codeblocks from the subband. Note that `resno=1` for the smallest resolution and `bpno=1` for the most significant bit-plane.

```
if img.useownCDT==0
    CDT=load(img.CDT2use);
    …
end
```

Nested in loops on resolutions, subbands, codeblocks and bit-planes, two simple functions are applied to fill in the RD matrix.

```
[rc rnc]=get_rate(ctxt,coeff,bpno,CDT);
RD(cbno,offset+bpno,1)=rc;
RD(cbno,offset+bpno,2)=get_disto(disto,bpno);
```

The `get_rate` function computes the entropy of each bit-plane based on the given `CDT`. It returns the rate `rc` that would be obtained after a context-based entropy coding and the rate `rnc` that would be obtained after a non-context-based entropy coding (i.e. with a global probability distribution on the whole image). To do so, it simply uses the entropy, which gives

$$\sum_{c=1}^{nctxt}\Big(-n_0(c)\log_2\big(p(0\,|\,c)\big)-n_1(c)\log_2\big(p(1\,|\,c)\big)\Big) \qquad (10.26)$$

where *nctxt* is the total number of context (12 in our case), $n_0(c)$ and $n_1(c)$ are the number of '0' symbols and '1' symbols with context *c*, respectively. This simply translates in MATLAB code as:

```
entropy = sum(-(log2(CDT(1:end-1))).*ctxt(:,2) ...
          -(log2((1-CDT(1:end-1)))).*(ctxt(:,1)-ctxt(:,2)));
```

The `get_disto` function computes the difference between square errors obtained with successive bit-planes.

```
if bpno<numel(disto)
    D = disto(bpno)-disto(bpno+1);
else
    D = disto(end);
end
```

The matrix `RD` of each subband will then be used in the next Section to find for each codeblock the best truncation point, that is, the one that will minimize the distortion for a given global bit budget.

Here we simply use an intermediary matrix `R_res` to store and compare the global compression performance obtained with and without context. Fig. 10.19 shows the compression ratio obtained for each resolution level. The total number of uncompressed bits taken into account to compute these values is the number of bits truly processed by the entropy coder, i.e. wavelet coefficients excluding the non-significant bit-planes. Fig. 10.20 computes the global compression ratio, comparing the original number of bits in the pixel domain, with the rate obtained after the entropy coding step. As expected, the context-based approach is more efficient than the other one, as it more accurately estimates the probability of getting a '1' or a '0' depending on the coefficient location. In particular, in Fig. 10.19, we see that the non-context-based entropy coding actually expands the LL subband rather than compresses it. This is because the global probability distribution used in this case is very different from the one really observed for this resolution level. On the contrary, when using contexts, and even if those contexts do not take the resolution level into account, the entropy coder still achieves compression on the low frequencies.

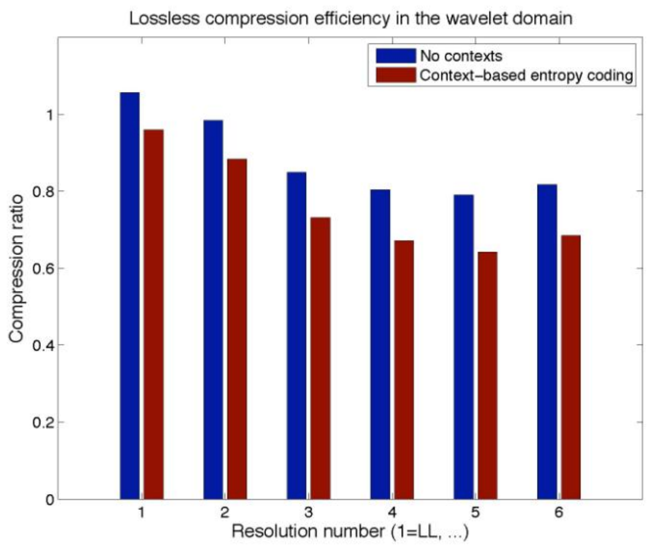


Fig. 10.19 Bar-graph plotting the compression ratio obtained for each resolution level, with and without contexts.

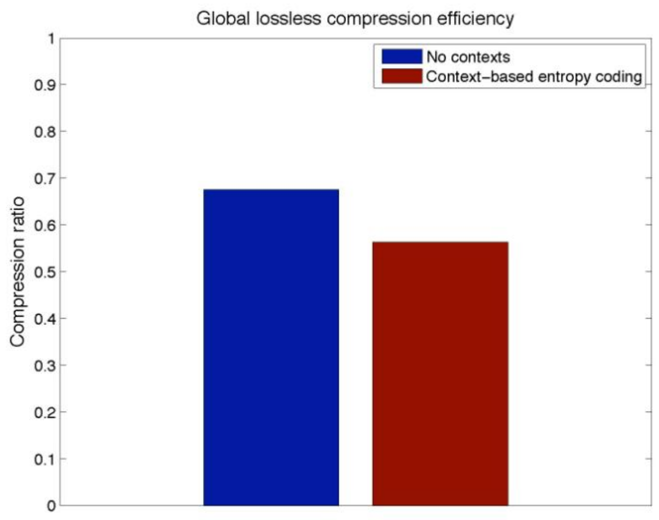


Fig. 10.20 Bar-graph presenting the global lossless compression efficiency.

Rate-Distortion optimal allocation

In this section, we demonstrate the benefit of rate-distortion optimal bit allocation. We consider an image whose wavelet coefficients have been split into codeblocks, and compare a naive and RD optimal bit allocation. The set of coefficients of a codeblock are encoded bit-plane by bit-plane, to define `bpno` (d_{ij}, b_{ij}) pairs, corresponding to the `bpno` rate-distortion trade-offs resulting from the approximation of the coefficients of the i^{th} codeblock by its j^{th} most significant bit-planes, with $0 \le j <$ `bpno`. The envisioned naive bit allocation strategy simply assigns a constant number of bit-planes to all image codeblocks. In contrast, the RD optimal strategy first computes the convex-hull RD points and selects them in decreasing order of distortion reduction per cost unit.

As previously presented, `img.res(resno).sb(sbno).RD(i,j,k)` denotes a structure that conveys, for all image resolution and image sub-band, the incremental reduction of distortion and the increase of bits corresponding to the addition of the j^{th} bit-plane to the definition of codeblock i. Index k is used to differentiate the cost in bits ($k = 1$) from the decrease in distortion ($k = 2$).

To implement the RD optimal bit allocation method, the *hull structure* `HS` is defined to collect the convex-hull RD points of every codeblock in the image. For each convex-hull RD point, the `HS` structure records the decrease of distortion per bit unit provided by the bit-planes corresponding to the convex-hull RD point. It also records the codeblock resolution, subband, index, and number of bit-planes corresponding to the convex-hull RD point.

```
% Loop on codeblocks of resolution i and subband j.
for k=1:size(img.res(i).sb(j).RD,1)
  % Number of bitplanes for the codeblock.
 nbbp_cb = size(img.res(i).sb(j).RD(k,:,:), 2);
 % Number of bitplanes corresponding to the last
 % operating point found on the convex-hull.
 lhbp=0;
 % To enter in the while loop.
 gain_max=1;
 while (lhbp<nbbp_cb && gain_max>0)
  nhbp=lhbp;
  gain_max=0;
  for bp=(lhbp+1):nbbp_cb
   gain=sum( img.res(i).sb(j).RD(k,(lhbp+1):bp,2) )
              / sum( img.res(i).sb(j).RD(k,(lhbp+1):bp,1) );
   if gain > gain_max
     gain_max=gain;
     nhbp=bp;
   end
  end
  nbHS=nbHS+1;
  deltaR = sum( img.res(i).sb(j).RD(k,(lhbp+1):nhbp,1) );
  deltaD = sum( img.res(i).sb(j).RD(k,(lhbp+1):nhbp,2) );
  HS(nbHS,:)=[gain_max,i,j,k,nhbp, deltaR, deltaD];
  lhbp=nhbp;
 end % End while loop.
end % End for loop.
```

Once the `HS` structure has been defined, the RD optimal allocation strategy simply consists in selecting the convex-hull RD points in decreasing order of distortion reduction per cost unit. Hence, the `HS` structure is sorted to control the bit-planes allocation strategy for a target-rate `R_T` as follows:

```
ascendingHS = sortrows(HS);
indHS = size(HS,1);

Rtmp = 0;
while (Rtmp < R_T && indHS>0)
       Rtmp = Rtmp + ascendingHS(indHS,6);
       Dtmp = Dtmp - ascendingHS(indHS,7);
       img.res(ascendingHS(indHS,2)).sb(ascendingHS(indHS,3))
                .nbrplanes(ascendingHS(indHS,4)) =
ascendingHS(indHS,5);
       indHS = indHS - 1;
end
```

In this MATLAB code sample, `Rtmp` and `Dtmp` respectively define the global image bit-budget and distortion, while the variable `img.res(i).sb(j).nbrplanes(k)` records the number of bit-planes allocated to the k^{th} codeblock of the j^{th} subband of the i^{th} resolution.

On the MATLAB CDrom, the global image RD trade-offs obtained when naively allocating a constant number of bit-planes to each codeblock are compared with the RD points obtained when RD optimal allocation targets the same bit-budgets as the ones obtained for naive allocation. In Fig. 10.21, we present a comparison of the RD points achieved using both allocation strategies.

MATLAB code is also provided to reconstruct the image based on the number of bit-planes allocated to each codeblock. It permits to compare both allocation methods from a perceptual point of view. Reconstructed images using naive and optimum bit allocation strategies are presented in Fig. 10.22 and Fig. 10.23, respectively. It is clear that the visual quality of the optimum bit allocation version largely outperforms that of the naive allocation for both cases[19].

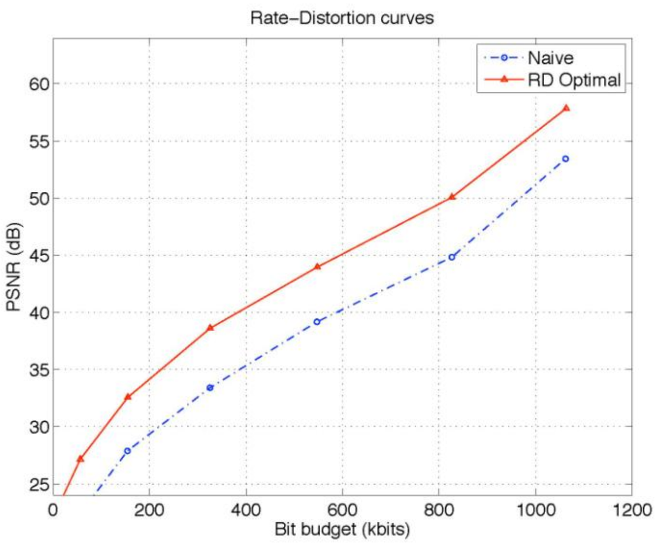


Fig. 10.21 Rate-Distortion curves comparing the naïve and the optimal bit allocation strategies.

[19] Recall that Cameraman size is 256x256 while Barbara size is 512x512; hence, the so different target rates

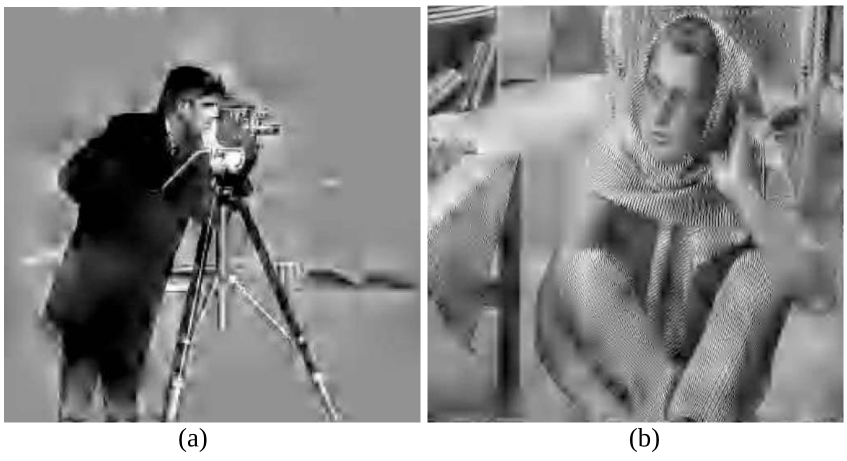


Fig. 10.22 Reconstructed version, using naïve bit allocation, of the image (a) *Cameraman* at 9.4 kbits (PSNR = 26.06 dB) and (b) *Barbara* at 65.56 kbits (PSNR = 20.59 dB).

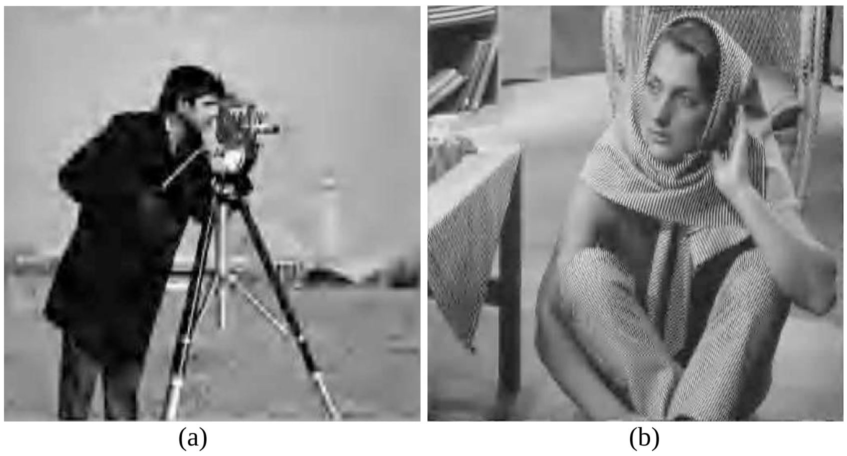


Fig. 10.23 Reconstructed version, using optimal bit allocation, of the image (a) *Cameraman* at 9.4 kbits (PSNR = 30.67 dB) and (b) *Barbara* at 66.00 kbits (PSNR = 25.86).

10.5 Going further: From concepts to compliant JPEG2000 codestreams

To complete this Chapter, Fig. 10.24 presents the entire pipeline to generate a JPEG2000-compliant codestream.

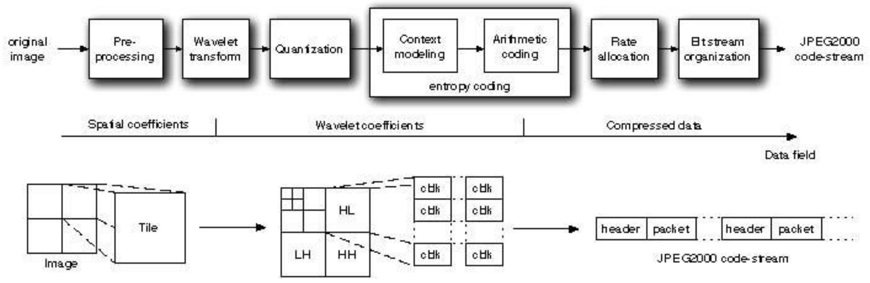


Fig. 10.24 The JPEG2000 coding steps: Pixels are first transformed in wavelet coefficients. Then, the various subbands from each resolution level are divided into codeblocks that are independently entropy-coded. Eventually, entropy-coded data are distributed in packets that compose the final JPEG2000 codestream.

We recognize the wavelet transform, the bit-plane quantization, the context modelling, and the rate allocation mechanisms described above. For completeness, we now survey the additional stages involved in the pipeline.

Three operations can performed during the initial pre-processing step. First of all, the image can be split into rectangular blocks called tiles, which will be compressed independently. This is particularly useful for applications with limited memory resources. Then, if pixels are represented by unsigned values, they are shifted to get the corresponding signed values. Eventually, in case of an image made of several components, an inter-component decorrelation can be applied. Typically, the RGB to YCbCr transformation is exploited to increase the subsequent compression efficiency by reducing the correlation between components.

Regarding *arithmetic coding*, as explained in Section 10.2, the MQ-coder (Mitchell and Pennebaker 1988) encodes the bits according to the probability distribution estimated by the modeller, and progressively generates codestream segments. It is worth mentioning here that JPEG2000 further refines the way bits are scanned within a bit-plane, so as to first encode the bits for which a large benefit in quality is expected per unit of rate. This process is based on the observation of already encoded bit-planes, and the algorithm used to select the bits that will be encoded first is named *Embedded Block Coding with Optimized Truncation* (EBCOT) (Taubman 2000) by the JPEG2000 community.

Note that when decoding an image, the context modelling part only provides the context to the arithmetic coding part, and waits for the decoded bit. This bit is computed by the MQ-decoder that progressively consumes the compressed bitstream.

The last step involves *packetization* and *bitstream organization*. Once all bit-planes have been encoded, rate-distortion optimal allocation is considered for a set of increasing target bit-budgets, or equivalently for a decreasing sequence of λ parameters. As explained in Section 10.3, the sequence of corresponding RD optimal allocations progressively adds bit-planes to the image codeblocks.

In JPEG2000, the incremental contributions added from one target rate to another (or from one Lagrangian parameter to another) are grouped in the so-called *quality layers*. Several quality layers are thus defined for an image, corresponding to distinct rate-distortion trade-offs.

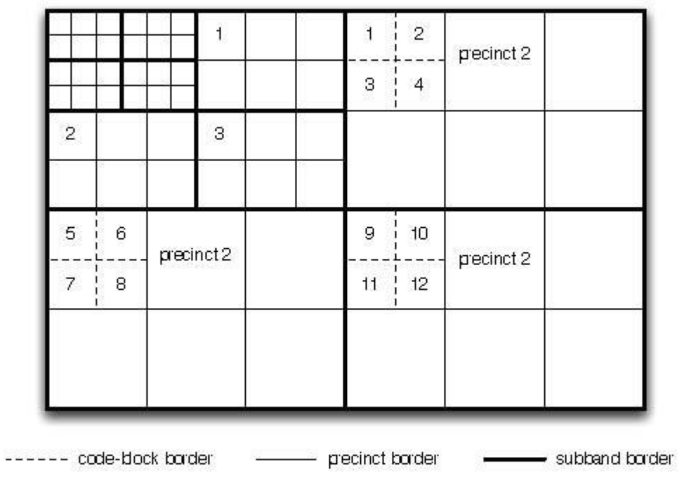


Fig. 10.25 Precinct and codeblocks subdivisions: A precinct includes all codeblocks belonging to a given resolution level and covering a given spatial area. For example, in the highest resolution, the first precinct is made of 12 codeblocks, 4 from each subband. The size (i.e. the number of pixels) of the area covered by a precinct can vary from one resolution level to another. This is the case in the figure where this area size has been adapted in each resolution level so as to have the same number of precincts (6) in each level.

Once the rate allocation step has distributed the incremental contributions from all codeblocks over the specified quality layers, the compressed data is divided in packets. A packet contains the information related to a certain quality layer from a certain resolution, in a certain spatial location of one of the image components. A spatial location is called a *precinct* and corresponds to the set of codeblocks from all subbands of a given resolution, and covering a given spatial area (see Fig. 10.25). Packets, along with additional headers, form the final JPEG2000 codestream.

10.6 Conclusion

Although quite elaborated in its functional capabilities, the JPEG2000 image compression standard relies on the combination of rather simple basic principles: discrete wavelet transform, context-based entropy coding, and rate-distortion optimal allocation mechanisms. This combination leads to a flexible and powerful image compression standard that is mainly used in professional markets requiring a high quality level and a solid intrinsic scalability.

Various software implementations of the JPEG2000 standard are available. Among them, we should cite the OpenJPEG library (open-source C-library, http://www.openjpeg.org), the Jasper project (open-source C-library, http://www.ece.uvic.ca/~mdadams/jasper), Kakadu (C++ commercial library, http://www.kakadusoftware.com), and JJ2000 (freely available java implementation, http://jj2000.epfl.ch/).

For completeness, it is also worth mentioning that several extensions have been defined in addition to the JPEG2000 core coding system. Among them, we can cite a specific communication protocol that enables efficient and flexible remote access to JPEG2000 content (JPIP, Taubman and Prandolini 2003, Prandolini 2004), additional error protection techniques that increases the robustness of JPEG2000 transmissions in wireless environment (JPWL), tools that allow applications to generate and exchange secure JPEG2000 codestreams (JPSec), extensions to address 3D and floating point data (JP3D), etc. We invite the interested reader to visit www.jpeg.org/jpeg2000 for more information on this topic.

10.7 References

Boliek M., Christopoulos C., and Majani, E., JPEG2000 image core coding system (Part 1). Technical report, ISO/IEC JTC1/SC29 WG1, July 2001

Daubechies I., Ten Lectures on Wavelets. Society for Industrial and Applied Mathematics, 1992. ISBN 0-89871-274-2.

DCI. Digital Cinema System Specifications. Digital Cinema Initiatives (DCI), March 2005. URL http://www.dcimovies.com/.

Foos D.H., Muka E., Slone R.M., Erickson B.J., Flynn M.J., Clunie D.A., Hildebrand L., Kohm K.S., Young S.S., JPEG2000 compression of medical imagery. Proceedings of SPIE, 3980:85, 2003.

Fossel S., Fottinger G., Mohr J., Motion JPEG2000 for high quality video systems. Consumer Electronics, IEEE Transactions on, 49(4):787–791, 2003.

Kellerer H., Pferschy U., Pisinger D.. Knapsack problems. Springer Verlag, 2004. ISBN 3-540-40286-1.

Janosky J., Witthus R.W., Using JPEG2000 for Enhanced Preservation and Web Access of Digital Archives. In IS&T's 2004 Archiving Conference, pages 145–149, April 2004.

Mallat S., A Wavelet Tour of Signal Processing. Academic Press., 2nd edition, 1999.

Marpe D., George V., Cycon H.L., Barthel K.U.,. Performance evaluation of Motion-JPEG2000 in comparison with H. 264/AVC operated in pure intracoding mode. Proceedings of SPIE, 5266:129–137, 2003.

Mitchell J.L., Pennebaker W.B., Software implementations of the Qcoder. IBM J. Res. Develop., 32(6):753–774, November 1988.

Ortega A., Ramchandran K., Vetterli M., Optimal trellis-based buffered compression and fast approximation. IEEE Trans. on Image Processing, 3(1): 26–40, January 1994.

Ortega A., Optimal bit allocation under multiple rate constraints. In Data Compression Conference, pages 349–358, Snowbird, UT, April 1996.

Ortega A., Ramchandran K., Rate-distortion methods for image and video compression. IEEE Signal Processing Magazine, 15(6):23–50, November 1998.

Prandolini R., 15444-9:2004 JPEG2000 image coding system - part 9: Interactivity tools, apis and protocols. Technical report, ISO/IEC JTC1/SC29 WG1, March 2004.

Rabbani M., Joshi R., An overview of the JPEG2000 still image compression standard. Signal Processing: Image Communication, 17(1):3–48, January 2002.

Santa-Cruz D., Grosbois R., Ebrahimi T., JPEG2000 performance evaluation and assessment. Signal Processing: Image Communication, 17(1):113– 130, January 2002.

Shoham Y., Gersho A., Efficient bit allocation for an arbitrary set of quantizers. IEEE Trans. on Signal Processing, 36(9):1445–1453, September 1988.

Skodras A., Christopoulos C., Ebrahimi T., The JPEG2000 still image compression standard. Signal Processing Magazine, IEEE, 18(5):36–58, 2001.

Smith M., Villasenor J., Intra-frame JPEG-2000 vs. Inter-frame Compression Comparison: The benefits and trade-offs for very high quality, high resolution sequences. SMPTE Technical Conference, Pasadena, CA, Oct, 2004.

Symes P., JPEG2000, the Professional compression scheme. Content Technology Magazine, 3(3), June 2006. URL http://svc126.wic003tv.server-web.com/CT-pdf/CT-May-June-2006.pdf.

Taubman D., High performance scalable image compression with ebcot. IEEE Trans. on Image Processing, 9(7):1158–1170, July 2000.

Taubman D., Marcellin M.W., JPEG2000: Image Compression Fundamentals, Standards and Practice. Kluwer Academic, Boston, MA, USA, 2002.

Taubman D., Prandolini R., Architecture, philosophy and performance of JPIP: Internet protocol standard for JPEG2000. In International Symposium on Visual Communications and Image Processing (VCIP), Lugano, Switzerland, July 2003.

Taubman D., Rosenbaum R., Rate-distortion optimized interactive browsing of JPEG2000 images. In IEEE International Conference on Image Processing (ICIP), September 2003.

Tzannes A., Ebrahimi T., Motion JPEG2000 for Medical Imaging. ISO/IEC wg1n2883, Medical Imaging Ad Hoc Group, 2003.

Wolsey L., Integer Programming. Wiley, 1998.

Zhang D.R., Wang X.X., The manipulation approach of JPEG2000 compressed remote sensing images. Proc. SPIE, 6044:315–324, 2005.