République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

UNIVERSITÉ DE BATNA-2
Faculté de Technologie
Département d'Électronique

THÈSE

Présentée pour l'obtention du diplôme de

DOCTORAT EN SCIENCES

Spécialité : Électronique

Option : Contrôle Industriel

Par

Abdelhamid MESSAOUDI

THÈME

Contribution à l'amélioration et la mise en œuvre de nouveaux algorithmes pour la compression des signaux

Thèse soutenue publiquement le : **Mardi** *04/07/2017*

Devant le Jury :			
SAIDI Lamir	Prof.	Université Batna -2	Président
SRAIRI Kamel	Prof.	Université Biskra	Rapporteur
OUAFI Abdelkrim	M.C.A.	Université Biskra	Examinateur
ABDOU Latifa	M.C.A.	Université Biskra	Examinateur
AISSI Salim	M.C.A.	Université Batna -2	Examinateur

Colour image compression algorithm based on the DCT transform using difference lookup table

A. Messaoudi✉ and K. Srairi

A simple and efficient method for lossy colour image compression is proposed. Here, the discrete cosine transform (DCT) is applied to the YCbCr image obtained from the original RGB image. The bisection method is used to define the required threshold for a prefixed user peak signal to noise ratio as a controlled quality criterion. The thresholded and quantised DCT coefficients are encoded with a new technique. The proposed technique uses the difference of the indexes of the retained coefficients in coordination with DCT block adaptive scanning to encode efficiently the coefficients. The difference of the indexes is stored in a lookup table called (dLUT). When compared with recent methods, the obtained results show that the proposed algorithm achieves high performance.

Introduction: The compression can lead to an efficient transmission and storage of images. It reduces the size of the images by decreasing the redundancy that contains. In general, there are two kind of compression: lossless and lossy compression. The lossless compression guaranties that the original and reconstructed images are identical. The efficiency and the compression ratios are weak compared with those of the lossy compression. In this last, the quality of the reconstructed image is degraded to an acceptable level in the benefit of high compression ratios.

The lossy compression of the colour image has been widely studied and it can be further categorized in two principle methods [1]:

- The direct methods act directly on the image samples as in block truncation [2] and vector quantisation based techniques [3].
- The transform methods use transforms to concentrate the energy of the image in a few number of coefficients. Many transforms are used such as principal component analysis [4], discrete cosine transform (DCT) [5] and wavelets [6].

In this Letter, we propose a new colour image compression algorithm based on DCT, adaptive scanning technique [5] and a new efficient method encoding. As well known, the YCbCr transform reduces the correlation between the R, G and B spaces, thus a pre-processing step is effectuated on the original image. The bisection algorithm [5] based on the peak signal to noise ratio (PSNR) criterion is used to threshold the DCT coefficients. A quantisation and entropy encoding steps are performed to complete the algorithm.

Method presentation: The compression algorithm scheme is composed from several steps as depicted in Fig. 1. The RGB colour space is widely used in computer graphics. The three planes R, G and B contain an important redundancy which can be reduced by conversion to another colour space such as YCbCr [5].

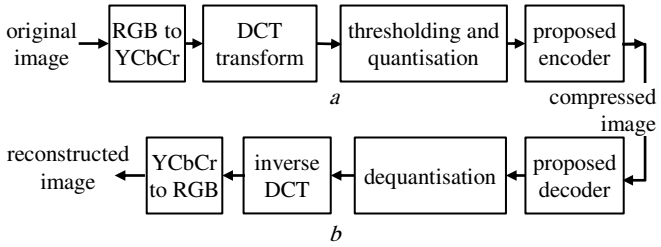


Fig. 1 *Compression algorithm scheme*
a Compression phase
b Decompression phase

The RGB to YCbCr conversion is simply calculated by:

$$\begin{bmatrix} \mathrm{Y} \\ \mathrm{Cb} \\ \mathrm{Cr} \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.144 \\ 0.16875 & 0.33126 & 0.5 \\ 0.5 & 0.41869 & 0.08131 \end{bmatrix} \cdot \begin{bmatrix} \mathrm{R} \\ \mathrm{G} \\ \mathrm{B} \end{bmatrix} \quad (1)$$

Each plane (Y, Cb and Cr) is first partitioned to blocks n× n (8×8, 16×16 or 32×32) and the DCT transform is applied to the all blocs of the image. The search of the minimum set of the DCT coefficients that ensure a prefixed user PSNR is done by the bisection method by finding the threshold which all quantised DCT coefficients less this threshold are zeroed. The quantisation is performed by a uniform scalar quantizer given by:

$$\mathrm{XDCT} = (2^{Q} - 1) \cdot \frac{\mathrm{XDCT} - \mathrm{XDCT}_{\min}}{\mathrm{XDCT}_{\max} - \mathrm{XDCT}_{\min}} \quad (2)$$

Where $\mathrm{XDCT}_{\min}$ and $\mathrm{XDCT}_{\max}$ are the minimum and maximum of XDCT values, respectively.

The measure of the efficiency of a compression method uses generally the compression ratio and the quality of the reconstruction metrics. Here, we have used the bits per pixel (bpp) as compression ratio metric and the PSNR for the quality of the reconstruction. The bpp and the PSNR are given by:

$$\mathrm{bpp} = \frac{\text{size of compressed image in bits}}{\text{number of pixels}} \quad (3)$$

$$\mathrm{PSNR} = 10 \cdot \log_{10}\left(\frac{255^2 \cdot 3}{\mathrm{MSE(R)} + \mathrm{MSE(G)} + \mathrm{MSE(B)}}\right) \quad (4)$$

Where $\mathrm{MSE}(\cdot)$ is the mean squared errors throw the R, G or B planes.

$$\mathrm{MSE} = \frac{1}{N \cdot M} \cdot \sum_{i=1}^{N} \sum_{j=1}^{M} \left(I_{i,j} - \hat{I}_{i,j}\right)^2 \quad (5)$$

I and $\hat{I}$ are, respectively, the original and reconstructed R, G or B planes, N and M are the size of the image.

Proposed lossless encoder: A new technique is developed here to encode the thresholded and quantised DCT coefficients. It uses a differential lookup table (dLUT) of the indexes of the retained coefficients in coordination with the adaptive scanning [5] to encode efficiently the block-based DCT coefficients.

The proposed method stores the non-null quantised DCT coefficients arranged in a vector called NN and put theirs indexes generated according to one of the four scanning order used by Douak *et al*. [5] in a temporary LookUp Table (LUT) vector. The differences of these indexes are stored in a vector called dLUT which is used in the decompression stage. The optimal scan is which gives the minimum bits to encode the dLUT vector. An illustrative example is given below to explain the method.

To encode each DCT block, a header containing the control bits is needed. The format of encoded block is depicted in Fig. 2 and composed from:

- L (P bits): The size of NN quantised coefficients, $P = \log_2(n \times n)$ bits.
- AS (2 bits): Refers to the best scan.
- q (P bits): Number of bits necessary to code the dLUT vector.
- dLUT ($q \times L$ bits): The dLUT vector.
- NN ($Q \times L$ bits): Contains the non-null quantised coefficients of width (Q bits).

The dLUT vector is obtained from the LUT vector by:

dLUT(1)=LUT(1)
for i =2 to L do
  $\mathrm{dLUT}(i) = \mathrm{LUT}(i) + \mathrm{LUT}(i-1)$
endfor

The header has a fixed size and is equal to $(P + 2 + P)$ bits.

Example : The quantised (Q=7 bits quantizer) bloc of the thresholded (threshold=3) DCT coefficients BDCTQ of the first 8×8 block intensities depicted from the Y plane of Lena colour image is given by:

$$\mathrm{BDCTQ} = \begin{matrix} 101 & 26 & 0 & 0 & 0 & 0 & 25 & 26 \\ 26 & 0 & 0 & 25 & 0 & 0 & 0 & 0 \\ 25 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$$

The application of the different adaptive scans [5] produces four scans. The better order scan for this block is the horizontal scan (AS=01) which corresponds to the minimum value of q ($q = 4$ for

Zigzag scan (AS=00), $q = 5$ for Vertical scan (AS=10), and $q = 6$ for Hilbert scan (AS=11)).

For the Horizontal scan (AS = 01) we have:

NN=	101	26	25	26	26	25	25
LUT=	1	2	7	8	9	12	17
dLUT=	1	1	**5**	1	1	3	**5**
L=	7						
q=	3						

q is calculated by :

$$q = \lceil \log_2(\max(\text{dLUT})) \rceil \quad (5)$$

where: $\lceil \cdot \rceil$ represents the rounding to the highest integer.
The maximum value of dLUT is in bold.

We have: $Q = 7$ and $P = \log_2(8 \times 8) = 6$ bits.

The block header as follows: 000111 10 000011

The number of bits for this block is:

$S = P + \text{AS} + P + (q \cdot L) + (Q \cdot L) = 84$ bits.

Usually, a lossless entropy encoder is used to compress the resulted blocks. Due to wide difference of the values of the block headers, NN vectors and the dLUT vectors, it is more suitable to encode theme separately. An arithmetic encoder is applied to the concatenated blocks headers, concatenated NN vectors and the concatenated dLUT vectors separately for more lossless compression.

Table 1: Comparison between the proposed method, CDABS [5] and GA-DWT [7] algorithms.

Image	Proposed method		GA-DWT [7]		CDABS [5]	
	PSNR	bpp	PSNR	bpp	PSNR	bpp
Airplane	31.16	0.48	31.16	0.49	31.40	0.72
Peppers	31.19	0.88	31.20	0.83	30.33	0.88
Lena	32.65	0.74	32.76	0.66	32.77	1.00
Girl	35.86	0.38	35.90	0.41	36.96	0.69
Couple	32.62	0.79	32.87	0.89	33.07	1.13
House	23.27	0.79	32.10	0.83	31.32	0.75
Zelda	32.01	0.82	31.98	0.76	32.05	1.09
Average	32.54	0.70	32.57	0.70	32.56	0.89

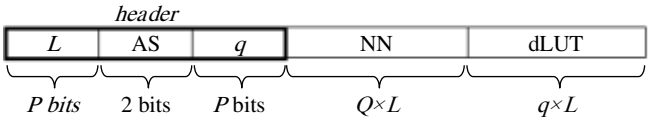


Fig. 2 *Format of proposed encoder*

At the decompression stage, the headers blocks are first obtained followed by NN and the dLUT vectors. After decoding the header block. Its first P bits are the size of the NN coefficients of the block (L). Thus, the NN coefficients and dLUT vector are obtained.

The LUT vector is generated form the dLUT vector by:

LUT(1)= dLUT(1)

for i =2 to L do

$\text{LUT}(i) = \text{LUT}(i-1) + \text{dLUT}(i)$

endfor

Results and discussion: With aim to compare our method with other methods, we have used the same test colours images used in [5, 7]. The DCT block size 16×16 gives the best results comparing with other block sizes 8×8 and 32×32. Table 1 presents comparative results with CDABS [5] and GA-DWT [7]. It is clearly that the proposed method outperform those reported in [5, 7]. The method in [7] uses a genetic search for plan transform to concentrate the energy in the first plan which slows the compression process strongly. Our results are very close to those obtained by GA-DWT [7] but without any evolutionary algorithms such genetic algorithm.

The curves of the Fig. 3 give the performances in PSNR vs bpp for the different quantizer width for all test images. The original and the reconstructed image of *Lina* are given in Fig. 4 showing the visual and quantitative performance of the proposed method.

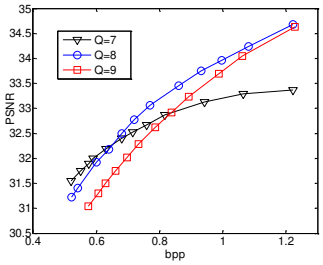


Fig. 3 *Performances in PSNR vs bpp for different quantizer width for all test images*

a *b*

Fig. 4 *Lina image*
a Original image
b Reconstructed image PSNR = 32.15 and bpp = 0.64

Conclusion: In this work, a new efficient algorithm for colour images compression has been proposed. Our proposed algorithm is based on DCT transform and a simple and efficient lossless encoder. The non-null thresholded DCT coefficients are stored with a vector containing the difference of theirs indexes according to an adaptive scanning order. The experimental results obtained on different colour images showed clearly that the proposed method consistently outperforms the methods CDABS [5] and GA-DWT [7].

Submitted: *9 June 2016*
doi: 10.1049/el.2016.2115
One or more of the Figures in this Letter are available in colour online.

A. Messaoudi and K .Srairi (*MSE Laboratory, University of Biskra, BP145, 07000, Biskra, Algeria.*)
A. Messaoudi: *Department of Electronics Faculty of Science and Engineering, University of Hadj Lakhdar Batna 05000, Algeria*.
E-mail: messaoudi.ahamid@gmail.com

References

1 Khalid S.: 'Introduction to data compression' (Elsevier, San Francisco USA, 2006, fourth edition)
2 Delp, E.J., Mitchell, O. R.: 'Image compression using block truncation coding'. *IEEE Trans Commun*., 1979, **27**, pp. 1335-1342.
3 Feng, Y.S., Nasrabadi, N.M.: 'Dynamic address-vector quantisation of RGB colour images'. *IEE Proceedings I, Communications, Speech and Vision.*, 1991, **138**,pp. 225-231.
4 Abadpour, A., Kasaei, S.: 'Color PCA eigen images and their application to compression and watermarking'. *Image and Vision Computing.*, 2008, **26**, pp. 878-890.
5 Douak, F., Benzid, R., Benoudjit, N.: 'Color image compression algorithm based on the DCT transform combined to an adaptive block scanning'. *AEU-International Journal of Electronics and Communications.*, 2011, **65**, pp. 16-26.
6 Pearlman, W., Islam, A., Nagaraj,N., Said, A.: 'Efficient, low-complexity image coding with a set partitioning embedded block coder'. *IEEE Transactions on Circuits and Systems for Video Technology.*, 2004, **14**, pp. 1219-1235.
7 Boucetta, A., Melkemi, K.E.: 'DWT based-approach for color image compression using genetic algorithm'. *Image and Signal Processing Springer Berlin Heidelberg.*, 2012, pp. 476-484.

إهداء

- إلى أمي العزيزة، أهدي هذا العمل ليكون ردًا متواضعاً لتضحياتها طوال عمرها.

- إلى روح أبي راجياً من المولى أن يتغمده بالرحمة والعفو.

- إلى زوجتي الغالية وأولادي: سيرين، إياد ودارين.

- إلى أخي سمير وزوجته وأولاده: رامي، آية وإيمان.

- إلى أخواتي: صليحة، نزيهة ولوبزة وعائلاتهن.

- إلى أصدقائي: حكيم، كمال، خليل، توفيق وفاتح.

- إلى كل استاذ وأستاذة درسني وكان له فضل عليّ.

عبد الحميد مسعودي

Table des matières

Table des figures

Liste des tableaux

Remerciements

C'est avec un grand plaisir que je réserve ces lignes en signe de reconnaissance à tous ceux qui ont de près ou de loin contribué à rendre ce travail possible. Même si des fois nos expressions nous trahissent et n'expriment pas réellement nos sentiments les plus sincères.

Ma profonde gratitude va en premier, à Allah le tout puissant, le miséricordieux de m'avoir donné la force et la patience d'arriver à mettre ce travail à terme.

Mes vifs remerciements vont à mon directeur de thèse Monsieur **Grairi Kamel** professeur à l'université de **Biskra**, d'avoir accepté de m'encadrer au cours de ce lent parcours. Pour la confiance et la liberté qu'il m'a accordée dans la réalisation de ce travail.

J'adresse tous d'abord mes remerciements à Monsieur **Laidi Lamir**, professeur à l'université de **Batna-2**, pour l'honneur qu'il me fait en acceptant d'assurer la présidence de mon jury de thèse.

Je suis ravie de pouvoir compter parmi les membres de mon jury, **Abdou Latifa**, maître de conférences à l'université de **Biskra**.

Mes remerciements, vont particulièrement à Monsieur **Ouafi Abdelkrim**, maître de conférences à l'universsité de **Biskra** qui m'a honoré en acceptant d'être parmi mon jury.

Mes remerciements vont de même à Monsieur **Aissi Salim**, maître de conférences à l'université de **Batna-2**, qui n'a pas hésité d'être parmi mon jury.

Quoi que je dise, je ne pourrais certainement pas remercier assez Monsieur **Benzid Redha**, professeur à l'université de **Batna-2** de m'avoir mise sur la voie de la compression, pour ses conseils précieux, d'avoir toujours montré pour moi, une amitié digne de l'être ainsi pour son soutien moral. Qu'il trouve ici l'expression de ma profonde gratitude.

Enfin merci à tous ceux qui m'ont aidé afin que je puisse mettre fin à de longues années de travail. Qu'ils trouvent ici l'expression de ma reconnaissance la plus sincère.

Le : 18 Mai 2017,

Messaoudi Abdelhamid.

Résumé

Les domaines d'applications mettant en jeu l'utilisation des images numériques ne cessent de s'étendre sous l'impulsion des progrès réalisés dans les technologies de l'information et des télécommunications. Ces applications génèrent des quantités de données considérables pouvant rapidement saturer les systèmes de transmission et de stockage. La nécessité de compresser les images apparaît donc incontournable.

C'est dans ce cadre de la compression d'images fixes couleurs que se situe ce travail de thèse. Par une exploitation efficace des techniques de compression existantes, nous avons proposé un nouveau codeur simple et efficace pour la compression des images en couleurs fixes. La méthode proposée en s'appuyant sur la *transformée en cosinus discrète* (DCT), utilise un scan *adaptatif* en association avec une table de différence d'indices pour coder efficacement les coefficients DCT *non nuls*.

Les résultats de comparaison de la méthode proposée avec divers travaux existants sur un ensemble d'images de test montrent clairement son efficacité et sa supériorité.

Mots clés : Compression d'images couleurs, DCT, Quantification scalaire, Scan adaptatif, LookUp Table, PSNR.

Abstract

The fields of applications involving the use of digital images are constantly expanding under the impetus of advances in information and telecommunications technologies. These applications generate huge amounts of data that can quickly saturate the transmission and storage systems. The necessity of the image compression appears therefore unavoidable.

It is in this framework of color fixed image compression that this thesis work is located. By efficiently using existing compression techniques, we have proposed a new, simple and efficient encoder for the compression of fixed color images. The proposed method based on the *discrete cosine transform* (DCT) uses an *adaptive* scan in conjunction with an index difference table to efficiently encode *non-null* DCT coefficients.

The results of comparison of the proposed method with various existing methods on a set of test images clearly show its efficiency and superiority.

Keywords : Image compression, DCT, Scalar quantization, Adaptive scan, LookUp Table, PSNR.

ملخص

إن استعمال الصور الرقمية في الحياة اليومية في ازدياد كبير مدفوعاً بالتطور الحاصل في مختلف المجالات كالمعلوماتية و الاتصالات. هذه المجالات تنتج و تستعمل كميات هائلة من المعلومات و الصور تستطيع بسهولةٍ كبيرةٍ أنْ تُشَبَّعَ أنظمة الاتصال و التخزين. من هذا المنطلق، يكون ضغط الصور أمراً لازماً و لا مفرَ منه. هذه الدراسة تندرج في سِياق ضغط الصور الثابتة الملونة باستعمال تحويل DCT.

بالاستغلال الفعَّال لأساليب الطرق الموجودة لضغط الصور، قمنا في هذا العمل، بإنشاء مُرَّمِز يعتمد على طريقة جديدة، بسيطة و فعالة. هذا المُرَّمِز يستعمل المسح أو القراءة المُتأقْلِمَة لمعاملات DCT و جدول الفرق بين مؤشرات المعاملات غير المعدومة وذلك لضغط المعلومات الخاصة بالصورة.

نتائج مقارنة الطريقة المدروسة مع طرق أخرى، و منها من يستعمل تحويل المويجات القوي، تثبت بساطة، فعالية و تفوق طريقتنا على هذه الطرق.

كلمات مفتاحية : ضغط الصور الثابتة، تحويل DCT ، التكميم العددي، المسح المُتأقْلِمَ ، جدول فرق المؤشرات، PSNR,

Liste des Abréviations

AC	Alternative Component
BMP	Windows Bitmap
bpp	Bit per pixel
CCD	Charge-Coupled Device
CCITT	Consultative Committee for International Telephony and Telegraphy
CDF	Cumulativ Distribution Function
CEI	Commission Electronic International
CGM	Computer Graphics Metafile
CMOS	Complementary metal oxide semi-conductor
CR	Compression Ratio
CWT	Contiuous Wavelet Transform
DC	Discret Component
DCT	Discret Cosine Transform
DCT-1D	DCT unidimentionnelle
DCT-2D	DCT Bidimentionnelle
DFT	Discret Fourier Transform
dLUT	Look Up Table de différence
DPCM	Differential Pulse Code Modulation
DPI	Dots Per Inch
DST	Discret Sine Transform
DWT	Discrete Wavelet Transform
DXF	Data eXchange Format
EBU	European Broadcasting Union
EPS	Postscript / Encapsulated Postscript
EZW	Embedded Zerotree Wavelet
FCC	Federal Communication Commission, USA
FFT	Fast Fourier Transform
GIF	Graphic Interchange Format
HSB	Hue, Saturation, Brightness
HSL	Hue, Saturation, Luminance
HSV	Hue, Saturation ,Value
IDCT	Inverse Discret Cosine Transform
IRCC	International Radio Consultative Committee
ISH	Intensity, Saturation, Hue
ISO	International Organization for Standardization
ITS	Intensité, Teinte, Saturation

JPEG	Joint Photography Experts Group
JPEG-LS	Joint Photography Experts Group *Lossless compression*
KLT	karhumen- Loeve Transform
LBG	Linde, Buzo, Gray
LCH	Luminance, Chroma, Hue
LUT	Look Up Table
LZW	Lempel, Ziv et Welch
MED	Median Edge Detector
MICD	Modulation par Impulsion et Codage Différentielle
MPEG	Moving Picture Experts Group
MSE	Mean Square Error
NTSC	National Television System Comittee
PAL	Phase Alternating Line
PAO	Publication Assistée par Ordinateur
PCX	PiCture eXchange
PNG	Portable Network Graphics
PPP	points par pouce
PSNR	Pic Signal to Noise Ratio
QS	Quantification Scalaire
QV	Quantification Vectorielle
RGB	Red Green Blue
RLE	Run Length Encoding
SNR	Signal to Noise Ratio
SPIHT	Set Partitioning In Hierarchical Trees
STFT	Short Time Fourier Transform
TIFF	Tag Image File Format
VHS	Visuel Human System
VLC	Variable Length Coding
YCbCr	Luminance, Chrominance bleue, Chrominance rouge
WHT	Walsh Hadamard Transform
WMF	Windows Meta File
WT	Wavelets Transform

Introduction Générale

La numérisation des données ; *signal*, *image* et *vidéo* ; est devenue une procédure technique de plus en plus employée dans notre vie. Diverses applications telles que la télécopie, la vidéoconférence, l'imagerie médicale, l'imagerie satellitaire, la télévision à haute définition, la télésurveillance et les services d'informations sur l'Internet et le mobile engendrent une circulation de l'information de plus en plus dense. Cette numérisation a pour objectif de faciliter, entre autres, leur analyse, leur traitement et leur transmission.

Face à l'accroissement massif de la quantité de données échangées, les supports de stockage et les réseaux de transmission arrivent à leur saturation. L'augmentation croissante et continue des capacités de stockage et l'amélioration des débits sur les réseaux informatiques apportent une réponse partielle à ce problème mais demeure la plupart du temps insuffisante. Dans ce contexte, la compression apparaît comme l'outil essentiel, impératif et incontournable à la continuité du progrès. A l'aide des techniques de compression, le stockage et la transmission des données deviennent plus efficaces et plus rapides.

Une image vaut mille mots, un proverbe chinois laisse entrevoir l'image comme un objet de prédilection pour appréhender le monde et ses réalités. Les techniques de compression d'images se doivent d'être constamment améliorées imposé par les challenges dans les divers domaines.

Diverses méthodes de compression d'images ont été élaborées et normalisées, cependant aucune ne présente un caractère universel. Le choix de la méthode de compression dépend fortement du contexte applicatif. En effet, la nature d'images traitées (images *fixes* ou séquence *vidéos*), leur type (*binaires*, *niveaux de gris* ou *couleurs*), leur dynamique, la qualité de reconstruction demandée (*sans perte* ou *avec perte*) et le débit ciblé jouent un rôle décisif pour le choix d'une ou telle technique de compression. Par exemple, pour les images destinées à être échangées via *Internet*, le débit doit être pris en premier lieu, tout en conservant une qualité acceptable de l'image reconstruite. Par contre, dans le domaine de l'imagerie médicale, la qualité est avantagée au détriment

du débit de transmission. Dans le cas des images destinées au grand public, une forte compression avec une perte d'informations en tenant compte de la perception de l'œil est fortement recommandée. Ces divers facteurs ne peuvent être pris en compte simultanément par un seul algorithme de compression.

Le point de départ des techniques de compression est que la simple numérisation des images ne permet pas une représentation compacte de l'information qu'elles contiennent. Cette représentation numérique est fortement redondante, ce qui invite la recherche d'une méthode pour la représenter sous une forme plus dense que la simple description numérique tout en conservant le même message visuel quel contient.

Les techniques de compression se classent en deux catégories principales, la compression *sans pertes* et la compression *avec pertes*. La compression *sans pertes* est complètement réversible, aucune perte d'informations n'est introduite par les processus de *compression/décompression*. Elle consiste à réduire les données en exploitant la redondance informationnelle dans l'image. La compression *sans pertes* trouve un intérêt particulier en imagerie, par exemple, dans le domaine liés aux applications militaires ou médicales. En effet, ce type de compression garantit que l'intégrité des données est conservée mais il souffre d'un faible taux de compression. Effectivement, la simple élimination de la redondance ne fournit pas des taux de compression acceptables pour plusieurs applications. Ainsi, pour obtenir une compression plus efficace, il est indispensable de négliger quelques informations jugées inutiles.

Dans ce contexte, la compression *avec pertes* maîtrisées peut être la réponse la plus appropriée, à condition bien entendu que la qualité des images reconstruites n'affecte pas leur usage régulier. Dans ce cas, les améliorations apportées à la compression ne sont pas dues à l'élimination des données redondantes, mais plutôt à l'abandon des informations estimées non pertinentes telles que les détails non perceptibles à l'œil nu. La pertinence d'une méthode de compression *avec perte* dépend en plus du taux de compression, de la qualité de l'image restituée. Le plus souvent en décrit cette pertinence par des critères mathématiques tel que le *Compression Ratio* (CR), le *Peak Signal to Noise Ratio* ($PSNR$) et le *bit par pixel* (bpp). Pour une compression efficace, il est impératif de prendre en compte que les images sont pour la plupart des cas fortement corrélées. Ainsi, un schéma de compression classique se décline généralement en deux étapes : une étape de décorrélation, réalisée soit par une méthode *prédictive* soit par une *transformée*, suivie d'une étape de codage *avec* ou *sans pertes*.

La compression *avec pertes* est généralement réalisée dans le domaine fréquentiel par des méthodes basées sur les transformées. Le rôle essentiel des transformées est de

décorréler la source de données, tandis que celui du codeur est d'éliminer l'information redondante. Il est très important de souligner qu'une transformée, en elle-même, ne réalise pas de compression ; seul le codeur appliqué en sortie de la transformée comprime l'image. La transformée permet seulement de générer un signal favorisant une compression plus efficace. Généralement, une transformation ordonnance l'information en séparant l'information *basse* fréquence de l'information *haute* fréquence, les corrélations *proches* des corrélations *lointaines* et les variations *rapides* des variations lentes. Les transformées effectuant la décorrélation en fréquence sont les plus utilisées en compression d'images et de séquences vidéo.

En littérature, on trouve diverses transformées telles que transformée de *Fourier* (FT), la transformée de *Hadamard* (HT), la transformée de *Karhunen-Loève* (KLT), la transformée en *cosinus discrète* (DCT= et la transformée en *ondelettes* (WT).

Dans le cadre de cette thèse, nous nous intéresserons aux méthodes de compression *avec pertes* des images naturelles (*non synthétiques*) fixes couleurs à usage général (*personnes*, *paysage*, etc.) pour lesquelles la problématique principale consiste à trouver le meilleur compromis entre le taux de compression et la dégradation introduite sur l'image reconstruite.

Plan du manuscrit

Ce manuscrit s'articule en quatre chapitres :

Le premier chapitre de cette thèse présente un bref descriptif sur les notions de la lumière et de couleur. Ce chapitre décrit en plus quelques modèles de représentation de la couleur ainsi que les principaux concepts concernant les images.

Le deuxième chapitre présente un état de l'art sur les techniques de compression *avec* et *sans pertes*. Il développe les outils et les notions nécessaires afin d'appréhender l'approche élaborée.

Le troisième chapitre se focalise particulièrement sur la *transformation en cosinus discrète* (DCT). En premier lieu, on donne les principales propriétés de cette transformation. A la fin de chapitre, on décrit le standard de compression JPEG qui se base sur cette transformation.

Le quatrième chapitre est consacré à la méthode de compression élaborée. Une partie de ce chapitre est réservée à la présentation détaillée de notre approche. Tandis que dans le reste, nous présentons les résultats obtenus sur les différentes images de test. Une analyse des résultats obtenus ainsi qu'une étude comparative de ces résultats sont proposées à la fin de ce chapitre.

Cette thèse s'achève sur une conclusion générale et sur les perspectives pouvant en découler.

Chapitre I

Généralités sur les images

I.1 Introduction

Dans de nombreux domaines, l'image numérisée remplace les images analogiques classiques telles que les photographies, radiographies, etc. Une image numérique est un signal bidimensionnelle (*largeur* plus *hauteur*) discret et fini (préalablement échantillonné et quantifié) sous forme d'une matrice où chaque élément représente un point de l'image (*pixel*) qui appartient à un espace de couleur défini.

Nous allons présenter au niveau de ce chapitre les notions de base liées à la lumière et la couleur. Puis, nous exposerons quelques modèles de représentation de la couleur. Enfin, nous aborderons les principaux concepts concernant les images.

I.2 Notion de la lumière

Isaac Newton a montré en 1666 que la lumière blanche pouvait être séparée par un prisme en différentes couleurs qui composent le spectre visible qui s'étale du *rouge* au *violet*. La lumière est un phénomène physique constituée d'un ensemble d'ondes électromagnétiques dont les longueurs d'onde vont pour le spectre visible de 380 *nm* (*violet*) à 780 *nm* (*rouge*) [ANW01]. La figure I.1 illustre le spectre de la lumière *blanche*.

I.3 Notion de la couleur

La couleur est une sensation que nous percevons de façon différente selon la longueur d'onde de la lumière que nous voyons. La perception humaine de la couleur est liée à la physiologie de l'œil. Ainsi, l'être humain est capable de voir des lumières dont la longueur d'onde est comprise entre 380 *nm* et 780 *nm*. Généralement, les sensations de *bleu*, *vert* et *rouge* se composent pour donner toute la gamme des sensations colorées que nous distinguons [BER16][ANW01].

Tomas Young (1793-1829) a démontré que trois couleurs primaires suffisent pour égaliser ou reconstituer pratiquement toute couleur. Plus tard, *James Clerk Maxwell* (1831-1879) a démontré que les couleurs *rouge*, *verte* et *bleue*, sont le jeu des primaires qui est le plus approprié pour l'égalisation couleur. *Helmholtz* a repris et approfondi les travaux de *Young* sur la base des expériences de *Maxwell*. Il a établi les bases de la théorie tri-chromatique qui stipule que tout stimulus couleur peut être reproduit par un mélange des trois stimuli de base : le *rouge*, le *vert* et le *bleu*. Cette théorie porte le nom de théorie de *Young-Helmholtz* ou théorie de synthèse *additive* des couleurs [HEU09].

I.4 Synthèse additive et soustractive

La synthèse de la couleur peut se faire de deux manières, *additive* et *soustractive*. Pour le premier type, une couleur est obtenue par l'addition de trois composantes primaires *Rouge*, *Verte* et *Bleue* (RVB). Ainsi, pour obtenir par exemple la couleur *blanche*, on additionne les composantes : *rouge*, *verte* et *bleue*. L'absence de ces trois composantes donne la couleur *noire*. La synthèse *additive* est généralement utilisée pour les moniteurs ou les télévisions en couleurs. Les couleurs *cyan*, *magenta* et *jaune* sont dites secondaires car :

- Le *vert* combiné au *bleu* donne du *cyan*,
- Le *bleu* combiné au *rouge* donne du *magenta*,
- Le *vert* combiné au *rouge* donne du *jaune*.

À l'opposé de celle-ci, la synthèse *soustractive* n'est pas le fait d'une projection, mais de l'absorption de la lumière par une encre physique. Elle permet de restituer une couleur par soustraction d'une ou plusieurs composantes de la lumière *blanche*. La soustraction est réalisée par des filtres correspondant aux couleurs complémentaires : le *jaune*, le *magenta* et le *cyan*. La combinaison (ajout) de ces trois couleurs produit donc le *noir* et leur absence donne du *blanc*. Les composantes de la lumière sont ajoutées

après réflexion sur un objet, ou plus exactement sont absorbées par la matière. La figure I.2 représente les deux synthèses de couleurs.

Ce procédé est utilisé en photographie et pour l'impression des couleurs [PIL15a] [BEN07]. Pour cette synthèse, les couleurs secondaires sont le *bleu*, le *rouge* et le *vert* puisque :

- Le *magenta* combiné avec le *cyan* donne du *bleu*,
- Le *magenta* combiné avec le *jaune* donne du *rouge*,
- Le *cyan* combiné avec le *jaune* donne du *vert*.

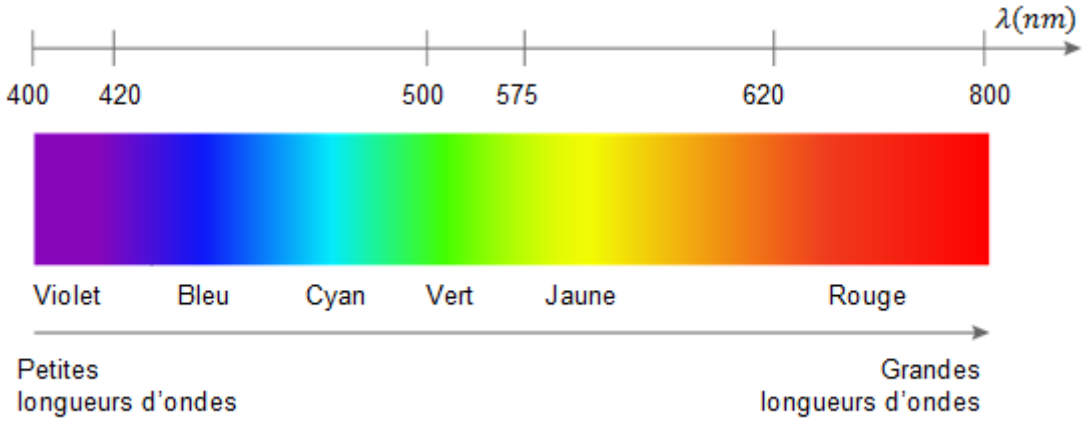
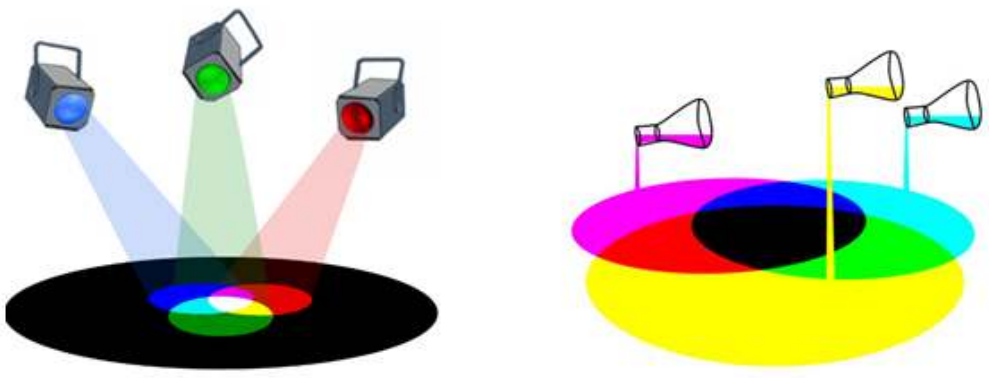


FIGURE I.1 – Spectre de la lumière *blanche*.



(a) (b)

FIGURE I.2 – Synthèse de couleurs : (a) synthèse *additive* et (b) synthèse *soustractive*.

I.5 Fonctionnement de l'œil humain

L'œil humain est un organe complexe et fragile destiné à détecter et analyser la lumière, à la manière d'un appareil photographique à performances étonnantes. Les yeux sont des globes d'environ 24 *mm* de diamètre, logés dans des orbites. Ils sont pourvus de muscles qui leur permettent d'orienter le regard. La figure I.3 illustre les différentes parties qui composent l'œil.

La *cornée*, assimilable au verre qui protège l'*objectif*, est une membrane externe transparente où arrive la lumière, qui atteint un milieu liquide appelé l'*humeur aqueuse*, située entre la *cornée* et l'*iris* [SAN14].

La *pupille* est l'orifice d'entrée de la lumière à l'intérieur de l'œil. A la manière d'un diaphragme, elle se dilate et se contracte en fonction de la luminosité, grâce à un muscle circulaire, l'*iris*. Derrière l'iris, se trouve le *cristallin*, lentille transparente biconvexe, capable de se modifier au fur et à mesure que l'objet se rapproche [SAN14].

Au fond de l'œil, les rayons lumineux atteignent la *rétine*, où les images se forment. La *rétine* est une structure neuronale photosensible située au fond de l'œil. Elle est constituée de cellules de deux types différents : les *bâtonnets* et les *cônes*. Les *bâtonnets* permettent de percevoir la luminosité et le mouvement permettant ainsi la vision nocturne, tandis que les *cônes* permettent de différencier les couleurs assurant la vision diurne. Enfin, le *nerf optique* achemine les informations visuelles au cerveau et c'est comme ça que l'image réelle prend forme.

La perception visuelle est la combinaison d'un capteur, l'*œil* et d'un système d'analyse, le *cerveau*. La perception des couleurs se définit comme une sensation, elle est donc associée à la subjectivité humaine [ANW01].

L'œil discerne une immense variété de couleurs différentes pourtant, il ne possède que trois types de cônes ayant une sensibilité plus grande à certaines radiations :

- Cônes sensibles au *rouge* (580 *nm*),
- Cônes sensibles au *vert* (540 *nm*),
- Cônes sensibles au *bleu* (450 *nm*).

D'autre part, il est à noter que la sensibilité de l'œil humain aux intensités lumineuses relatives aux trois couleurs primaires est inégale.

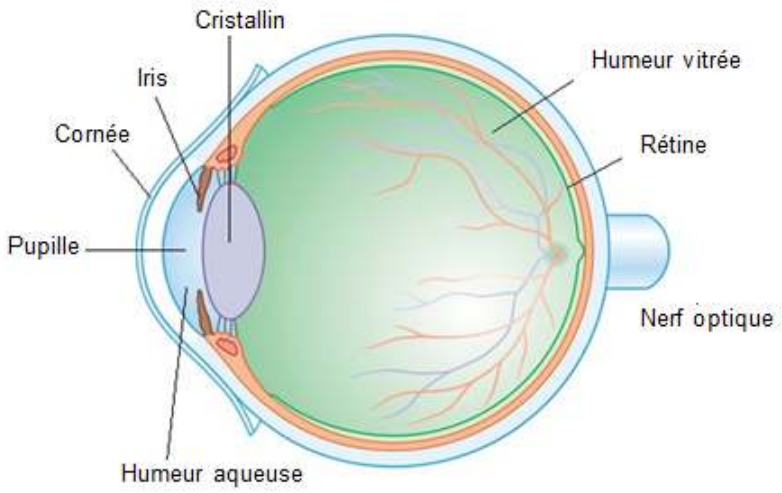


FIGURE I.3 – Anatomie de l'œil humain.

I.6 Espaces de couleur

Afin de choisir et d'échanger des informations concernant les couleurs, il est indispensable d'avoir un moyen ou un modèle qui permet de sélectionner une couleur bien spécifique. Dans la vie quotidienne, on choisit souvent la couleur d'une voiture ou d'un vêtement avant qu'il soit même fabriqué. Dans ce cas, une palette de couleurs nous est présentée, dans laquelle, nous choisissons la couleur convenant le mieux notre choix. Des expériences psycho-visuelles d'égalisation [KUN93] ont montré qu'en combinant trois stimuli de longueurs d'ondes particulières, il est possible de synthétiser presque toutes les couleurs existantes [MAR10].

On appelle ainsi *espace colorimétrique* ou *espace de couleur* la représentation mathématique d'un ensemble de couleurs. Cette représentation associe des nombres aux couleurs visibles. Chaque couleur est caractérisée par un point dans un espace à trois dimensions sous la forme d'un triplet [BOU10][BEN07].

La plupart des modèles de couleurs sont orientés aspect Matériel (*Hardware*) comme pour les moniteurs couleurs et les imprimantes ou orientés aspect Application (*Software*) comme pour la création de couleur de graphisme pour animation. En effet, il existe plusieurs espaces de couleur qui sont utilisés dans différents domaines d'applications. Par exemple, l'espace CIE XYZ est utilisé en photométrie, RGB pour les moniteurs, CIE LAB dans les textiles, CIE LUV en visualisation scientifique, CMY pour l'impression, YIQ pour la télévision, HSV, HSI et HLS pour la sélection de couleurs, *Munsell* en psychologie et *Ostwald* en peinture [GON77][BOU10].

Le passage d'un système de représentation à un autre se fait par divers types de transformations soit *linéaires* ou *non linéaires* et *réversibles* ou *irréversibles*. D'une façon générale, la plus parts des systèmes de représentation de la couleur appartiennent soit aux systèmes de couleur des primaires soit aux systèmes *Luminance-Chrominance*.

I.6.1 Les systèmes des primaires

Les systèmes dits des *primaires* sont définis selon le principe de la tri-variance visuelle qui stipule que chaque couleur peut être représentée par le mélange additif de trois couleurs de base appelées couleurs primaires. Comme exemple de ce type de systèmes, on trouve les systèmes RGB, XYZ, LMS et CMY.

I.6.1.1 L'espace RGB

L'espace RGB (*Red Green Blue*, pour *Rouge Vert Bleu*, en français RVB), mis au point en 1931 par la *Commission Internationale de l'Eclairage* (CIE) consiste à représenter l'espace de couleur à partir de trois rayonnements monochromatiques de couleurs (*rouge*, *vert* et *bleu*).

Le modèle RGB correspond à la façon dont les couleurs sont codées informatiquement, ou plus exactement à la manière dont les tubes cathodiques des écrans d'ordinateurs représentent les couleurs. Chaque couleur est représentée par un groupe de trois valeurs. C'est-à-dire un octet pour chaque composante de couleur, ce qui correspond à 256 intensités de *rouge* (2^8), 256 intensités de *vert* et 256 intensités de *bleu*, soient 16777216 possibilités théoriques de couleurs différentes. Cela est supérieur à ce que peut en discerner l'œil humain (environ 2 millions) [BEN07].

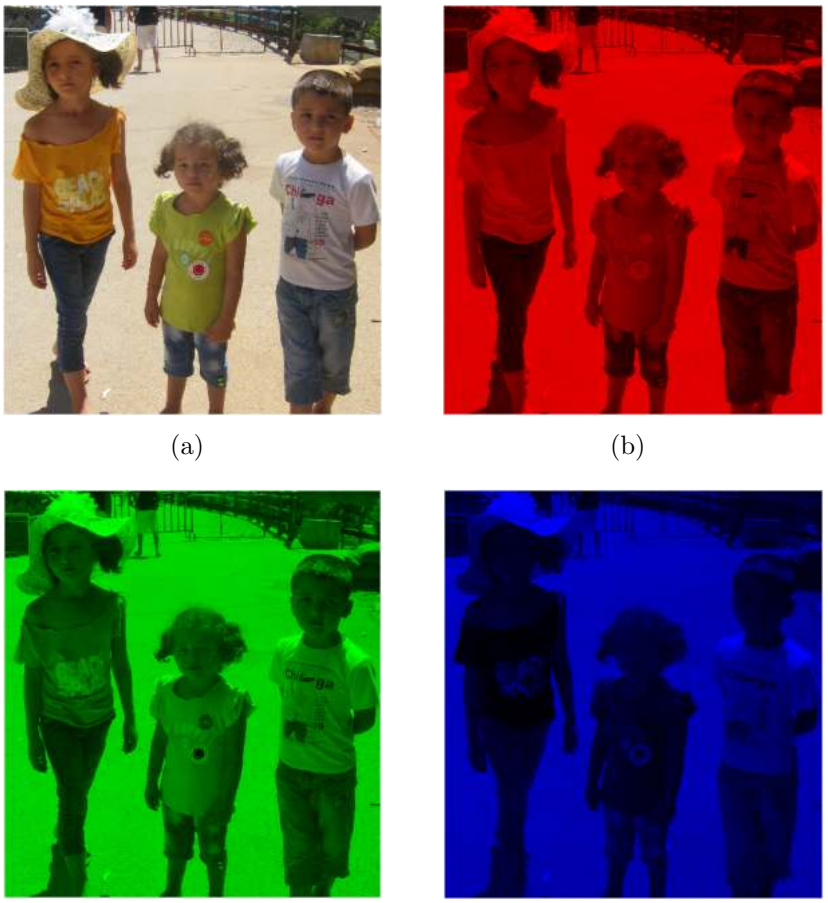
L'espace RGB est généralement représenté par un cube dont chacun des axes correspond à une couleur primaire. La figure I.4.a donne la représentation graphique de ce modèle. Si les trois composantes sont nulles, la couleur résultante est *noire* et la couleur *blanche* est issue de la valeur maximale de chaque composante. Lorsque toutes les composantes sont égales, on obtient une nuance de *gris* neutre. Il est à noter que cet espace est à synthèse *additive* (figure I.4.b) cela veut dire que ses trois couleurs primaires (*rouge*, *verte*, et *bleue*) se combinent d'une manière *additive* pour produire toutes les couleurs désirées. La figure I.5 représente une image *RGB* avec ses composantes principales.

I.6.1.2 L'espace CMY

L'espace CMY (*Cyan Magenta Yellow*) est un espace de couleur à synthèse soustractive. Ce modèle consiste à décomposer une couleur en valeurs de *Cyan*, *Magenta* et *Jaune*. L'absence de ces trois composantes donne du *blanc* tandis que leur addition

(a) (b)

FIGURE I.4 – Espace RGB : (a) Représentation graphique et (b) Composition *additive* des couleurs.



(a) (b)

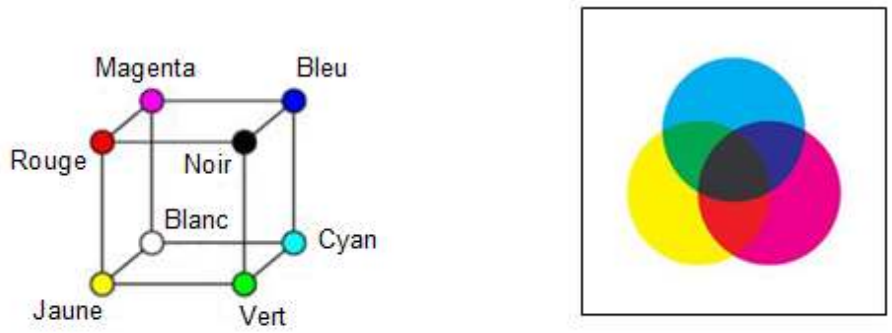
(c) (d)

FIGURE I.5 – Image RGB et ses composantes : (a) Image RGB, (b) composante R et (c) composante G et (d) composante B.

produit du *noir* ce qui n'était pas le cas avec l'espace RGB. L'espace CMY est donc le complémentaire de l'espace RGB. Cet espace de couleur est principalement utilisé en imprimerie. Toutefois, le *noir* obtenu par mélange de couleurs *Cyan*, *Magenta* et le

Jaune n'étant pas parfait et coûtant cher en pratique, une quatrième couleur (cartouche d'encre) est ajoutée avec un *noir* pur.

On parle alors de quadrichromie, ou modèle CMYK (*Cyan*, *Magenta*, *Yellow*, *blacK*). La figure I.6 donne la représentation graphique de l'espace de couleur CMY.



(a) (b)

FIGURE I.6 – Espace CMY : (a) Représentation graphique et (b) Composition *soustractive* des couleurs.

I.6.2 Les systèmes Luminance Chrominance

Dans ce type de systèmes, on dissocie l'information *chromatique* des couleurs de leur *intensité lumineuse*. Pour cela, le stimulus couleur est décrit par une composante de *luminance*, et deux autres composantes *chromatiques*. Parmi les systèmes *Luminance-chrominance*, on trouve les espaces LTS, YCbCr, YIQ et YUV.

I.6.2.1 Les systèmes de type Luminance-Teinte-Saturation

Il paraît difficile selon la perception humaine des couleurs d'évaluer une couleur selon des coordonnées RGB abstraites. Il convient d'utiliser à leur place des notions subjectives telles que la *luminance*, la *teinte* et la *saturation* [BRA04].
En effet, ces notions sont faciles à interpréter et sont liées à nos habitudes quotidiennes pour différencier les couleurs entre elles. Les systèmes de type *Luminance-Teinte-Saturation* (LTS) sont surtout considérés comme des représentations descriptives de la couleur. Plusieurs systèmes quantifiant ces notions sont proposés.

La famille des systèmes LTS est mentionnée en littérature sous différentes nominations, citons à titre d'exemple [HEU09] :

- HSV pour *Hue*, *Saturation* et *Value*,
- ISH pour *Intensity*, *Saturation* et *Hue*,

- ITS pour *Intensité*, *Teinte* et *Saturation*,
- LCH pour *Luminance*, *Chroma* et *Hue*,
- HSL pour *Hue*, *Saturation* et *Luminance*,
- HSB pour *Hue*, *Saturation* et *Brightness*.

L'espace HSL

Naturellement l'éclaircissement d'une couleur se fait par un changement de *Luminance* seule or ce n'est pas le cas pour le modèle RGB. Ce denier impose de changer les trois composantes primaires pour avoir cet éclaircissement. Le modèle HSL (*Hue*, *Saturation*, *Luminance*, ou en français TSL) dit *naturel*, n'a pas cette faiblesse.

Basé essentiellement sur les travaux du peintre *Albert H.Munsell*, le modèle HSL est proche de la perception physiologique de la couleur par l'œil humain. En effet, l'objectif de ces modèles est de classer les couleurs selon un principe basé sur la perception psycho-visuelle. Ils sont surtout considérés comme des représentations descriptives de la couleur [PIL15c]. En effet, la couleur dans ce modèle est composée de :

1. La **teinte** (*Hue*), correspondant à la couleur, *jaune*, *rouge*, etc.
2. La **saturation**, mesure la *pureté* ou la *vivacité* de la couleur décrivant ainsi son caractère *pâle*, *vif*, *délavé*, *éclatant*, etc.
3. La **luminance**, correspondant la quantité de lumière de la couleur (*brillance*), c'est-à-dire son aspect *clair* ou *sombre*.

La représentation graphique du modèle HSL est composée d'un cercle correspondant à la *chromatique* et de deux axes pour la *luminance* et la *saturation*. La figure I.7 illustre la représentation graphique du modèle LTS.

I.6.2.2 L'espace YCbCr

La diffusion des émissions de télévision en couleurs a poussé les chercheurs à développer un système permettant d'assurer une compatibilité entre les téléviseurs couleurs et les téléviseurs *Noir* et *Blanc*. Puisque ces derniers devaient continuer à diffuser les émissions en *Noir* et *Blanc* même si le signal reçu était en couleur. Le système développé a permis de coder l'information *couleur* selon une méthode qui prend en charge les longueurs d'onde d'émission des tubes cathodiques. En effet, ces tubes possèdent des luminophores qui synthétisent la couleur selon des primaires couleurs différentes des primaires RGB [HEU09].

Ce système nommé YCbCr, où Y représente la *luminance*, et les deux composantes *Cb* et *Cr* représentent la *chrominance*. La figure I.8 représente une image au format YCbCr avec ses composantes principales. Il existe plusieurs systèmes du type YCbCr :

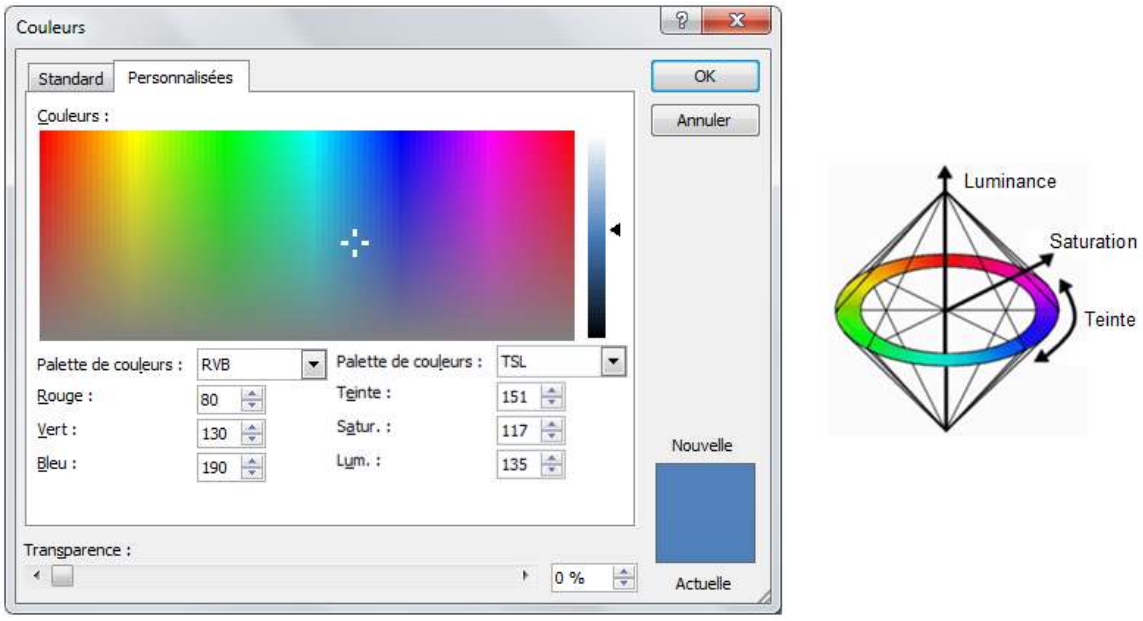
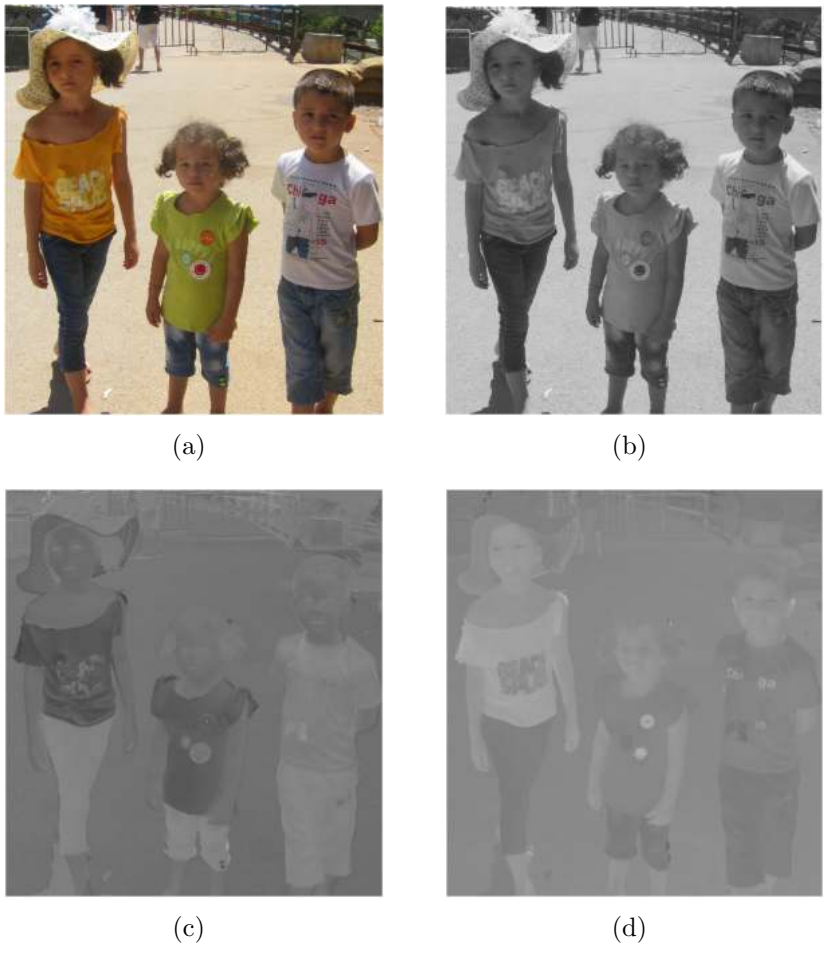


FIGURE I.7 – Représentation graphique du modèle LTS.



(a) (b)

(c) (d)

FIGURE I.8 – Image YCBCr et ses composantes : (a) Image YCbCr, (b) composante Y, (c) composante Cb et (d) composante Cr.

- le système YIQ qui correspond à la norme NTSC,
- le système YUV lié à la norme PAL,
- le système YDbDr de la norme PAL.

Le système YCbCr peut se calculer à partir des composantes EBU RGB (*Europe*) par [ANW01] :

$$\begin{cases} Y = & 0.222 \cdot R & +0.707 \cdot G & +0.071 \cdot B & \\ C_b = & -0.119 \cdot R & -0.381 \cdot G & +0.500 \cdot B & = 0.538 \cdot (B - Y) \\ C_r = & 0.500 \cdot R & -0.454 \cdot G & -0.046 \cdot B & = 0.643 \cdot (R - Y) \end{cases} \tag{I.1}$$

A partir des composantes FCC RGB (*USA*) on a :

$$\begin{cases} Y = & 0.299 \cdot R & +0.587 \cdot G & +0.114 \cdot B & \\ C_b = & -0.169 \cdot R & -0.331 \cdot G & +0.500 \cdot B & = 0.564 \cdot (B - Y) \\ C_r = & 0.500 \cdot R & -0.418 \cdot G & -0.081 \cdot B & = 0.713 \cdot (R - Y) \end{cases} \tag{I.2}$$

Le système YCbCr défini par l'*International Radio Consultative Committee* (IRCC), est souvent utilisé dans la compression des images. Il est largement appliqué dans compression d'images et de la vidéo telles que JPEG et MPEG [ANW01].

I.7 Acquisition des images

La photographie argentique est une technique photographique permettant l'obtention d'une image par un processus photochimique comprenant l'exposition d'une pellicule sensible à la lumière puis son développement et, éventuellement, son tirage sur papier [WIK16a]. Pour la photographie numérique, remplaçant la pellicule, le capteur photosensible fonctionne sur le même principe. Ce dernier est constitué de cellules photovoltaïques qui mesurent l'intensité de la lumière. Cette dernière est ensuite transformée en courant électrique [PIL15b].

Le capteur photographique met à profit l'effet photoélectrique, qui permet aux photons incidents d'arracher des électrons à chaque élément actif (*photosite*) d'une matrice de capteurs élémentaires constitués de *photoDiodes* ou *photoMOS* [WIK16b]. Trois types de capteurs existent, le capteur CCD, le capteur CMOS et le capteur *Foveon* X3.

I.7.1 Capteur CCD

Le CCD (*Charge-Coupled Device* ou *Dispositif à Transfert de Charge*) est inventé par *George E.Smith* et *Willard Boyle* dans les Laboratoires *Bell* en 1969. Un CCD

transforme les rayons lumineux (*photons*) issus de la scène (image) à travers un *objectif optique* en paires *électron-trou* par effet photoélectrique dans le substrat semiconducteur. À la fin de la capture de l'image, les charges sont transférées de *photosite* en *photosite* par l'électronique associée au capteur. Ce système permet de vider régulièrement les charges analogiques de tous les pixels du capteur et les transforment en valeurs numérisées constituant ainsi l'image numérique.

Il est a noté que le nombre d'électrons collectés est proportionnel à la quantité de lumière reçue [HEA94][WIK16b] [WIK16c][ZIT13]. Le principe du transfert de charges est illustré à la figure I.9.

I.7.2 Capteur CMOS

Un capteur CMOS fonctionne sur le sur le même principe d'un CCD, mais au lieu de transférer la charge vers un collecteur, il la conserve et la transfère au convertisseur directement. Leur consommation électrique, beaucoup plus faible que celle des capteurs CCD, leur vitesse de lecture et le plus faible coût de production sont les principales raisons de leur grande utilisation [PIL16b][WIK16b]. Un tel capteur est représenté à la figure I.10.

Pour l'acquisition d'une image en couleur, les capteurs photosensibles CCD ou CMOS sont généralement constitués d'une seule matrice photosensible recouverte d'un filtre coloré appelé grille de *Bayer*. Cette grille est vue comme une juxtaposition de filtres *rouge*, *vert* et *bleu* alignés ne mesurant que quelques microns. La mosaïque de la grille de *Bayer* est constituée de millions de pixels forme une image uniforme. A la fin, le capteur photosensible combine ces trois couleurs primaires RGB pour créer par synthèse *additive* une image en couleurs [LUX16]. La figure I.11 illustre le filtre de *Bayer*.

I.7.3 Capteur Foveon X3

Dans ces capteurs dont la conception est radicalement différente des précédents, les photosites ne sont plus juxtaposés mais superposés. En plus, un photosite de capteur CCD/CMOS capture seulement une couleur primaire (*Rouge*, *Verte* ou *Bleue*) par contre, un photosite de capteur X3 recueille une composante RGB complète.

Ce capteur, développé par la société américaine *Foveon*, exploite le fait que les grandes longueurs d'onde de la lumière pénètrent plus profondément dans le silicium. Cela permet la capture des trois couleurs par un seul photosite, au moyen de trois couches de silicium recouvertes de photosites et disposées en sandwich et filtrées chacune par un filtre *bleu*, *vert* ou *rouge*. Chacune des couches de photo-récepteurs est

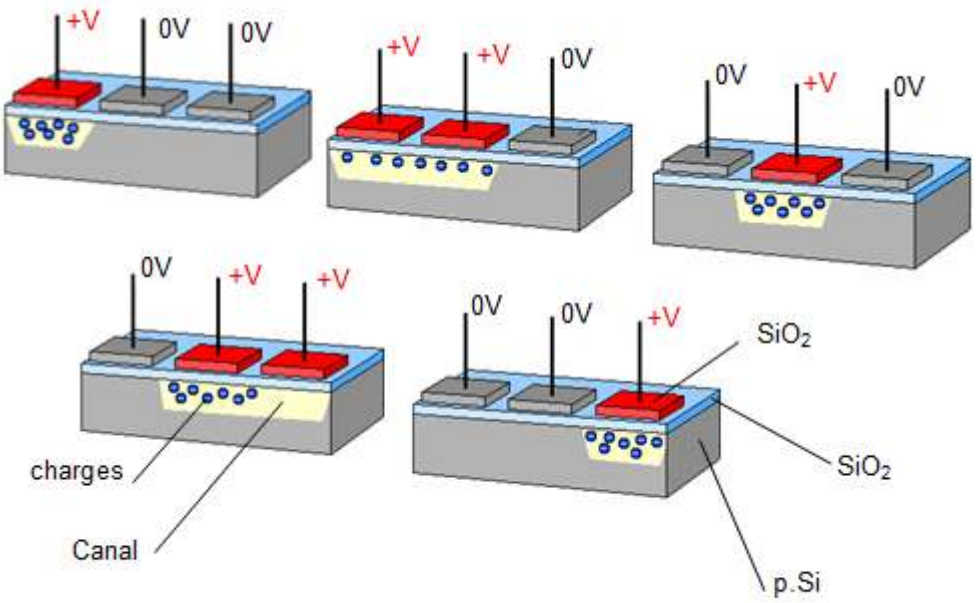


FIGURE I.9 – Principe du transfert de charges (inspirée de [WIK16b]).

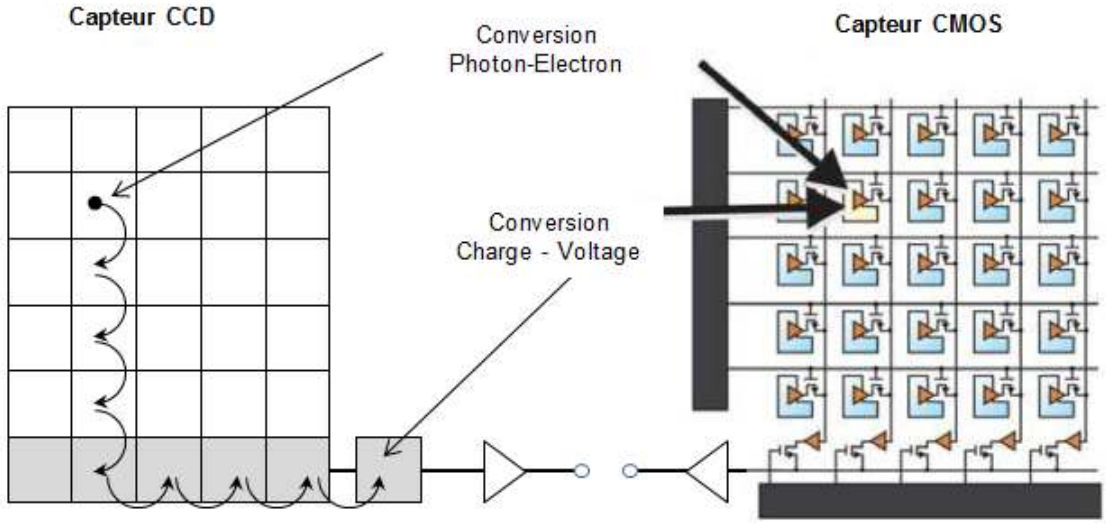


FIGURE I.10 – Capteur CMOS.

précisément espacée relativement aux longueurs d'onde *bleue*, *verte* et *rouge* de la lumière visible. Pour un rayon lumineux incident au capteur, la couche superficielle du silicium arrête le *bleu*, alors que la couche médiane bloque le *vert* et enfin le *rouge* est stoppé par la couche inférieure [WIK16b]. La figure I.12 représente une comparaison entre un capteur CCD/CMOS et un capteur *Foveon X3*.

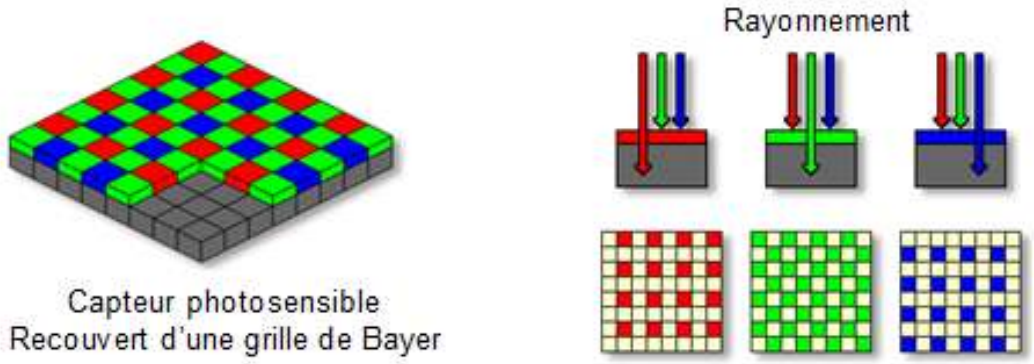
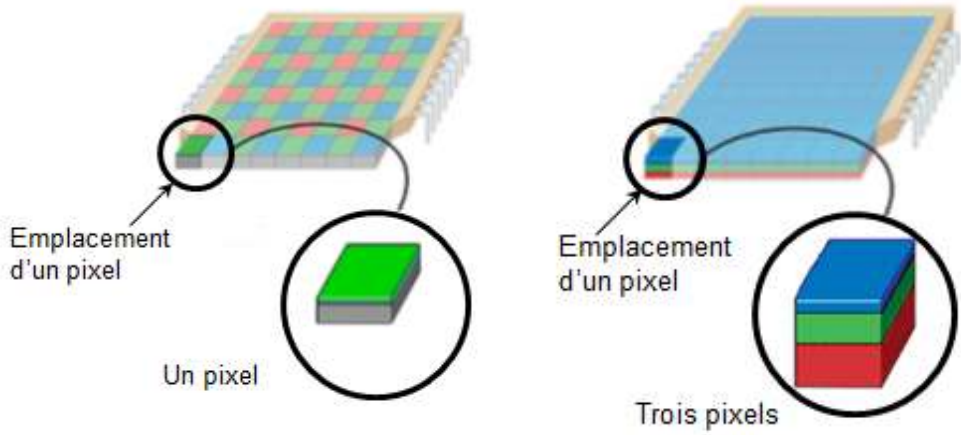


FIGURE I.11 – Filtre de *Bayer* [LUX16].



(a) (b)

FIGURE I.12 – Capteur d'image : (a) Capteur CCD/CMOS et (b) Capteur *Foveon X3*.

I.8 Caractéristiques d'une image numérique

Une image numérique est généralement rangée sous forme de matrice notée I de N lignes et M colonnes. Chaque élément $I(n, m)$ représente un pixel de l'image et sa valeur est codée par des valeurs numériques qui peuvent être scalaires (images en niveaux de gris), ou bien vectorielles (images en couleurs). Lors de l'affichage d'une image l'écran effectue un balayage de gauche à droite et de haut en bas cela signifie que les axes de l'image sont orientés orienté de gauche à droite (axe X) et de haut en bas (axe Y) contrairement aux notations conventionnelles en mathématiques, où l'axe Y est orienté vers le haut.

L'image est un ensemble structuré d'informations caractérisé par des paramètres tels que : *dimension*, *résolution*, *luminance*, *contraste*, etc [GON02][ZIT13].

I.8.1 Pixel

Mathématiquement une image peut être vue comme une matrice à deux dimensions où chaque élément représente un point de l'image. Ce point est appelé *pixel* (*PICture ELement*) et constitue le plus petit élément de l'image. L'ensemble des valeurs d'intensité lumineuse de chaque pixel constitue une image.

I.8.2 Dimension

La *dimension* d'une image correspond au nombre de pixels en hauteur N et en largeur M qui la compose. C'est aussi la taille de l'image, et on la note $M \times N$. La figure I.13 représente les notions de *pixel* et de *dimension* d'une image.

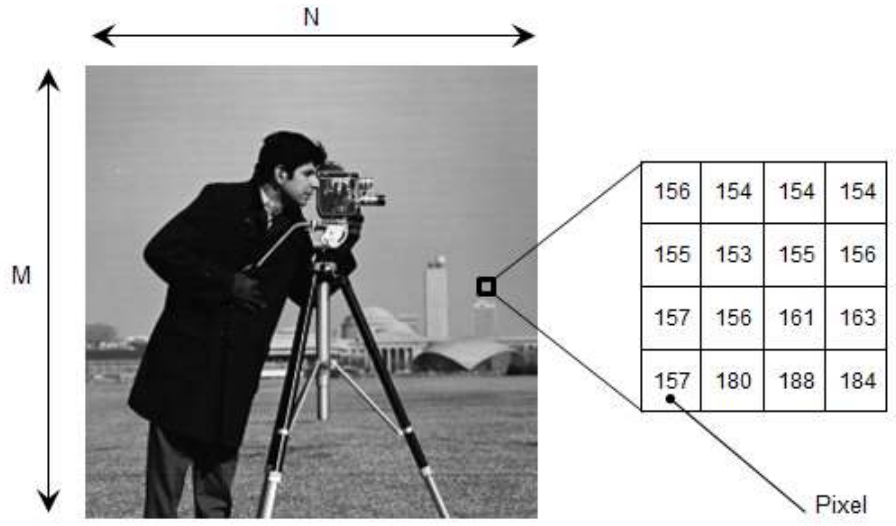


FIGURE I.13 – Caractéristiques d'une image numérique, *pixels* et *dimension* (*cameraman.tif*).

I.8.3 Résolution

La *résolution* d'une image numérique est définie par la *densité* des pixels par unité de surface en pouce. La *résolution* est exprimée en *points par pouce* (PPP) ou *Dots Per Inch* (DPI). Il est noté que plus le nombre de points par pouce est important, plus la résolution est élevée plus l'image a une bonne qualité. Ce paramètre est défini lors de la numérisation et dépend principalement des caractéristiques du matériel utilisé lors de la numérisation.

I.8.4 Histogramme

Un *histogramme* permet de représenter la distribution des intensités des pixels d'une image, c'est-à-dire le nombre de pixels pour chaque intensité lumineuse. Couramment,

un *histogramme* est représenté par un graphe à deux axes, en abscisse le niveau d'intensité allant du plus foncé (à gauche) au plus clair (à droite) et en ordonnée le nombre de pixels correspondant. La figure I.14 représente une image avec son histogramme.

L'*histogramme* de l'image I avec des intensités dans l'intervalle $[0, L]$ est la fonction discrète H définie par [GON77] :

$$\begin{aligned} H : \quad & [0, L] \longmapsto R \\ & H(k) = n_k \end{aligned} \tag{I.3}$$

avec L est le nombre de niveaux de gris dans l'image. C'est à dire que $H(k)$ traduit la fréquence d'apparitions n_k du niveau de gris k dans l'image I. Généralement, on utilise un *histogramme* normalisé en divisant H par la taille de l'image. Communément on utilise l'*histogramme* pour [GON77][BEN07] :

- Améliorer la qualité d'une image (Rehaussement d'images),
- Diminuer l'erreur de quantification,
- Comparer deux images obtenues sous des éclairages différents,
- Donner un grand nombre d'informations sur la distribution des niveaux de gris (couleurs),
- Voir entre quelles bornes est repartie la majorité des niveaux de gris (couleurs),
- Mesurer certaines propriétés sur une image.

I.8.5 Luminance

La *luminance* est une quantité proportionnelle à la puissance d'une source lumineuse. Pour une image, il s'agit du degré de *luminosité* de ses points. Elle correspond au quotient de l'intensité lumineuse d'une surface source par l'aire apparente de cette surface projetée sur la perpendiculaire à la direction d'observation. En d'autres mots, cela correspond à l'éclat d'un objet [KAD99].

I.8.6 Contraste

Le *contraste* quantifie la différence de luminosité entre les parties claires et sombres d'une image. Si L_1 et L_2 sont les degrés de luminosité respectivement de deux zones voisines A_1 et A_2 d'une image, le contraste C est défini par le rapport [KAD99][MAR10] :

$$C = \frac{L_1 - L_2}{L_1 + L_2} \tag{I.4}$$

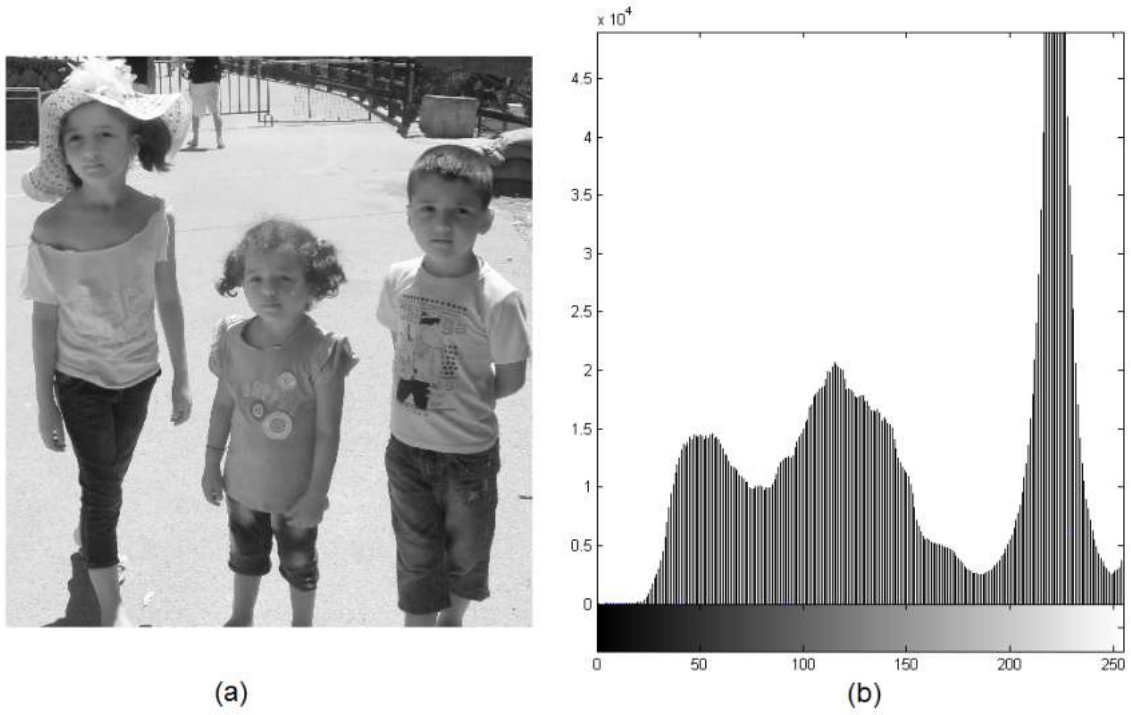


FIGURE I.14 – Exemple d'une image et son *histogramme* associé : (a) image en niveaux de gris et (b) son *histogramme*.

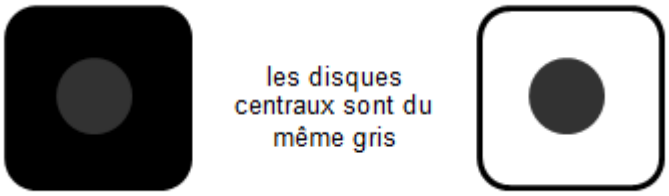
La formule de l'équation (I.4) définit la notion du *contraste global* d'une image. Souvent une définition locale du *contraste* ; qui correspond à la perception des contours des éléments d'une image ; est mieux adaptée à la photographie. Entouré d'une surface sombre, le disque de la figure I.15.a paraît plus clair qu'entourer d'une surface claire (les disques centraux sont du même gris). Le *contraste local* est définit par :

$$C = \frac{L_{Zone} - L_{Fond}}{L_{Fond}} \tag{I.5}$$

I.8.7 Profondeur

La *profondeur* des couleurs d'une image, est mesurée par le *nombre de bits par pixel* (*bpp*). Cette valeur définit le nombre de couleurs ou de niveaux de gris d'une image. Il existe plusieurs valeurs de *bpp* tels que :

- 1 *bit/pixel* = 2 couleurs (image *binaire*),
- 8 *bits/pixel* = 256 couleurs (image en *niveaux de gris*),
- 16 *bits/pixel* = 65 536 couleurs (image en *couleurs*),
- 24 *bits/pixel* = 16 777 216 millions de couleurs (image en *couleurs*),
- 32 *bits/pixel* = 4 294 967 296 de couleurs (image en *couleurs*).



(a)

(b)

FIGURE I.15 – *Contraste* d'une image : (a) même *contraste* et (b) *contraste* à 100% et 50%.

I.8.8 Contours et textures

Les frontières entre les objets d'une image constituent les *contours* que contient celle-ci. Donc, les *contours* sont les frontières entre les pixels dont les valeurs (niveaux de gris) représentent une différence *significative*. Les régions ou zones dont les valeurs de pixels sont proches sont communément appelés *textures*. L'identification des points qui séparent deux *textures* d'une image est appelée extraction de *contours* [KAD99][KUN93].

I.9 Les différents types d'images

Généralement on a trois types d'images : *binaires*, en *niveaux de gris* et en *couleurs*. Ces trois types sont dus au nombre de bit sur lequel est codée la valeur de chaque pixel. Soit $I((L+1)\times(C+1))$ définit dans un espace à p dimensions avec $M+1$ couleurs :

$$I : [0, L] \times [0, C] \longmapsto [0, M]^p \tag{I.6}$$

- Si $p = 1$ et $M = 1$, I est une image *binaire*,
- Si $p = 1$ et le plus souvent $M = 255$, I est une image en niveaux de *gris*,
- Si $p = 3$ et le plus souvent $M = 255$, I est une image en couleurs [BEN07].

I.9.1 Les images binaires

Une image *binaire* est une image où chaque pixel est codé sur un *bit* ($p = 1$ et $M = 1$) et qui ne peut prendre que la valeur 0 ou 1. Les pixels *noirs* ont la valeur 0 et les pixels *blancs* prennent la valeur 1. La figure I.16 représente une image binaire.

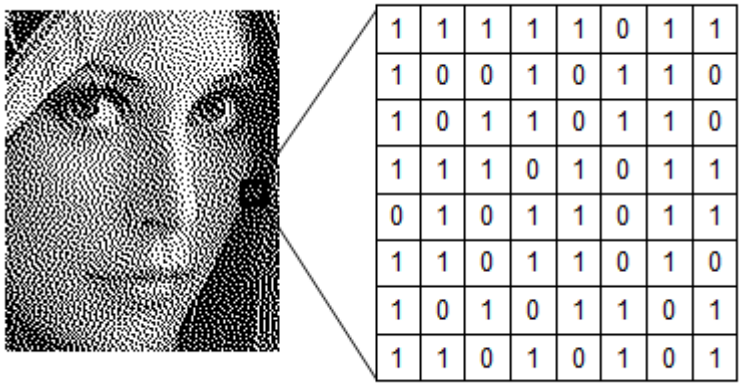


FIGURE I.16 – Exemple d'une image *binaire* (*Lena.tiff* en Halftone).

I.9.2 Les images en niveaux de gris

Une image en *niveaux de gris* ($p = 1$ et $M = 255$) est une image où la valeur du pixel peut prendre des valeurs allant du *noir* au *blanc* en passant par un nombre fini de niveaux intermédiaires. Ces niveaux définissent une gradation de gris entre le *noir* (0) et le *blanc* (255). Chaque pixel n'est donc plus représenté par un *bit*, mais par un *octet*. Un exemple d'une image en *niveaux de gris* est donnée à la figure I.17.

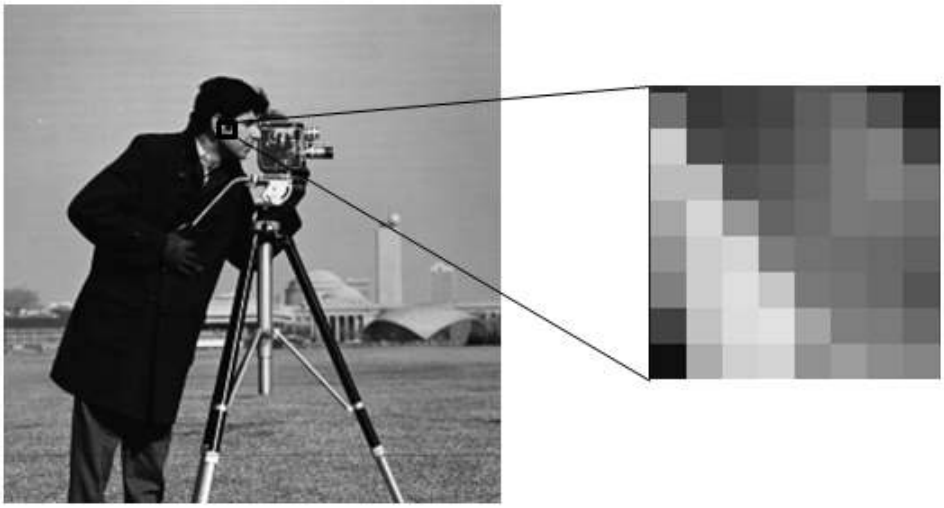


FIGURE I.17 – Exemple d'une image en *niveaux de gris* (*cameraman.tif*).

I.9.3 Les images en couleurs

Une image en couleurs ($p = 3$ et $M = 255$) est composée de trois images en niveaux de *gris* sur trois composantes le plus souvent le *rouge*, le *vert* et le *bleu* (modèle RGB). La couleur finale est obtenue par un mélange de ces composantes primaires. La figure I.18 représente un exemple d'une image en couleurs. Selon le nombre de bits utilisés pour le stockage de l'information couleur, on distingue deux types d'images en couleurs.

I.9.3.1 Les images en couleurs vraies (en true colors)

Dans ce type d'images, l'information couleur est codée sur 24 bits (3 *octets*) soit un octet pour chacune des composantes RGB. Le nombre de couleurs couvert par ce type d'images est plus de 16.7 millions ($256 \times 256 \times 256$) couleurs. Il est possible de rajouter une quatrième composante permettant d'ajouter une information de *transparence* ou de *texture*, chaque pixel est alors codé sur 32 bits [BEN07].

I.9.3.2 Les images indexées

Les images en *true colors* sont qualifiées comme de lourdes images lors de leurs manipulations. Dans le but de réduire la charge de travail et de gagner l'espace de stockage, on peut utiliser une autre représentation dite en couleurs *indexées*. Le principe consiste à déterminer le nombre de couleurs différentes utilisées dans l'image, puis à créer une table de ces couleurs en attribuant à chacune une valeur numérique correspondant à sa position dans la table (*palette*) de couleurs. Généralement, on attache une *palette* de 256 couleurs à l'image. Ces 256 couleurs sont choisies parmi les *16777216* couleurs de la *palette* RGB les plus pertinentes de l'image considérée [KAD99][BOU10].

Lors de la visualisation de l'image, la correspondance se fait entre le numéro de la couleur de chaque pixel (compris entre 0 et 255) et le code de la couleur RGB correspondant issu de la *palette* [BOU10]. Un exemple d'une image en couleurs *indexées* avec sa *palette* est donné à la figure I.19.

I.10 Les formats standards d'images

On distingue généralement deux grands types de formats d'images, les images *vectorielles* utilisées principalement dans le monde du graphisme et de la conception assistée par ordinateur et les images *matricielles* utilisées dans le domaine du traitement et de l'analyse d'images.

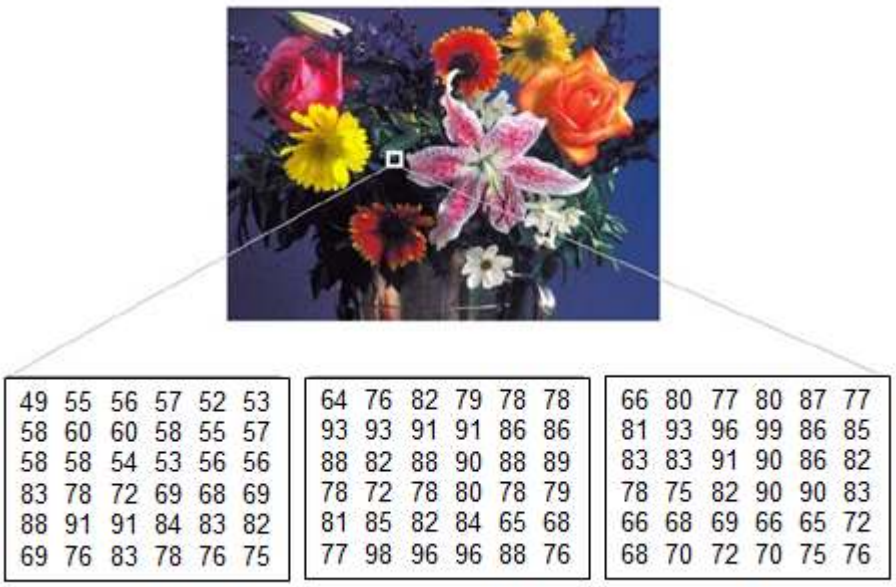


FIGURE I.18 – Exemple d'une image en couleurs (*Flowers.tiff*).

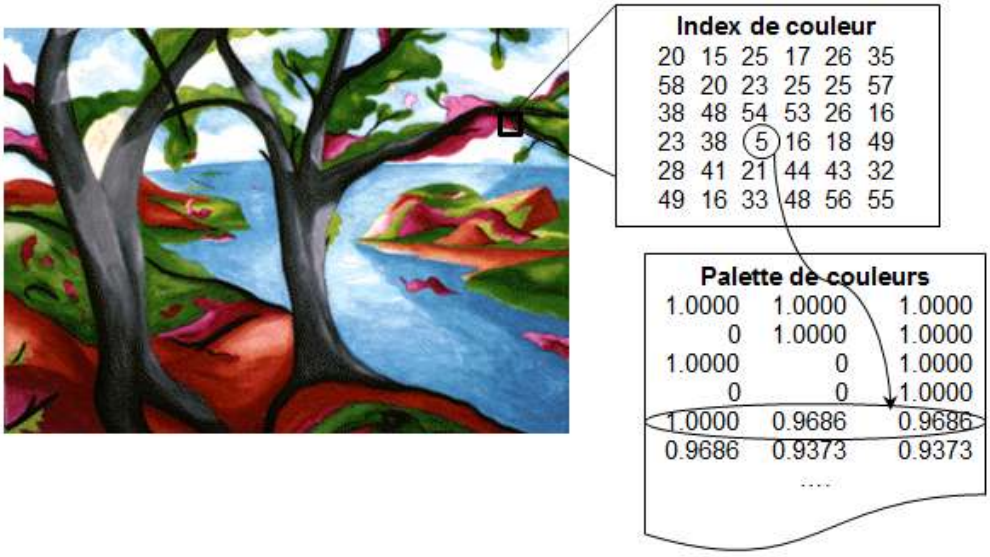


FIGURE I.19 – Exemple d'une image en couleurs *indexées* (*trees.tif*) avec sa *palette* associée.

I.10.1 Les images vectorielles

Une image *vectorielle* est une image numérique composée d'*objets* ou *formes géométriques* simples (*segments* de droite, *polygones*, *arcs* de cercle, etc.) définis d'un point de vue mathématique par divers attributs de *forme*, de *position*, de *couleur*, etc. Par exemple, un *cercle* est décrit par une information du type (*cercle*, *coordonnées du centre*, *rayon*, *couleur*,...).

Ce type d'images est essentiellement utilisé pour réaliser des dessins, des schémas ou des plans avec des logiciels tels que les logiciels de dessin industriel, logiciels de bu-

reautique, PAO, etc. La figure I.20 représente une image *vectorielle*. Il existe plusieurs formats pour ce type d'images dont on peut citer WMF, EPS, CGM, DXF et PCT.

Généralement, ces images occupent peu de place en mémoire et peuvent être *redimensionnées* sans perte d'informations et sans effet d'escalier (figure I.21.a) mais elles souffrent d'être inutilisables pour des images complexes et de certaines difficultés de compatibilité entre les différents formats.

FIGURE I.20 – Exemple d'une image *vectorielle*.

I.10.2 Les images matricielles

Une image *matricielle*, ou *bitmap* (carte de points) est constituée d'un ensemble de points (*pixels*) contenus dans une matrice, chacun de ces points possédant une ou plusieurs valeurs décrivant sa couleur. Ces images supportent mal les opérations de *redimensionnement* (figure I.21.b) ainsi que leurs grandes tailles en mémoire surtout pour les hautes définitions. Par contre, elles sont facilement adaptées au fonctionnement des écrans, convenables aux images complexes et flexibles avec les algorithmes de traitement d'images au niveau du pixel.

La majorité des images utilisées dans le domaine du traitement et de l'analyse d'images sont des images du type *matriciel* tels que les formats BMP, PCX, GIF, JPG, PNG et TIFF.

I.11 Conclusion

L'image est un ensemble de points structuré sous forme d'une matrice. La valeur de chaque élément de cette matrice représente la couleur du pixel correspondant. Une image est caractérisée par sa *dimension*, *résolution*, *luminance*, *contraste*, etc. Les images se divisent en plusieurs types : images *binaires* où chaque pixel est codé sur un *bit*, les images en *niveaux de gris* dont le pixel est codé sur un *octet* et les images en *couleurs* lorsque le pixel est codé dans *plusieurs* plans. Deux principaux formats d'images existent les images *vectorielles* essentiellement utilisées dans la conception assistée par

(a) (b)

FIGURE I.21 – Redimensionnement des images : (a) *vectorielles* et (b) *matricielles*.

ordinateur et les logiciels de dessin industriel et les images *matricielles* largement utilisées dans la vie quotidienne et dans le domaine du traitement et de l'analyse d'images.

Dans ce chapitre, nous avons vu les principaux concepts du domaine de la colorimétrie et les systèmes de couleurs dits de référence. Ces systèmes sont développés par différents organismes et selon divers contraintes technologiques dans le but de mieux représenter les images dans les divers domaines d'applications. Dans le chapitre suivant, nous allons présenter les différentes techniques de compression de données ainsi que les transformations et les méthodes de codage utilisées.

Chapitre II

État de l'art des méthodes de compression d'images

II.1 Introduction

L'utilisation des données numériques concerne de plus en plus plusieurs domaines de la science et de la technologie. La compression des images est appelée à prendre un rôle encore plus important en raison du développement des réseaux et du multimédia. Cela est dû surtout au retard qui existe entre les possibilités matérielles des dispositifs que nous utilisons (débits sur Internet, capacité des mémoires) et les besoins qu'exigent les utilisateurs (visiophonie, transfert de quantités d'informations toujours plus importantes dans des délais toujours plus brefs).

Bien que cette thèse porte sur la compression d'images (signaux 2D), on constate généralement que les approches adoptées pour ce type de données utilisent des schémas dérivés des méthodes de compression destinées aux signaux 1D. Il est donc important de présenter ces différents algorithmes afin de fournir une meilleure compréhension au lecteur.

Dans ce chapitre, nous allons passer en revue sur un état de l'art sur les différentes techniques de compression. Nous parlerons dans un premier temps des méthodes de compression *sans pertes*, puis des méthodes de compression *avec pertes* basées sur les transformations, pour finir sur les méthodes de codage sous bandes.

II.2 Classification de la compression

D'une façon générale, on peut classer la compression selon :

Le type de données qu'on manipule en compression *physique* ou *logique*. Dans la première classe, on traite directement sur les données en supprimant les redondances qui existent. Par contre ; dans la deuxième classe, on substitue les données originaux par d'autres équivalents mais moins volumineux.

Le type de méthodes utilisées en compression et décompression en compression *symétrique* ou *asymétrique*. Le premier type est fréquemment utilisé en transmissions de données. Dans ce cas, on utilise le même algorithme pour la compression et la décompression, donc la même quantité de travail effectué. Dans le deuxième type, la quantité de travail dans une direction et important que dans l'autre direction. Généralement, l'étape de compression nécessite beaucoup plus de temps et de ressources que l'étape de décompression. Mais dans l'archivage (fichiers compacts à accès rare), on utilise des algorithmes plus rapides en compression qu'en décompression.

Le type de données qu'on reconstitues en compression *sans pertes* ou compression *avec pertes*. Cette classification sera détaillée ci-dessous.

II.3 Méthodes de codage sans pertes

II.3.1 Codage entropique

Le codage *entropique* a comme but de s'approcher le plus possible de l'*entropie* du message définie par l'équation (II.1). Son principe de base est d'affecter à chaque symbole un code de longueur variable inversement proportionnel à sa probabilité d'apparition. Ainsi, les symboles les plus probables sont alors codés avec des codes courts et les symboles les moins probables sont alors codés avec des codes plus longs de manière à ce que la moyenne s'approche de l'*entropie* du message à coder. L'*entropie* d'un message est calculée par la formule :

$$H = -\sum_{i=1}^{M} p_i \times \log_2(p_i) \tag{II.1}$$

où p_i représente la probabilité du i^{eme} symbole de l'*alphabet* Ω du message.

Il existe plusieurs codeurs *entropiques*, dont les plus connus sont le codeur de *Huffman* et le codeur *arithmétique*.

II.3.1.1 Le codage de Huffman

En 1952, *Huffman* [HUF52] a mis en évidence une méthode pour construire des codes de longueur variables plus efficace que celle de *Shannon-Fano* [SHA48]. Après la construction de l'*alphabet*, cette méthode produit des codes binaires de préfixe unique suivant la probabilité d'apparition de chaque symbole de l'*alphabet*. Son principe est d'attribuer un code court à un symbole fréquent et un code plus long à un symbole plus rare.

L'algorithme considère chaque symbole à coder comme étant une *feuille* d'un *arbre* qu'il reste à construire. Cet *arbre* dit arbre de *Huffman* est construit de manière ascendante, à partir de ses *feuilles*. A chaque *nœud* de l'arbre, on associe la fréquence d'apparition de l'ensemble de ses *feuilles*. Un *nœud* est généré à partir de deux *nœuds* (*feuilles*) ayant la probabilité la plus faible. On construit ainsi un *arbre* jusqu'à ce qu'il ne reste plus que deux probabilités. Pour obtenir le code de chaque symbole, on parcourt l'*arbre* de sa racine jusqu'à la *feuille* qui lui correspond.

Généralement, ce type de codage est utilisé dans les normes de compression d'images (JPEG) ou de vidéos (MPEG). Mais cela n'empêche pas qu'il souffre d'inconvénients tels qu'il ne se rapproche pas suffisamment de l'*entropie* du signal puisque chaque symbole est codé par un nombre entier de bits [BEA97][BAT12].

Un exemple illustratif de l'application du codage de *Huffman* sur le premier bloc 8×8 de l'image en niveaux de gris *Lena* (figure II.1) est donné ci-dessous. Les probabilités associées aux différents symboles de l'*alphabet* ainsi que les différents codes binaires associés à chaque symbole de l'*alphabet* Ω sont données à la table II.1, la figure II.2 présente l'*arbre* généré.

L'*alphabet* Ω de ce bloc est : $\Omega = \{221,222,223,224,225,226,227,228,229,230\}$.

TABLE II.1 – Les probabilités associées aux différents symboles de l'*alphabet* Ω.

Symbole	**Niveau de gris**	**Probabilité**	**Code**
A	226	24/64	0
D	227	9/64	100
J	228	9/64	101
B	223	12/64	110
E	225	3/64	11100
I	229	3/64	11101
G	221	1/64	111100
H	222	1/64	111101
C	224	1/64	111110
F	230	1/64	111111

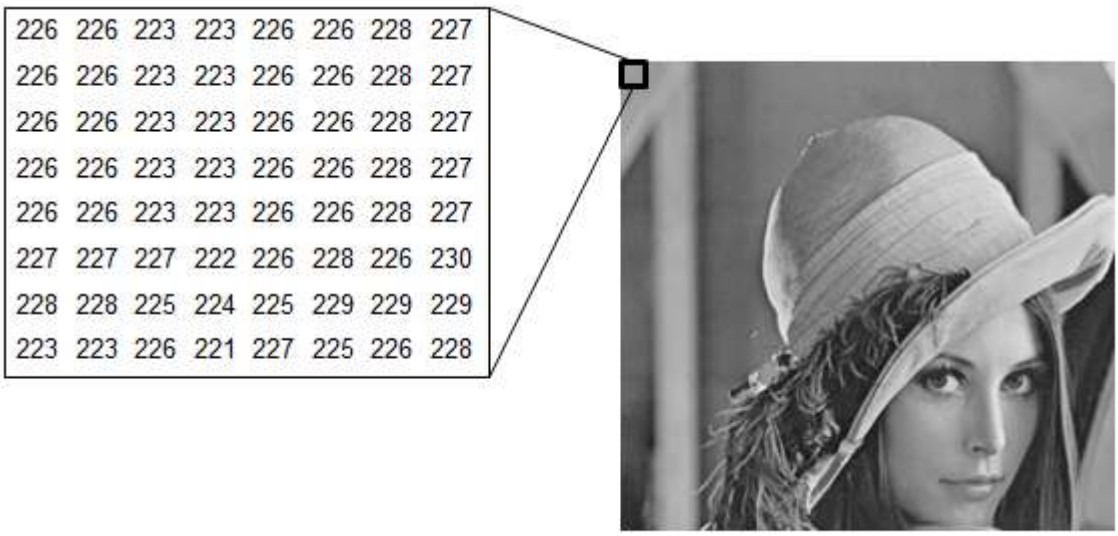


FIGURE II.1 – Le premier bloc 8×8 de l'image *Lena*.

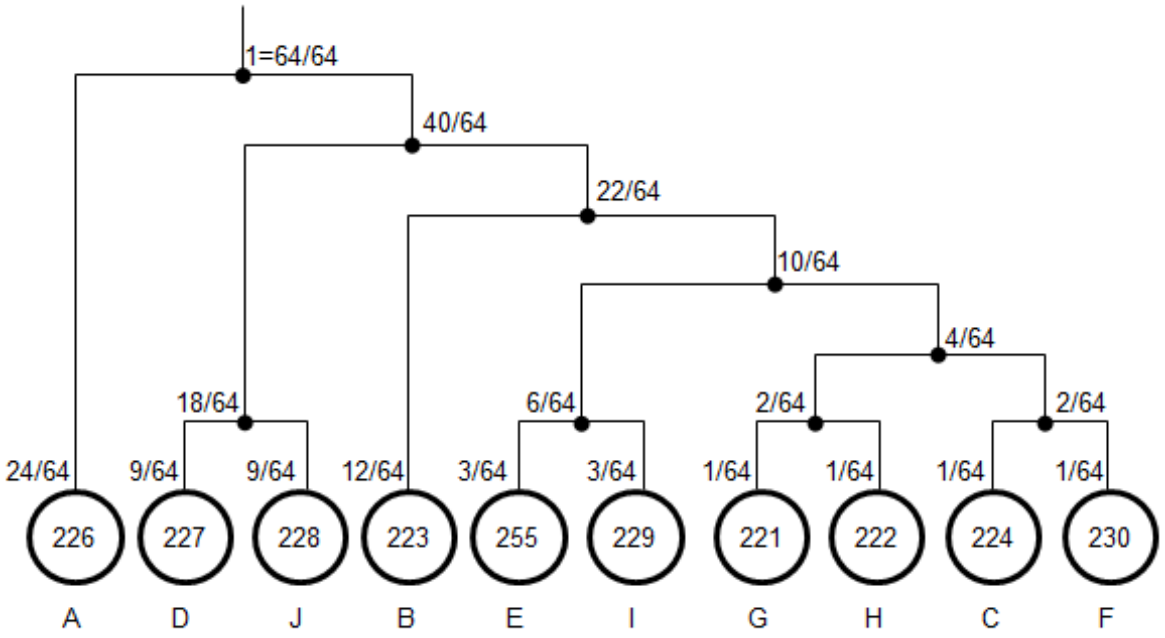


FIGURE II.2 – Arbre de *Huffman* généré à partir de l'exemple de la figure II.1.

A partir de la table II.1, l'*entropie* est égale à :

$$\begin{aligned} &-(\tfrac{24}{64} \times log_2\left(\tfrac{24}{64}\right) + \tfrac{12}{64} \times log_2\left(\tfrac{12}{64}\right) + \tfrac{9}{64} \times log_2\left(\tfrac{9}{64}\right) + \tfrac{9}{64} \times log_2\left(\tfrac{9}{64}\right) \\ &+\tfrac{3}{64} \times log_2\left(\tfrac{3}{64}\right) + \tfrac{1}{64} \times log_2\left(\tfrac{1}{64}\right) + \tfrac{1}{64} \times log_2\left(\tfrac{1}{64}\right) + \tfrac{1}{64} \times log_2\left(\tfrac{1}{64}\right) \\ &+\tfrac{1}{64} \times log_2\left(\tfrac{1}{64}\right)) \approx 2.3614\, bits/symbole. \end{aligned} \tag{II.2}$$

Avec le codage de *Huffman*, on obtient un code de longueur :
$24 \times 1 + 12 \times 3 + 9 \times 3 + 9 \times 3 + 3 \times 5 + 3 \times 5 + 1 \times 6 + 1 \times 6 + 1 \times 6 + 1 \times 6 = 168$ bits soit 2.65625 *bits/symbole*.

II.3.1.2 Le codage arithmétique

Le concept du codage *arithmétique* a été formalisé par *Abramson* et repris par *Rissanen* [RIS76], *Witten* [WIT87] et *Moffat* [MOF98]. Ce codage utilise un modèle statistique, tout comme le codeur de *Huffman*. Contrairement à ce dernier, le codage *arithmétique* traite l'ensemble des symboles comme une seule entité, et non pas symbole par symbole. Le code de sortie est modifié de façon incrémentale à chaque lecture d'un nouveau symbole. Ainsi le code de sortie est un nombre réel à virgule flottante compris entre 0 et 1, dont le nombre de chiffres après la virgule correspond au nombre de symboles [BAT12][BEA97]. Cela permet au codeur *arithmétique* de franchir la limite du 1 *bit/symbole* minimum et ainsi se rapprocher de manière quasi-idéale de l'*entropie* de la source.

L'algorithme repose sur la subdivision de l'intervalle $I = [L, U[$ ($[0, 1[$ à l'initialisation), conformément à la fonction de répartition de chaque symbole. La procédure du codage *arithmétique* peut se résumée en [BAT12] :

- Calculer la probabilité associée à chaque symbole a_i du message à coder,
- Initialiser l'*intervalle courant* à $I = [0, 1[$,
- Tant qu'il reste un symbole dans le message à coder :
  - Diviser l'*intervalle courant* I selon les fréquences d'apparition des symboles de l'*alphabet*,
  - Lire le prochain symbole S,
  - Calculer les limites du sous-intervalle représentant S selon l'équation :

$$\begin{cases} L_k = L_{k-1} + F(a_{i-1}) \times (U_{k-1} - L_{k-1}) \\ U_k = L_{k-1} + F(a_i) \times (U_{k-1} - L_{k-1}) \end{cases} \quad \text{(II.3)}$$

  - L'*intervalle courant* I devient le sous-intervalle représentant S.
- Toute valeur qui se situe à l'intérieur du dernier *intervalle courant* code le message de manière unique.

Le codeur *arithmétique* est plus performant que le codeur de *Huffman*, mais il est plus complexe à implémenter [BEA97].

Pour éclaircir les idées, l'application du codage *arithmétique* sur les 8 premières valeurs de l'exemple précédent est donnée ci-dessous :
Dans ce cas, l'*alphabet* $\Omega = \{A,B,C,D\} = \{223,226,227,228\}$ et le message à coder est : $m = \{226,226,223,223,226,226,228,227\} = \{BBAABBCD\}$.

Les probabilités d'apparitions p_i et la fonction de répartition F de chaque symbole sont données à la table II.2. La figure II.3 et la table II.3 illustrent les subdivisions successives de l'intervalle I.

TABLE II.2 – La fonction de répartition et les probabilités associées aux différents symboles de l'*alphabet*.

Symbole	**Niveau de gris**	**Probabilité**	F
A	223	2/8	0,25
B	226	4/8	0,75
C	227	1/8	0,785
D	228	1/8	1

TABLE II.3 – Les probabilités associées aux différents symboles de l'*alphabet*.

Étape	**Symbole lu**	L	U	**Code**
1	∅	0,0	1,0	0,5
2	B	0,25	0,75	0,50
3	B	0,375	0,625	0,500
4	A	0,3750	0,4375	0,4063
5	A	0,37500	0,39063	0,38281
6	B	0,378907	0,386719	0,382813
7	B	0,3808594	0,3847657	0,3828125
8	C	0,38427734	0,38476573	0,38452148
9	D	0,384643555	0,384704590	0,384674072

II.3.2 Codage par plages

Le *codage par plages* RLE ou (*Run-Length Encoding*) est l'un des codages les plus simples. Il est basé sur le codage des répétitions de symboles contenues dans un signal ou dans une image. Par exemple, les images contiennent généralement des zones uniformes dans lesquelles les pixels ont la même valeur. Le but du codage RLE est d'exploiter ces répétitions en générant des codes du type (*longueur, valeur*) ce qui permet de réduire la taille des données [BAT12].

Ce type de codage est très efficace dans les situations où l'on a de longues séquences d'un même symbole tel que les images en *noir* et *blanc*. Dans le cas contraire, le gain

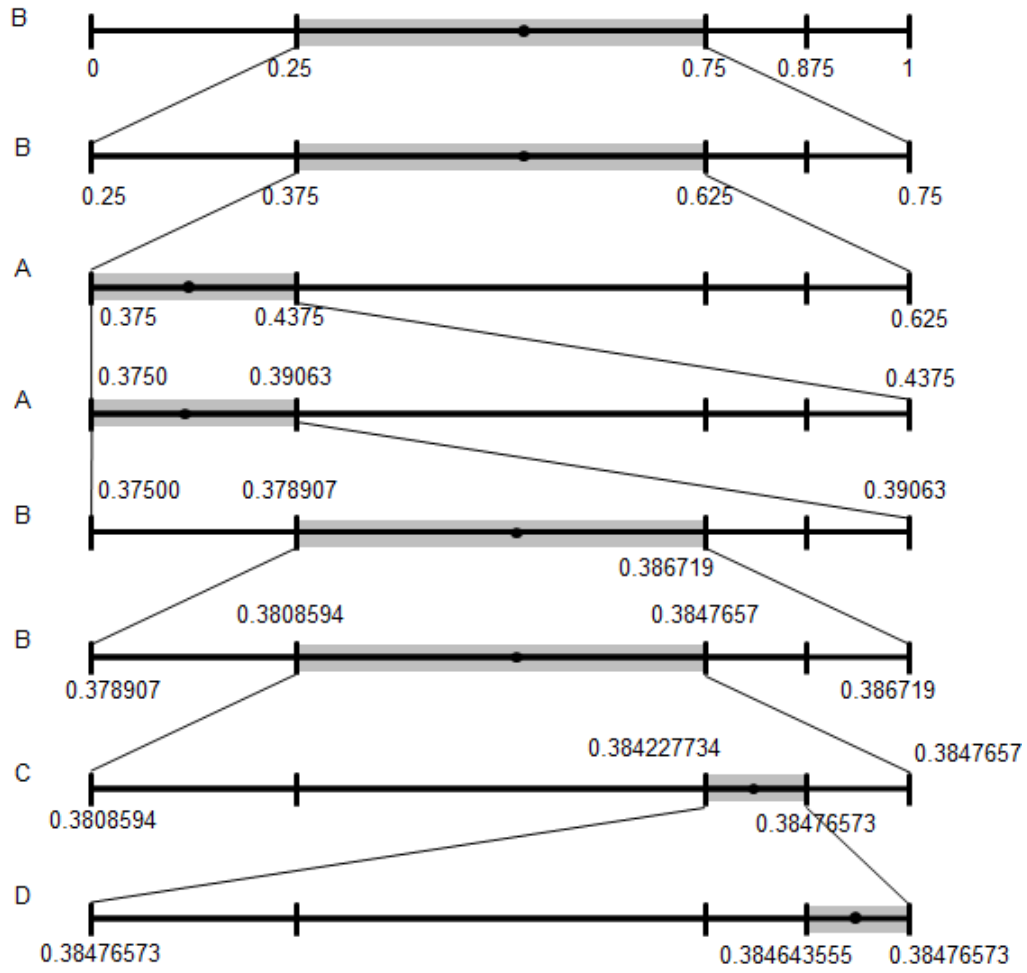


FIGURE II.3 – Subdivisions successives de l'intervalle I.

.

obtenu par le codage des répétitions va rapidement être limité par l'encodage des symboles non répétés.

Le codage RLE est souvent utilisé dans les méthodes de compression par transformation pour le codage des coefficients quantifiés tronqués ou arrondis ; coefficients DCT ou coefficients d'*ondelettes* ; où on repère d'importantes plages de valeurs nulles, ce qui se prête bien à ce type de codage [BEA97].

Reprenons l'exemple de la section précédente, le message à coder est :

$m=$ {*226,226,226,226,226,227,228,223,226,226,226,226,226,227,228,223,223,223,223,223,223,227, 225,226,223,223,223,223,223,222,224,221,226,226,226,226,226,226,225, 227,226,226,226,226, 226,228,229,225,228,228,228,228,228,226,229,226,227,227,227,227,227,230,229,228*}.

Ce bloc d'image nécessite 64 *octets* pour son stockage. L'application du codage RLE sur ce bloc génèrera la séquence suivante :

$S=$ (5,*226*)-(1,*227*)-(1,*228*)-(1,*223*)-(5,*226*)-(1,*227*)-(1,*228*)-(6,*223*)-(1,*227*)-(1,*225*)-(1,*226*)
(5,*223*)-(1,*222*)-(1,*224*)-(1,*221*)-(6,*226*)-(1,*225*)-(1,*227*)-(5,*226*)-(1,*228*)-(1,*229*)-(1,*225*)
(5,*228*)-(1,*226*)-(1,*229*)-(1,*226*)-(5,*227*)-(1,*230*)-(1,*229*)-(1,*228*).

Cette séquence ne nécessite maintenant que 60 *octets*.

II.3.3 Codage à base de dictionnaires

Ziv et *Lempel* en 1977 [ZIV77], ont mis au point le premier algorithme à base de *dictionnaires*. Une année après [LEM78], ils ont améliorés leur premier algorithme en introduisant un *dictionnaire* sur tous les symboles précédemment rencontrés et non pas par une fenêtre. En 1984, *Lempel, Ziv* et *Welch* [Wel84] ont publiés leur nouvel algorithme appelé *LZW* qui est une amélioration du fameux *LZ78*. Dans ce type d'algorithmes, le codeur lit un groupe de symboles et recherche des équivalences avec des chaînes de symboles déjà rencontrées.

Dans cet algorithme, le *dictionnaire* est initialisé avec les symboles constituant l'*alphabet* de base. Chaque nouveau symbole lu est concaténé pour former une chaîne I tant que cette chaine est présente dans le *dictionnaire*. Dans le cas inverse, la chaîne I est ajoutée au *dictionnaire* et on recommence le processus en prenant comme chaîne de départ le dernier symbole lu [BEA97][BAT12].

L'application de l'algorithme *LZW* sur les 8 premières valeurs de l'image *Lena* (exemple de la section précédente) est donnée ci-dessous.
Le message à coder est :
m=$\{$*226,226,223,223,226,226,228,227*$\}$=$BBAABBCD$ (table II.2).

Le codage se déroule comme suit :

1. Le premier symbole à coder est "B". Ce symbole existe dans le *dictionnaire*.
2. Puisque le symbole "B" existe dans le *dictionnaire*, le symbole suivant "B" est concaténé au symbole "B" pour former le symbole "BB".

   - Ce dernier n'existe pas dans le *dictionnaire*, il faut l'ajouter au *dictionnaire* et aura 256 comme *index*.
   - On ajoute au code l'*index* du symbole "B" (66).

3. Puisque un nouveau symbole "BB" est ajouté au *dictionnaire*, on reprend du dernier symbole "B". Le symbole suivant "A" est concaténé au symbole "B" pour former le symbole "BA".

- Ce dernier n'existe pas dans le *dictionnaire*, il faut l'ajouter au *dictionnaire* et aura 257 comme *index*.
- On ajoute au code l'*index* du symbole "B" (66).

4. Puisque un nouveau symbole "BA" est ajouté au *dictionnaire*, on reprend du dernier symbole "A". Le symbole suivant "A" est concaténé au symbole "A" pour former le symbole "AA".

   - Ce dernier n'existe pas dans le *dictionnaire*, il faut l'ajouter au *dictionnaire* et aura 258 comme *index*.
   - On ajoute au code l'*index* du symbole "A" (65).

5. Puisque un nouveau symbole "AA" est ajouté au *dictionnaire*, on reprend du dernier symbole "A". Le symbole suivant "B" est concaténé au symbole "A" pour former le symbole "AB".

   - Ce dernier n'existe pas dans le *dictionnaire*, il faut l'ajouter au *dictionnaire* et aura 259 comme *index*.
   - On ajoute au code l'*index* du symbole "A" (65).

6. Puisque un nouveau symbole "AB" est ajouté au *dictionnaire*, on reprend du dernier symbole "B". Le symbole suivant "B" est concaténé au symbole "B" pour former le symbole "BB".

   - On ajoute au code l'*index* du symbole "B" (256).

7. Puisque le symbole "BB" existe dans le *dictionnaire*, le symbole suivant "C" est concaténé au symbole "BB" pour former le symbole "BBC".

   - Ce dernier n'existe pas dans le *dictionnaire*, il faut l'ajouter au *dictionnaire* et aura 260 comme *index*.
   - On ajoute au code l'*index* du symbole "C" (67).

8. Puisque un nouveau symbole "BBC" est ajouté au *dictionnaire*, on reprend du dernier symbole "C". Le symbole suivant "D" est concaténé au symbole "C" pour former le symbole "CD".

   - Ce dernier n'existe pas dans le *dictionnaire*, il faut l'ajouter au *dictionnaire* et aura 261 comme *index*.
   - On ajoute au code l'*index* du symbole "D" (68).

A la fin, le code de sortie est : $\{(66),(66),(65),(65),(256),(67),(68)\}$. Les étapes d'évolution de l'algorithme LZW de cet exemple sont données à la figure II.4.

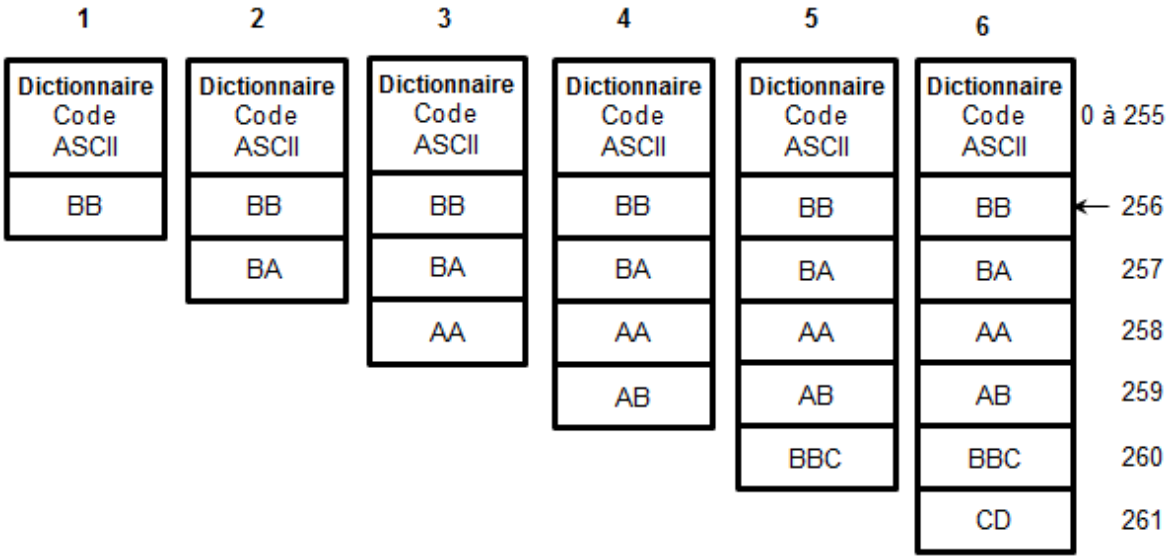


FIGURE II.4 – Étapes d'évolution de l'algorithme LZW.

.

II.4 Méthodes de codage avec pertes

Les méthodes compression *sans pertes* sont souvent utilisées dans des domaines bien spécifiques pour lesquels aucune dégradation ne doit altérer l'image. L'utilisation de ces méthodes est limitée étant donné qu'elles ne permettent pas d'avoir des taux de compression élevés. Les méthodes compression *avec pertes*, dites *non conservatives* ont la faculté d'accroitre considérablement le taux de compression en permettant des distorsions dans l'image reconstruite. Afin que ces distorsions soient le moins visibles possibles, les algorithmes de compression d'images *avec pertes* doivent prendre en compte les différentes propriétés du système de vision humain et exploiter à leur avantage ses faiblesses.
Concernant le système de vision humain, on accepte de manière générale les deux assertions suivantes [GOU02][BAT12] :

- L'œil humain est plus sensible à la *luminance* qu'à la *chrominance*,
- L'œil humain est moins sensible aux détails (hautes fréquences) d'une image qu'à son allure générale (basses fréquences).

Une image en couleurs est caractérisée par une double redondance :

- Une redondance colorimétrique entre chaque composante.
- Une redondance spatiale 2D des pixels appartenant à une même zone de l'image.

Ces deux types de redondances devront être exploités au maximum afin de décorréler au mieux l'information et proposer des taux de compression les plus élevés possibles.

II.4.1 Codage par quantification

La *quantification* est un processus qui permet d'associer à un nombre réel (respectivement un vecteur de nombres réels) un nombre entier (respectivement un vecteur de nombres entiers). Dans un certain sens, on peut considérer qu'elle réalise une compression implicite (passage des réels aux entiers). Elle permet de réduire le nombre de bits nécessaire à la représentation de l'information en tenant en compte de l'aspect psychovisuel (l'*œil*) ou psychoacoustique (l'*oreille*) ce qui permet de déterminer la distorsion tolérable à apporter au signal à coder. De cette façon, elle est l'une des sources de perte d'informations dans le système de compression.

Généralement, il existe deux types de quantification : la quantification *scalaire* (QS) et la quantification *vectorielle* (QV) [TOT07][WAK93].

II.4.1.1 Quantification scalaire

La quantification dans son sens le plus général est l'approximation d'un signal d'amplitude *continue* par un signal d'amplitude *discrète*. Généralement, on la définit comme l'association de chaque valeur réelle x, à une autre valeur q qui appartient à un ensemble fini de valeurs $\{q_1, q_2, \ldots, q_L\}$. A toute valeur de x comprise dans l'intervalle $[x_n, x_{n+1}]$, on fait correspondre une valeur quantifiée q_i située dans cet intervalle. Cette valeur q peut être exprimée en fonction de la troncature utilisée : soit par l'arrondi *supérieur*, l'arrondi *inférieur*, ou l'arrondi le *plus proche*.
La quantification *scalaire* est réalisée indépendamment pour chaque valeur par [CZI99] :

$$\begin{aligned} QS: \quad & \mathbb{R} \longmapsto \{q_1, q_2, ..., q_L\} \\ & x \longmapsto q_i \end{aligned} \tag{II.4}$$

$$QS(x) = \lfloor \frac{x + 0.5}{q} \rfloor \tag{II.5}$$

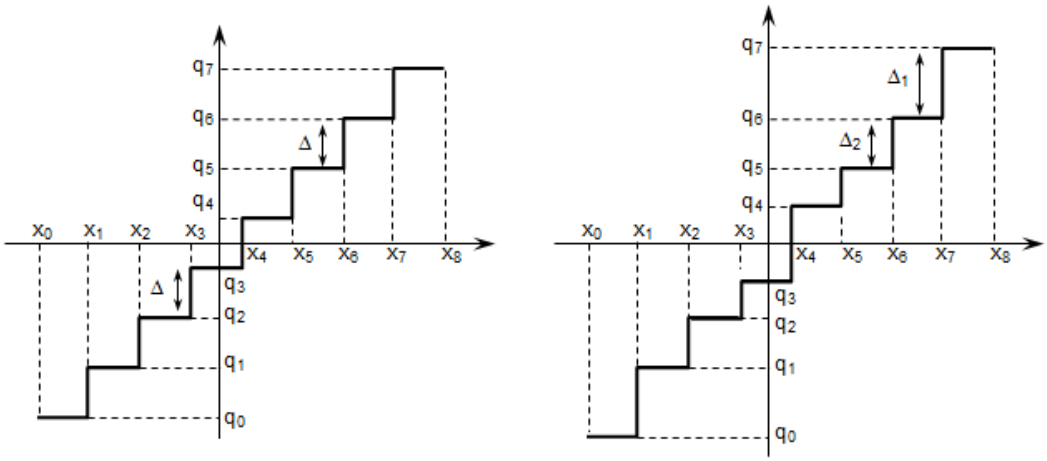
avec $\lfloor \cdots \rfloor$ désigne l'arrondi choisi.
Pour quantifier un signal $X = \{x_1, x_2, \cdots, x_N\}$ prenant des valeurs entre X_{min} et X_{max}, il faut suivre les étapes suivantes :

1. Diviser l'intervalle $[X_{min}, X_{max}]$ en L niveaux différents $q_1, q_2, \cdots, q_L$,

2. Associer à chaque intervalle $[x_n, x_{n+1}[$ une valeur q_i,

3. Chaque valeur $x_n \in X$ sera coder par q_i si $x_n \in [x_n, x_{n+1}[$.

L'écart $\triangle$ entre chaque valeur q s'appelle le *pas de quantification*. Un écart $\triangle$ constant définit une quantification scalaire *uniforme*. Dans le cas où $\triangle$ n'est pas constant, la quantification est dite *non-uniforme*. La figure II.5 montre un exemple d'une quantification *scalaire*. Le fait d'arrondir le signal X provoque une erreur de quantification appelée *bruit de quantification*.



(a) (b)

FIGURE II.5 – Quantification *scalaire* (a) *uniforme* et (b) *non uniforme*.

.

II.4.1.2 Quantification vectorielle

Contrairement à quantification *scalaire*, la quantification *vectorielle* (QV) s'effectue sur un groupe d'éléments de la source représenté par un vecteur $\vec{X}$ de dimension N. Elle est considérée comme une généralisation de la QS. La QV consiste alors à remplacer le vecteur $\vec{X}$ par un vecteur $\vec{C_i}$ de même dimension appartenant à un *dictionnaire* C. Ce dernier est un ensemble fini de *vecteurs codes*.

On appelle quantificateur *vectoriel* de dimension N et de taille L l'application de $\mathbb{R}^N$ dans un ensemble fini C contenant L vecteurs de dimension N [TOT07][MOR09] définie par :

$$\begin{array}{rccl} QV: & \mathbb{R}^N & \longmapsto C & avec \quad C = \{\vec{C_1}, ..., \vec{C_L}\} \in \mathbb{R}^N \\ & \vec{X} & \longmapsto & \vec{C_i} = QV(\vec{X}) \end{array} \tag{II.6}$$

La quantification *vectorielle* se décompose en général en deux processus : un processus de *codage* (*codeur*) et un processus de *décodage* (*décodeur*). Le rôle du *codeur*

consiste, pour tous vecteur du signal (bloc d'image) en entrée à rechercher dans le *dictionnaire* C le *code-vecteur* $\vec{C}_i$ le plus *proche* du vecteur source $\vec{X}$. La notion de proximité utilise la règle du plus *proche voisin* ; au sens de la distance euclidienne entre deux vecteurs ; pour réaliser le codage. C'est uniquement l'indice i du *code-vecteur* $\vec{C}_i$ ainsi sélectionné qui sera transmis, c'est donc à ce niveau que s'effectue la compression.

$$\begin{array}{rccccc} QV: & \mathbb{R}^N & \longmapsto & C & \longmapsto & I \\ & \vec{X} & \longmapsto & \vec{C}_i & \longmapsto & QV(\vec{X}) = i \end{array} \tag{II.7}$$

Cette opération en comprimant les données perd de l'information et le signal original ne pourra plus être restitué exactement. Le *décodeur*, quant à lui, dispose d'une réplique du *dictionnaire* et consulte celui-ci pour fournir le *code-vecteur* correspondant à l'indice reçu. Le *décodeur* réalise donc l'opération de décompression [TOT07][TAQ11] :

$$\begin{array}{rccc} DQV: & I & \longmapsto & C \\ & i & \longmapsto & \vec{C}_i \end{array} \tag{II.8}$$

On dit également que C représente l'*alphabet* de reproduction et $\vec{C}_i$ les symboles de reproduction conformément au vocabulaire habituel en théorie de l'information.

L'élaboration du *dictionnaire* est une phase très importante pour la QV. Elle est faite à partir d'un processus d'apprentissage et peut être obtenue selon l'algorithme de LBG (*Linde*, *Buzo*, *Gray*) [LIN80][TAQ11]. La figure II.6 illustre le schéma de principe de la quantification *vectorielle*.

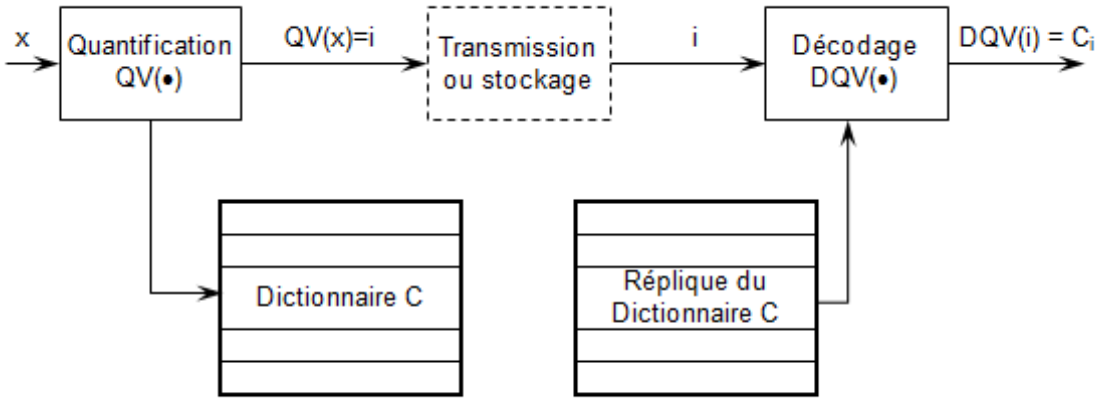


FIGURE II.6 – Schéma général de la quantification *vectorielle*.

II.4.2 Codage par prédiction

Les méthodes utilisant le codage par *prédiction* exploitent directement la corrélation entre un symbole (pixel) et ses voisins. On prédit la valeur d'un symbole à partir des

valeurs précédemment codées. Seul l'écart entre la valeur réelle et la valeur prédite est codé et envoyé au *décodeur*. Dans le cas le plus simple, on code le premier symbole puis on calcule la différence avec le second symbole et on code cette différence. Celle dernière nécessite moins de bits que les symboles eux-mêmes car cette différence est souvent faible. On code ensuite la différence entre le deuxième symbole et le troisième, etc. On obtient alors une suite de symboles d'une dynamique plus faible que la suite originale et qui par conséquent peut être codée efficacement.

Dans des situations plus complexes, on utilise une fonction de *prédiction* plus complexe que la simple différence pour estimer la valeur d'un symbole en fonction de la valeur de ses symboles voisins. On code alors l'erreur de *prédiction*, qui est l'écart entre la vraie valeur du symbole et la valeur prédite [TOT07][BEA97][WAK93]. La fonction de *prédiction* peut être plus ou moins complexe selon [BEA97] :

- l'ordre de *prédiction* (nombre de symboles impliqués dans la fonction de *prédiction*),
- la *topologie* (position des voisins utilisés dans le calcul),
- l'utilisation de *pondération* (poids affectés aux voisins selon leurs positions relatives par rapport au symbole prédit ou selon les propriétés statistiques de l'ensemble de symboles).

Les équations (II.9) à (II.11) représentent un codage par *prédiction* du $3^{ème}$ ordre où les coefficients α_k indiquent l'importance relative accordée aux voisins $x(i-k)$.

$$\textit{Prédiction} : \quad \hat{x}(i,j) = \alpha_1 \cdot x(i-1,j-1) + \alpha_2 \cdot x(i-1,j) + \alpha_3 \cdot x(i,j-1) \tag{II.9}$$

$$\text{Erreur de } \textit{prédiction} \ : \quad e(i,j) = x(i,j) - \hat{x}(i,j) \tag{II.10}$$

$$\text{Reconstitution} : \quad x(i,j) = \hat{x}(i,j) + e(i,j) \tag{II.11}$$

Le codage *prédictif* ainsi décrit correspond à la *modulation par impulsion et codage différentielle* (MICD) ou DPCM en anglais.

II.4.3 Codage par transformation

Les méthodes de compression par *transformation* n'agissent pas directement sur l'image d'entrée, mais leur action est dans le domaine de la transformée. Dans ces méthodes, l'image à compresser de dimension $N \times N$ est subdivisée en sous-images ou blocs de taille réduite. Chaque bloc subit une *transformation* mathématique orthogonale inversible linéaire du domaine spatial vers le domaine fréquentiel, indépendamment

des autres blocs [WAK93].

Pour la compression, on utilise des transformées inversibles avec de bonnes propriétés de décorrélation, afin que l'information contenue dans l'espace transformé soit plus compacte que dans l'espace d'origine. La décomposition de l'image dans une base de fonctions adéquate permet d'avoir des coefficients indépendants et garantit qu'un petit nombre d'eux contiennent une proportion importante de l'énergie de l'image. Ainsi, on pourra excluait le reste sans nuire de manière significative ni à la quantité d'énergie, ni à l'aspect visuel de l'image reconstruite.

Généralement, on utilise des transformations linéaires à bases orthogonales puisqu'elles possèdent des expressions analytiques simples et facilement implémentable. Les méthodes de codage par *transformation* suivent le schéma de principe présenté dans la figure II.7. Plusieurs transformations existent en littérature parmi elles, on peut citer la *transformation de Hadamard*, la *transformée de Karhunen-Loève*, la *transformée en cosinus discrète* et la *transformée en ondelettes*.

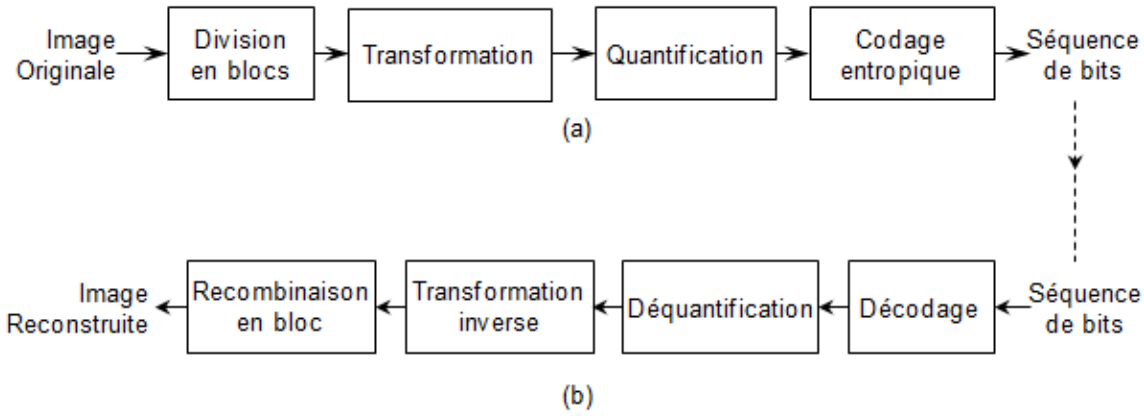


FIGURE II.7 – Schéma de principe de la compression/décompression par *transformation*.

Remarque : la quantité de calcul demandée pour effectuer la transformation sur l'image entière est très élevée pour cela, on préfère de subdivisée en sous-images ou blocs de taille réduite.

II.4.3.1 Transformation de Karhunen-Loève

La transformation de *Karhunen-Loève* KLT réalise une décomposition du vecteur X sur les vecteurs propres de sa matrice de covariance. En effet la matrice d'autocorrélation de X est donnée par [GOU02][ISA02][SAY06] :

$$R_{XX} = E(X \cdot X^{*^T}) \tag{II.12}$$

Notons par Φ la matrice qui diagonalise R_{XX} :

$$\Phi^{*T} \cdot R_{XX} \cdot \Phi = \Lambda \tag{II.13}$$

La matrice étant diagonale définie les valeurs propres de R_{XX}, notées λ_k ($\forall k \in \{1, ..., N\}$). La Transformation de *Karhunen-Loève* correspond à la matrice Φ^{*T} et son application à X produit un signal Y définit par :

$$Y = \Phi^{*T} \cdot X \tag{II.14}$$

Ainsi la matrice de covariance de Y est donnée par :

$$R_{YY} = E(Y \cdot Y^{*T}) = \Phi^{*T} \cdot E(X \cdot X^{*T}) \cdot \Phi = \Phi^{*T} \cdot R_{XX} \cdot \Phi = \Lambda \tag{II.15}$$

Cette transformée est optimale au sens où tous les coefficients obtenus sont décorrélés et que la quasi-totalité de l'énergie est conservée par un minimum de coefficients. En d'autres mots, si on décidait de ne garder qu'un nombre limité de coefficients transformés, les coefficients KLT sont ceux qui contiendraient la plus grande fraction de l'énergie totale par rapport à toutes les autres transformations possibles. Malheureusement, la matrice de transformation de la KLT dépend de l'image à compresser ce qui conduit à deux inconvénients très importants [TOT07][BEA97][WAK93] :

- Une lourde charge calcul pour la détermination des valeurs propres, vecteurs propres et de la matrice de covariance de l'image.
- Elle n'a pas d'algorithme de calcul rapide.

Pour ces raisons, la KLT est très peu utilisée en compression malgré sa supériorité théorique. On lui préfère des transformations qui sont indépendantes des images et qui ont des algorithmes rapides, tels que les transformations *spectrales* ou en *ondelettes*.

II.4.3.2 Transformations spectrales

La *transformation de Fourier discrète* (DFT) et ses dérivées sont des transformations importantes dans le domaine du traitement du signal et d'image. La DFT permet de passer du domaine spatial au domaine fréquentiel en décomposant une image (bloc d'image) par des bases sinusoïdales selon la formule (II.16) :

$$F(u, v) = \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} I(n, m) \cdot e^{-2\pi \cdot (\frac{u \cdot n}{N} + \frac{v \cdot n}{M})} \tag{II.16}$$

La fonction inverse permettant de remonter l'image originale I connaissant sa transformée F est :

$$I(n,m) = \frac{1}{N \times M} \sum_{u=0}^{N-1} \sum_{v=0}^{M-1} F(u,v) \cdot e^{-2\pi \cdot (\frac{u \cdot n}{N} + \frac{v \cdot n}{M})} \tag{II.17}$$

La variable de l'espace transformé étant la fréquence, une telle décomposition permet de mieux observer la répartition fréquentielle de l'image. Étant donné que ce sont les premiers harmoniques qui contiennent la quasi-totalité de l'énergie, il est donc possible de mettre à zéro une proportion importante des coefficients et de coder l'image à moindre coût.

La DFT décompose l'image en coefficients réels et imaginaire pouvant se convertir en module et argument ce qui n'est pas facile à interpréter. Les traitements de ces données peuvent s'avérer lourds, pour cela, une variante de la transformée de *Fourier* dite *rapide* (FFT) a été développée pour assouplir ce coût. D'une manière générale, on lui préfère la *transformation en cosinus discrète* (DCT) qui bénéficie de toutes les caractéristiques de la FFT mais avec des coefficients complètement réels [TOT07].

La DCT fera l'objet d'une étude détaillée au chapitre suivant car elle est à la base du travail élaboré dans cette thèse.

II.4.3.3 Transformation en ondelettes

Il est bien connu que la *Transformée de Fourier* et ses dérivées ont l'inconvénient majeur d'ignorer complètement la contribution temporelle exacte d'une fréquence dans un signal. En effet, elles permettent de représenter un signal fini comme une somme de fonctions sinusoïdales (et donc de longueurs infinies) en supposant que celui-ci est périodique en dehors de ses bornes de représentation [TAQ10].

En pensant à résoudre ce problème, *Gabor* en 1946 a introduit sa transformée appelée *Transformée de Fourier à Court Terme* (*Short Time Fourier Transform* STFT). Après une segmentation du signal à analyser en tranches de temps fixes (fenêtrage), la STFT applique la TF à chaque tranche. Une fenêtre de temps longue produise une bonne résolution fréquentielle mais la résolution temporelle pauvre. Par contre, une fenêtre de temps courte donne une résolution fréquentielle pauvre mais elle produit une résolution temporelle acceptable.

Ce compromis difficile à satisfaire pour la STFT a été résolu avec la découverte de la *Transformée en Ondelettes* (*Wavelets Transform* WT) [BEN08]. Contrairement à ces approches, la WT permet une représentation localisée, *temps/fréquence*, du signal en le projetant sur des fonctions oscillantes à support borné, appelées *ondelettes*.

La *Transformée de Fourier*, la STFT et la *Transformée en Ondelettes* sont définies par les équations suivantes :

$$TF(\tau, f) = \int_{-\infty}^{+\infty} f(t) \cdot e^{j2\pi ft} dt \tag{II.18}$$

$$TSFT(\tau, f) = \int_{-\infty}^{+\infty} f(t) \cdot g(t-\tau) \cdot e^{j2\pi ft} dt \tag{II.19}$$

$$WT(\tau, f) = \frac{1}{\sqrt{s}} \int_{-\infty}^{+\infty} f(t) \cdot \psi(\frac{t-\tau}{s}) \cdot e^{j2\pi ft} dt \tag{II.20}$$

La transformation en *ondelettes* et la SFTF décomposent le signal *f(t)* en multitude de coefficients dans le domaine *temps/fréquence*. La figure II.8 montre les découpages du plan *temps/fréquence* de la TF, STFT et la WT. Pour la STFT, la taille fenêtre $g(t)$ est constante pour toutes les fréquences par contre pour la WT celle-ci dépend de la fréquence. Ainsi, les transitoires rapides d'un signal sont prises en compte par les coefficients de hautes fréquences qui ont une résolution temporelle très fine, alors que les variations lentes de basses fréquences peuvent bénéficier d'une représentation fréquentielle plus précise [SIM05].

Ces *ondelettes* sont issues d'une *ondelette mère* ψ par *décalage* temporel τ et par *dilatation* s et forment ainsi une famille d'*ondelettes* [TAQ10].
Les transformées en *ondelettes* se divisent en deux grands types :

- La *transformée en ondelettes continue* ou *Contiuous Wavelet Transform* CWT,
- La *transformée en ondelettes discrète* ou *Discrete Wavelet Transform DWT*.

II.4.3.4 la transformée en ondelettes continue

La transformées en *ondelettes continue* utilise des *translations* et des *dilatations* d'une manière continue pour générer une famille d'*ondelettes*. Soit ψ une fonction de $L^2(\mathbb{R})$ et sa transformée de *Fourier* $\Psi(\omega) = TF(\psi(t))$, si cette transformée vérifie :

1. La condition d'admissibilité :

$$C_\psi = \int_0^\infty \frac{|\Psi(\omega)^2|}{\omega} < \infty \tag{II.21}$$

2. Si ψ a au moins un moment nul, c'est-à-dire $\Psi \in L^1(\mathbb{R})$ ou encore :

$$\int_{\mathbb{R}} \psi dt = 0 \tag{II.22}$$

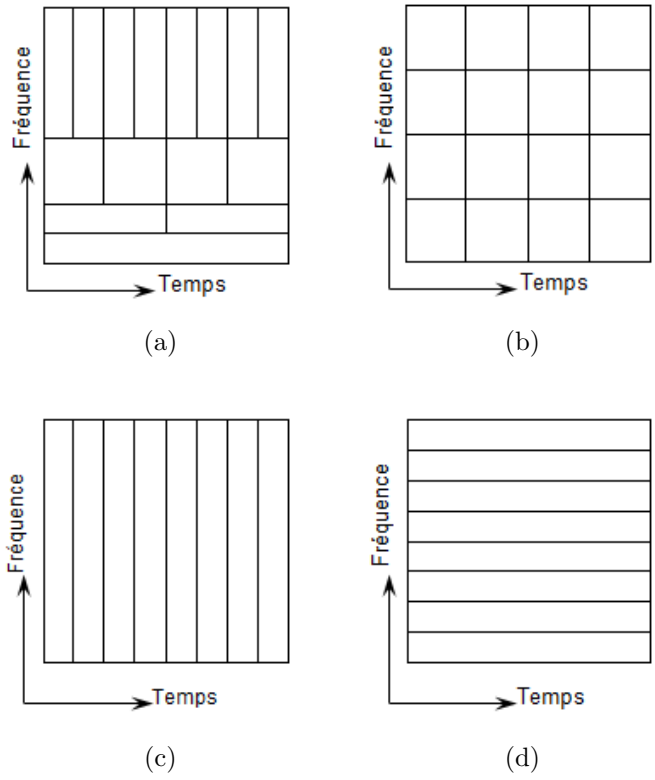


FIGURE II.8 – Découpage en *temps/fréquence* : (a) Transformée en *ondelettes*, (b) STFT, (c) Échantillonnage temporel et (d) TF.

Alors la fonction ψ est appelée *ondelette mère*. Autours de celle-ci, on peut construire une famille d'*ondelettes* $\psi_{s,\tau}(t)$ par *dilatation* et *translation* définie par :

$$\forall(s,\tau) \in \Gamma, \quad \psi_{s,\tau}(t) = \frac{1}{\sqrt{|s|}}\psi(\frac{t-\tau}{a}) \quad \text{avec} \quad \Gamma = \mathbb{R}_+^* \times \mathbb{R} \tag{II.23}$$

Le paramètre s est appelé paramètre d'*échelle* et permet de *dilater* l'*ondelette mère* $\psi(t)$, alors que le paramètre b permet de la *translater* comme il est claire à la figure II.9. Le paramètre d'échelle s permet d'obtenir des *ondelettes comprimées* (*support réduit*) ainsi que des *ondelettes dilatées* (*support étendu*). Les *ondelettes comprimées* sont utilisées pour déterminer les composantes hautes fréquences tandis que les *ondelettes dilatées* permettent de déterminer les composantes basses fréquences. Le paramètre τ, quant à lui, permet d'analyser le signal par *translations* successives jusqu'à ce qu'il soit entièrement parcouru [TOT07].

La figure II.10 illustre un exemple d'analyse d'un signal avec *dilatation* et *translation* d'une *ondelette*.

La CWT associée à cette *ondelette mère* $\psi(t)$ est alors définie par :

$$\forall f \in L^2(\mathbb{R}), \forall(s,\tau) \in \Gamma \quad CWT_f(s,\tau) = \frac{1}{\sqrt{|s|}} \int_{\mathbb{R}} f(t) \cdot \psi(\frac{t-\tau}{s})dt \tag{II.24}$$

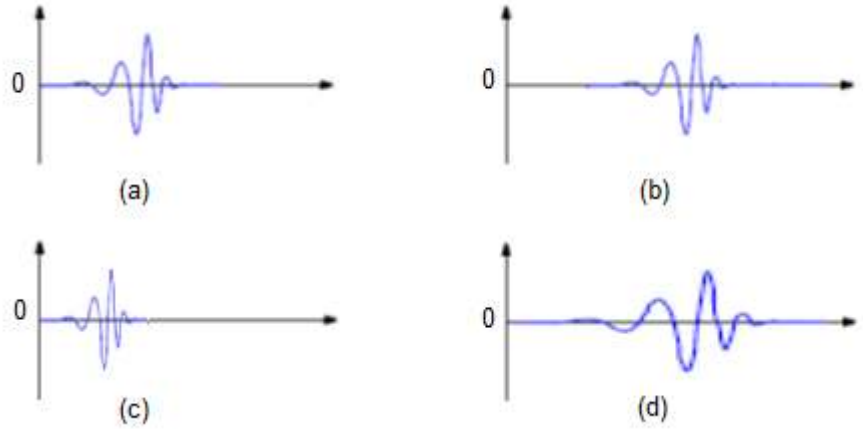


FIGURE II.9 – Illustration de la variation du facteur d'*échelle* et de *translation* (a) L'*onde mère*, (b) *Ondelette translatée*, (c) *Ondelette comprimée* pour $0 < s < 1$ et (d) *Ondelette dilatée* ($s > 1$) et *translatée*.

La CWT peut alors être définie par le produit scalaire :

$$CWT_f(s,\tau) =< \psi_{s,\tau}, f > \tag{II.25}$$

La reconstruction du signal original est effectuée par la transformée en inverse donnée par l'équation :

$$f(t) = \frac{1}{C_\psi} \cdot \int_{\mathbb{R}_+} \int_{\mathbb{R}} CWT_f(s,\tau) \cdot \psi(\frac{t-\tau}{s}) \cdot \frac{ds}{s^2} d\tau \tag{II.26}$$

Le signal $f(t)$ est vu comme une superposition d'*ondelettes*.
Dans le cas bidimensionnel, l'*ondelette* mère $\psi_{s,\tau}(x,y)$ est translatée et dilatée dans les directions x et y selon l'équation suivante :

$$\psi_{s,\tau}(x,y) = \frac{1}{\sqrt{|s_x| \cdot |s_y|}} \cdot \psi(\frac{t-\tau_x}{s_x}, \frac{t-\tau_y}{s_y}) \tag{II.27}$$

Alors que la transformée en *ondelettes continue* de la fonction $f(x,y)$ est donnée par :

$$CWT_f(s,\tau) = \frac{1}{\sqrt{|s_x| \cdot |s_y|}} \cdot \int\int f(x,y)\psi(\frac{t-\tau_x}{s_x}, \frac{t-\tau_y}{s_y}) dxdy \tag{II.28}$$

A noter que les coefficients $CWT_f(s,\tau)$ sont plus importants en valeurs absolues si la corrélation entre $\psi_{s,\tau}(x,y)$ et $f(x,y)$ est plus élevée.

II.4.3.5 La transformée en ondelettes discrètes

La *transformée en ondelettes continue* ainsi définie est fortement redondante puisque l'*ondelette* est *translatée* et *dilatée* d'une manière continue. D'une autre part, la na-

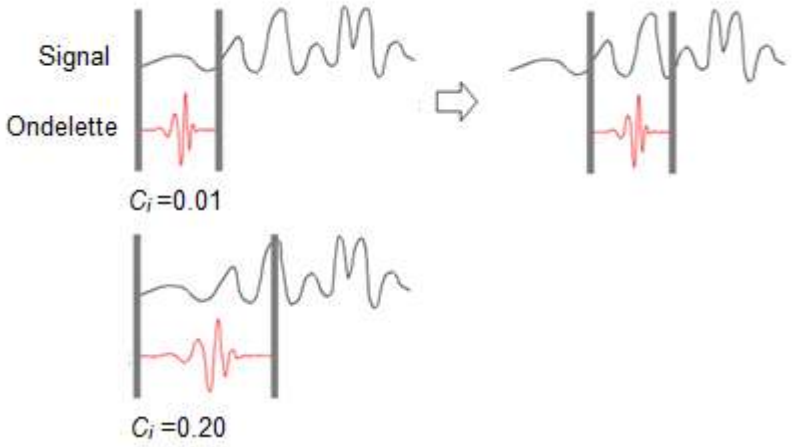


FIGURE II.10 – Décomposition en *ondelettes* d'un signal.

ture numérique des signaux à analyser, nous amène à discrétiser la transformation elle-même. Plusieurs discrétisations sont possibles, mais celle qui est, de loin, la plus étudiée fournit une décomposition en *octaves*, ce qui correspond :

$$s_m = s_0^{-m} \quad \text{et} \quad \tau_n = n \cdot \tau_0 \tag{II.29}$$

avec $(m, n) \in \mathbb{Z}$ et $s_0 > 1$ $\tau_0 > 0$.

Le réseau $\{((s_m, \tau_n)|(m, n)) \in \mathbb{Z}^2\}$ définit alors une grille *dyadique*. La famille d'*ondelettes discrètes* $\psi_{m,n}(t)$ issue de l'*ondelette* mère $\psi(t)$ s'écrit alors :

$$\forall (m, n) \in \mathbb{Z}^2, \quad \psi_{m,n}(t) = \frac{1}{\sqrt{s_m}} \psi(\frac{t - \tau_n}{s_m}) = s_0^{m/2} \cdot \psi(s^m \cdot (t - n \cdot \tau_0)) \tag{II.30}$$

Dans le cas particulier $s_0 = 2$ et $\tau_0 = 1$, il existe des *ondelettes* telles que s et τ peuvent être discrétisés de manière que les $\psi_{m,n}$ forment une base orthonormale [DAV04][TOT07].

$$\forall (m, n) \in \mathbb{Z}^2, \quad \psi_{m,n}(t) = 2_0^{-m/2} \cdot \psi(2^{-m} \cdot (t - n)) \tag{II.31}$$

La décomposition en *ondelettes* d'un signal f peut s'effectuer selon la forme :

$$f = \sum_{m,n} C_{m,n} \cdot \psi_{m,n} \tag{II.32}$$

avec $C_{m,n}$ les coefficients d'*ondelettes* qui mesurent les variations locales du signal.

L'obtention de ces coefficients est donnée par la relation suivante :

$$C_{m,n} = \int \psi_{m,n}(t) \cdot f(t)dt \tag{II.33}$$

L'implémentation de la DWT est réalisée à l'aide de l'algorithme de *Mallat* [MAL89] en utilisant l'*analyse multirésolution*. Cet algorithme est basé sur la définition d'une paire de filtres d'analyse appelés filtres à miroirs quadratiques LA (filtre basses fréquences) et HA (filtre hautes fréquences). Le signal $f(n)$ est décomposé par les deux filtres d'analyse LA et HA pour donner deux signaux $f_b(n)$ et $f_h(n)$ contenant chacun N échantillons. Le problème réside dans le fait qu'après cette première étape de filtrage, on multiplie par deux le nombre de coefficients ce qui n'est pas idéal pour la compression.

Toutefois, conformément au théorème de *Nyquist*, on peut *sous-échantillonner* les signaux $f_b(n)$ et $f_h(n)$ par un facteur de 2 sans perdre d'informations car la plage des fréquences a elle aussi été divisée par deux. Cette opération de *sous-échantillonnage* est généralement appelée *décimation* (*downsampling*) [BAT12][BEN08]. Les deux vecteurs résultants respectivement cA (*Coefficients d'Approximation*) et cD (*Coefficients de Détails*) sont de taille approximativement égale à la moitié du vecteur d'origine. La figure II.11 illustre la décomposition en cA et cD et la figure II.12 donne la décomposition d'un signal *sinus bruité* [MIS04].

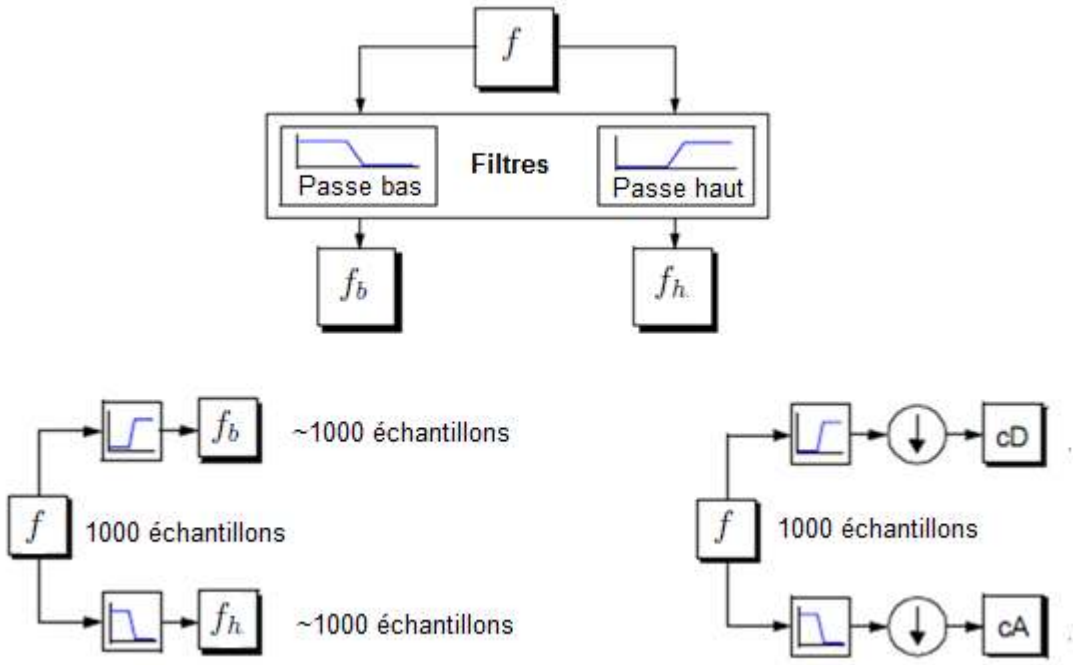


FIGURE II.11 – Décomposition en *Approximation* et *Détails*.

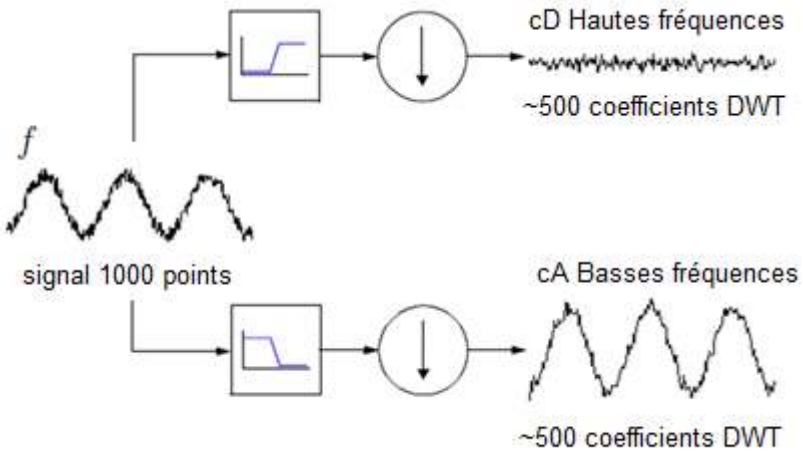


FIGURE II.12 – Exemple de décomposition en coefficients d'*Approximation cA* et coefficients de *Détails cD* [MIS04].

La reconstruction parfaite du signal original $f(n)$ à partir des vecteurs cA et cD est théoriquement possible par une opération de *sur-échantillonnage* (*upsampling*) suivie d'un filtrage par les filtres LR (filtre passe bas de reconstruction) et HR (filtre passe haut de reconstruction). Le quadruple (LA, HA, LR et HR) forme un banc de filtres miroirs en quadrature. La figure II.13 représente la *décomposition/reconstruction* par DWT-1D à l'aide de bancs de filtres.

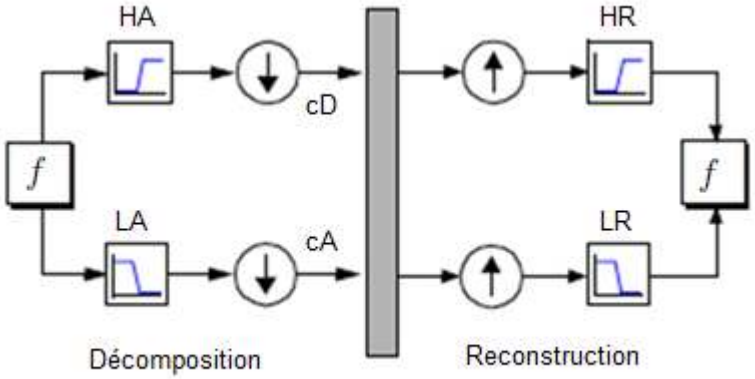


FIGURE II.13 – *Décomposition/Reconstruction* par DWT à l'aide de bancs de filtres.

Cas bidimensionnel

D'après *Mallat* [MAL89], dans le cas des signaux bidimensionnels tels que les images, la fonction d'*ondelette* monodimensionnelle ψ avec la fonction d'échelle ϕ peut être séparée en trois fonctions d'*ondelettes* distinctes (ψ_H, ψ_V et ψ_D) [DAV04][TOT07] :

- $\psi_H = \phi(x) \cdot \psi(y)$ exprime les variations selon l'axe *horizontal*,
- $\psi_V = \psi(x) \cdot \phi(y)$ exprime les variations selon l'axe *vertical*,
- $\psi_D = \psi(x) \cdot \psi(y)$ exprime les variations selon les deux axes (*diagonal*).

La DWT *bidimensionnelle* est souvent effectuée à l'aide de filtres 2D séparables qui permettent une implémentation rapide. Ainsi, pour chaque niveau de décomposition, la DWT est appliquée sur les lignes de l'image, générant des coefficients basses fréquences et des coefficients hautes fréquences *horizontaux*. Elle est ensuite de nouveau appliquée sur les colonnes des deux sous-ensembles (sous-bandes) ainsi obtenus. Cette approche est appelée décomposition *dyadique* ou décomposition en bandes par *octave* [TAQ10].

Cela, peut s'obtenir simplement par des convolutions de l'image avec des filtres miroirs en quadrature. A chaque niveau, on filtre l'image par des filtres d'analyse *passe-bas* et *passe-haut* et on la *sous-échantillonne* avec un facteur de 2 ce qui permet d'obtenir quatre sous-images chacune de taille quatre fois plus petite (figure II.14) :

- une image de faible résolution,
- une image de détails *horizontaux*,
- une image de détails *verticaux*,
- une image de détails *diagonaux*.

La DWT peut être modélisée par le schéma de la figure II.15 où *HDWT* et *VDWT* sont respectivement la décomposition en *ondelettes* horizontale et verticale. La sous-image LL_k est la représentation basse résolution de l'image lors de l'étape précédente (LL_0 est l'image originale). Après filtrage horizontal de LL_k, on obtient les sous images L_{k+1} celle de la basse résolution et H_{k+1} celle de la haute résolution. Ces sous images sont filtrées dans le sens vertical pour donner LL_{k+1} qui corresponde à la représentation basse résolution et les sous-images HL_{k+1} , LH_{k+1} et HH_{k+1} pour les représentations haute résolution dans le sens horizontal, vertical et diagonal.

De nombreux types d'*ondelettes* ont été proposés dans la littérature [DAU92]. On peut recenser les plus utilisées. Les *ondelettes* de *Morlet* et de *Sombrero* sont des *ondelettes* continues, tandis que les *ondelettes* (orthogonales) de *Haar*, *Shannon*, *Meyer*, *Spline* et *Daubechies* sont des *ondelettes* discrètes [TOT07].

II.5 Codage en sous-bandes

Depuis son introduction pour le codage de la parole à débit réduit, par *Crochiere et al* en 1976 [CRO76], le codage en *sous–bandes* s'est avéré une technique particulièrement intéressante. L'idée de base du codage en *sous-bandes* est la décomposition

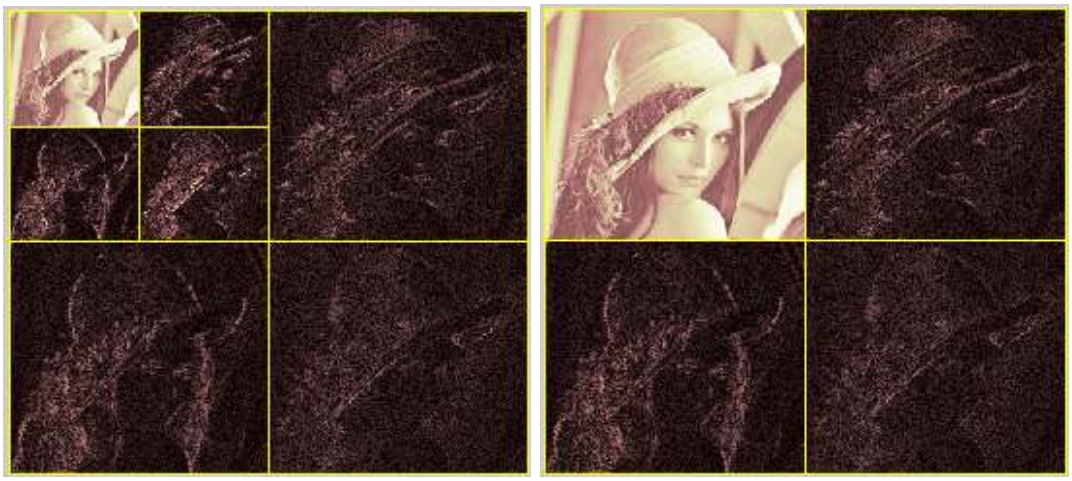


FIGURE II.14 – Exemple d'une décomposition en *ondelettes* de l'image *Lena* (ondelette *Bior 4.4*) à : (a) 1 niveau et (b) 2 niveaux de résolution.

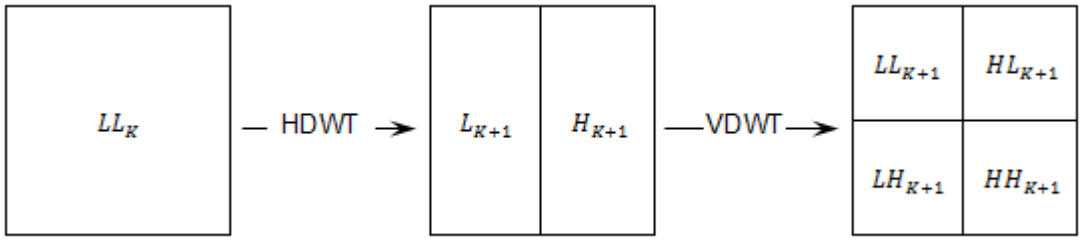


FIGURE II.15 – Décomposition par *ondelettes* 2D, Filtrage successif *Horizontal/ Vertical*.

du signal original en différentes bandes de fréquences complémentaires. L'avantage d'une telle décomposition est la possibilité de coder séparément les bandes avec un codeur approprié [RAF94].

Le codage en *sous–bandes* utilise une représentation assimilable à une représentation par transformation d'image. Celle-ci est filtrée de façon à générer un ensemble de *sous-images* ou *sous-bandes* de fréquences complémentaires. Les *sous-bandes* étant de bande de fréquence limitée, il est possible de les *sous-échantillonner*. La figure II.16 montre la décomposition d'une image en quatre *sous-bandes*, la première correspondant aux basses fréquences (*approximation* de l'image), la deuxième aux hautes fréquences *colonnes* (*détails horizontaux*), la troisième aux hautes fréquences *lignes* (*détails verticaux*), la quatrième aux hautes fréquences *lignes* et *colonnes* (*détails diagonaux*). Après une décomposition en *sous-bandes* et un *sous-échantillonnage*, les *sous-bandes* résultantes sont codées avec des stratégies adaptées à leur contenu énergétique.

Tout comme les méthodes par transformation, on tend à privilégier les basses fréquences qui sont riches en énergie, et à coder plus grossièrement les hautes fréquences

en respectant les particularités psycho-acoustiques (*son*) ou psycho-visuelles (*image*) de chaque bande [BEA97][RAF94].

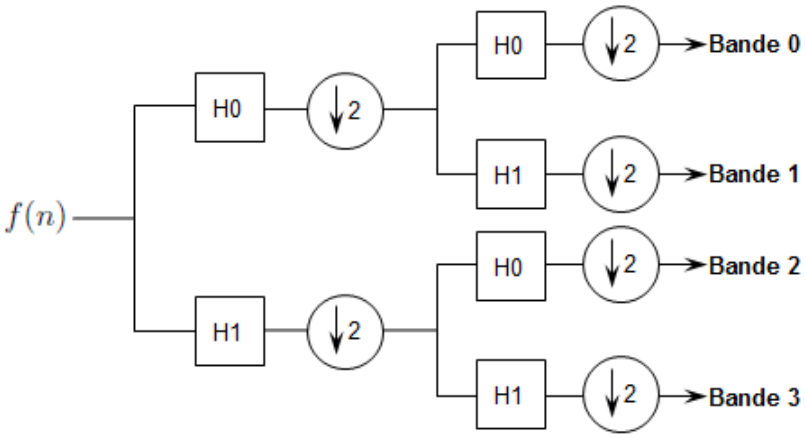


FIGURE II.16 – Décomposition en *sous-bandes*, H_0 filtre passe bas, H_1 filtre passe haut.

II.5.1 Algorithme de codage EZW

Introduit par *Shapiro*, EZW [SHA93] (*Embedded Zerotree Wavelet* ou *Encapsulation d'ondelettes par arbres de zéros*) est l'un des premiers codeurs basés sur les *ondelettes* à se démarquer. L'idée de l'algorithme EZW est de trouver le meilleur ordre de transmission des coefficients *ondelettes*, qui est l'ordre décroissant de leur valeur absolue. Ceci permet à l'algorithme EZW de faire une transmission progressive de l'image puisque le décodeur peut s'arrêter n'importe où dans la suite de bits transmise et produire la meilleure image reconstruite possible avec cette suite de bits tronquée [CHO05][TAQ10].

L'hypothèse principale de cet algorithme suppose la décroissance d'énergie des coefficients pour les plus hautes *résolutions/fréquences* dans l'arbre hiérarchique. Ainsi, les coefficients de faibles valeurs (*non significatifs*) dans les basses fréquences ont de fortes chances d'avoir des *descendants* (*enfants*) *non significatifs* dans les hautes fréquences. Ceux possédant une énergie importante ont généralement des *enfants* possédant moins d'énergie. Dans ce cas, si un coefficient d'*ondelette* W_{ij} à une certaine échelle est *non signifiant* pour un seuil T_n donné, alors tous ses *descendants* aux échelles plus fines ayant la même orientation ont une forte probabilité d'être *non signifiants* pour ce seuil T_n. Un coefficient W_{ij} est dit *non signifiant* pour un seuil T_n si $|W_{ij}| < T_n$.

Pour cela, EZW procède au regroupement des coefficients *non significatifs* sous forme d'arbre de zéros (*zerotree*). La structure *zerotree* permet de détecter les zones de l'image qui ne contiennent pas d'informations *significatives* et qui sont codées ensuite

en arbre de zéros [OUA09]. L'algorithme se déroule en un nombre fini d'étapes chacune composé de deux passes. Après avoir calculé la DWT de l'image, l'algorithme code les coefficients transformés W_{ij} à l'aide d'une suite décroissante de seuils :

$$T_n = \frac{T_0}{2^j} \quad \text{avec} \quad T_0 = |\frac{W_{max}}{2}| \tag{II.34}$$

où W_{max} est le coefficient DWT le plus grand en valeur absolue.

L'algorithme effectue récursivement deux passes successives, ne traitant à chaque fois que les coefficients *significatifs* par rapport au seuil courant T_n.

II.5.1.1 Passage signifiance (dominante)

Le codage des coefficients DWT passe par la détermination de deux listes de coefficients :

- Une *liste dominante D* contenant les coordonnées des coefficients qui sont *non significatifs* par rapport au seuil actuel T_n suivant le chemin donné par la figure II.17.a.
- Une *liste subordonnée S* contenant les valeurs des amplitudes des coefficients déjà trouvés *significatifs*.

Dans cette première passe, l'algorithme parcourt les coefficients suivant l'ordre indiqué dans la *liste D* à la recherche de ceux *significatifs* par rapport au seuil courant. Une liste C appelée *carte de signifiance* est obtenue en associant à chaque coefficient suivant sa valeur absolue et celle de ses *enfants* l'un des symboles suivants :

- ZeroTree (ZT) pour un arbre de coefficient *non significatif* et ayant tous ces *descendants non significatifs* aussi,
- Isolated Zero (IZ) pour un coefficient *non significatif* ayant un ou des *descendants* qui sont *significatifs*,
- *Positive Significant* (PS) et *Negative Significant* (NS) pour un coefficient *significatif* suivant son signe.

Chaque coefficient *significatif* est ensuite mis à zéro dans la DWT afin que sa position ne soit plus encodée et sa valeur absolue est placée dans la liste S pour la coder par approximations successives (passe *subordonnée*) en plus ses coordonnées sont retirées de la *liste D*. Les coefficients de la *liste S* passe par une étape de quantification et raffinement puis un codage entropique. Par contre les symboles de la *carte de signifiance C* sont codés par un codage entropique adaptatif [OUA09][BOU13].

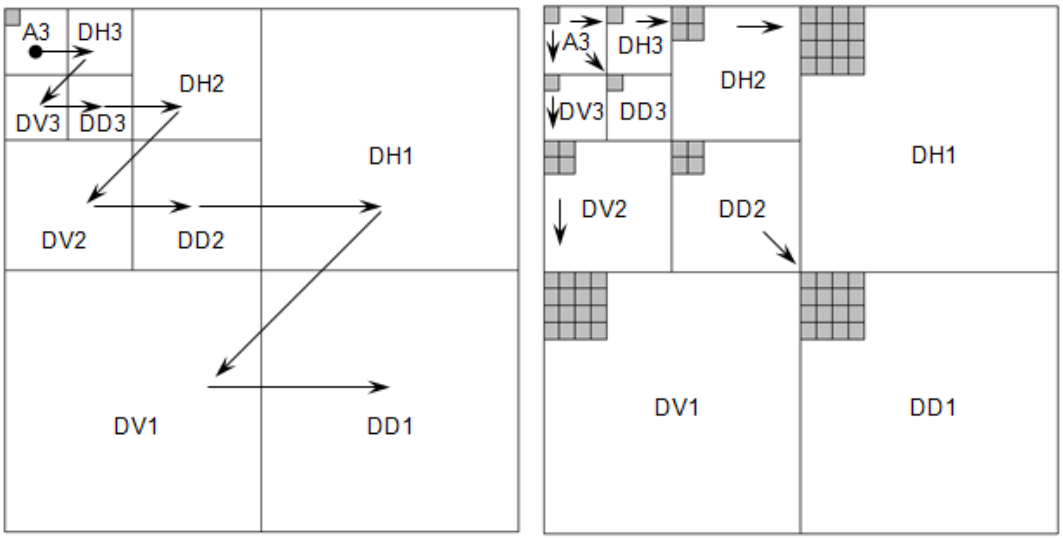


FIGURE II.17 – Les relations entre les coefficients d'*ondelettes* dans les différents *sous-bandes*. (a) ordre de parcours des coefficients et (b) modèle de dépendances *inter-bandes*.

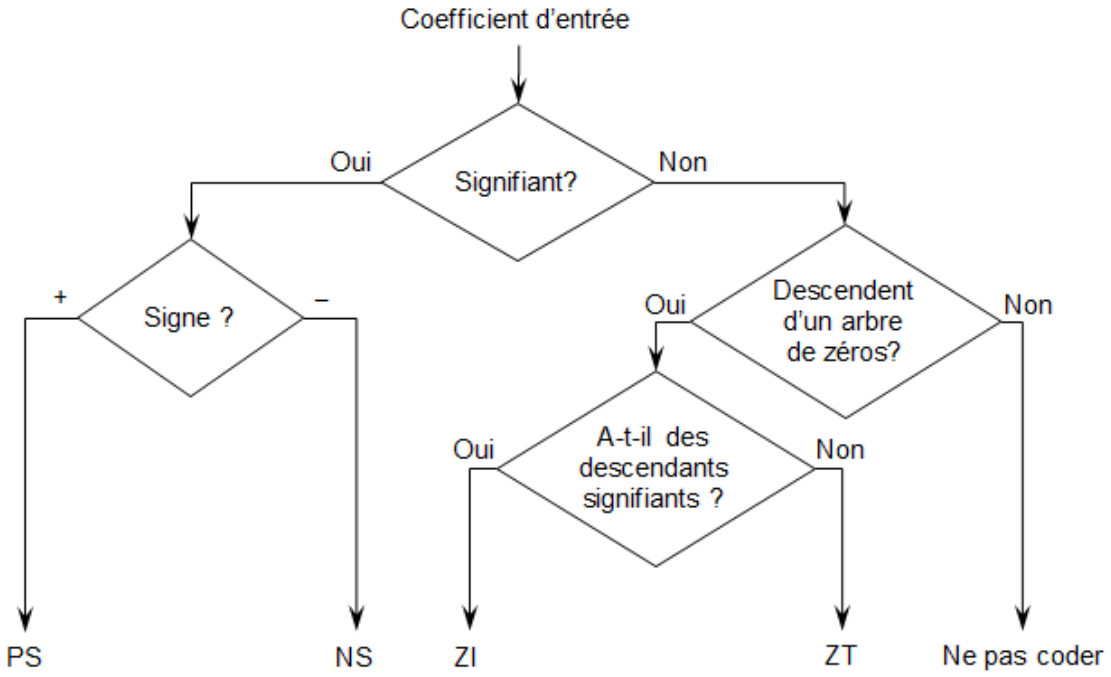


FIGURE II.18 – Test de *signifiance* des coefficients pour l'algorithme EZW.

II.5.1.2 Passe subordonnée (Quantification et Raffinement)

Cette étape s'attache au codage des éléments de la *liste S*. Cette liste contient les amplitudes des coefficients *significatifs* par rapport au seuil actuel T_n. Ces coefficients sont quantifiés par un quantificateur scalaire uniforme défini par rapport à ce seuil. L'étape de *quantification* se déroule en deux temps [OUA09] :

- *Raffinement* des coefficients qui étaient *significatifs* aux itérations précédentes,
- *Quantification* des nouveaux arrivants à la *liste S*.

Les intervalles de *quantification* sont définis par $[T_n, 2T_n]$ où n est le numéro de l'itération.

La *quantification* consiste à affecter un bit "*0*" aux coefficients dont la valeur absolue se trouve à la première moitié de cet intervalle c.-à.-d $[T_n, \frac{3}{2}T_n[$ alors que ceux se trouvant dans le second intervalle $]\frac{3}{2}T_n, 2T_n]$ on leur associe un bit "*1*".

Le *raffinement* consiste à diviser les sous-intervalles par deux. Ainsi, le bit "*0*" est affecté aux coefficients appartenant à la première moitié de l'intervalle alors que le bit "*1*" est affecté à la seconde moitié. Les procédures de *quantification* et de *raffinement* sont illustrées à la figure II.19.

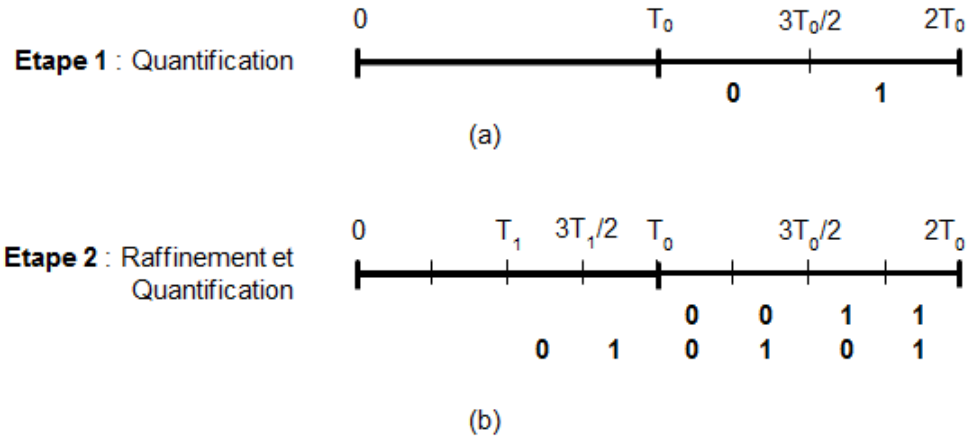


FIGURE II.19 – Principe de *quantification* et de *raffinement*.

Ce processus récursif s'arrête lorsque T_{N-1} est atteint (N niveau de décomposition), le nombre de bits désiré a été transmis ou le critère de qualité de l'image est satisfait [CHO05].

Exemple d'application

Prenons comme exemple celui de [SHA93] qui est donné à la figure II.20. Il correspond aux coefficients d'*ondelettes* d'une image 8×8 en trois niveaux de décomposition. Dans cet exemple, la valeur maximale des coefficients étant *63*, cela donne $T_0 = 32$.

Etape 1 :

Test de signifiance

En parcourant les coefficients de la figure II.20.a selon l'ordre de la figure II.17.a, on obtient la figure II.20.b. Les coefficients de cette figure qui n'ont pas de code appartient

à un *zerotree*. Les coefficients *signifiant* sont placés dans la *liste subordonnée* S et leurs symboles serons placés dans la *carte de signifiance* C.

63	-34	49	10	7	13	-12	7
-31	23	14	-13	3	4	6	-1
15	14	3	-12	5	-7	3	9
-9	-4	-14	8	4	-2	3	2
-5	9	-1	47	4	6	-2	2
3	0	-3	2	3	-2	0	4
2	-3	6	-4	3	6	3	6
5	11	5	6	0	3	-4	4

(a)

63 PS	-34 NS	49 PS	10 ZT	7 IZ	13 IZ	-12	7
-31 IZ	23 ZT	14 ZT	-13 ZT	3 IZ	4 IZ	6	-1
15 ZT	14 IZ	3	-12	5	-7	3	9
-9 ZT	-4 ZT	-14	8	4	-2	3	2
-5	9	-1 IZ	47 PS	4	6	-2	2
3	0	-3 IZ	2 IZ	3	-2	0	4
2	-3	6	-4	3	6	3	6
5	11	5	6	0	3	-4	4

(b)

FIGURE II.20 – Exemple d'application de l'algorithme EZW.

Passe subordonnée

La liste des coefficients *significatifs* pour cette itération est S={63,-34,49,47}. Selon la figure II.21 et pour un seuil $T_0 = 32$, ces coefficients seront quantifiés comme suit :

$$63 \longrightarrow 1, \qquad -34 \longrightarrow 0, \qquad 49 \longrightarrow 1, \qquad 47 \longrightarrow 0.$$

Nous obtenons la séquence de bits "*1010*". Le résultat des différentes itérations de l'algorithme EZW est donné à la table II.4. A noter qu'afin de simplifier l'affichage des résultats, les symboles *PS, NS, IZ* et *ZT* ont été remplacés par *P, N, Z* et *T* respectivement.

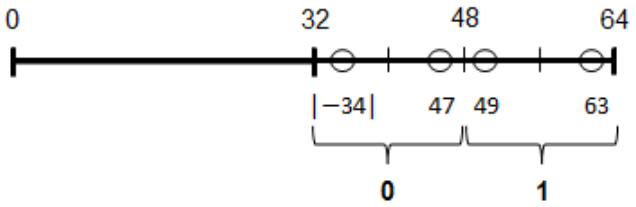


FIGURE II.21 – Quantification de la *liste* S pour $T_0 = 32$.

TABLE II.4 – Résultats de l'application l'algorithme EZW sur l'exemple de la figure II.20.

Itération	**Seuil**	*Liste C*	*Liste S*
1	$T_0 = 32$	PNZTPTTTTZTTTTTTTPTT	1010
2	$T_1 = 16$	ZTNPTTTTTTTT	100110
3	$T_2 = 8$	ZZZZZPPNPPNTTNNPTPTT NTTTTTTTTPTTTPTTTTT TTTTPTTTTTTTTTTTT	10011101111011011000
4	$T_3 = 4$	ZZZZZZZTZTZNZZZZPTTPT PPTPNPTNTTTTTPTPNP PPPTTTTTPTPTTTPNP	110111110110010000011 10110100010010101001
5	$T_4 = 2$	ZZZZZTZZZZZTPZZZTTPT TTTNPTPPTTPTTTNPPNTT TTPNNPTTPTTPPTTT	101111001101000101111 101011011001000000001 10110110011000111
6	$T_5 = 1$	ZZZTTZTTTZTTTTTNNTTT	

II.5.2 Algorithme de codage SPIHT

L'algorithme SPIHT (*Set Partitioning In Hierarchical Trees*) a été proposé par *Saïd* et *Pearlman* en 1996 [SAI96] pour la compression des images *avec* et *sans pertes*. Cet algorithme repose sur la gestion de trois listes : les coefficients *significatifs* (*LSP*), les coefficients *insignifiants* (*LIP*) et les ensembles *insignifiants* (*LIS*). La liste *LSP* est initialement vide, tandis que la liste *LIP* contient les racines de chaque arbre (coefficients de la bande basse fréquence) et la liste *LIS* contient l'ensemble des descendants de chaque arbre. Cette partition initiale est segmentée récursivement au moyen de deux principes [MOI07] :

- Si un ensemble de *descendants* d'un nœud est *significatif*, il est séparé en quatre coefficients fils directs de ce nœud, et l'ensemble des autres *descendants*. Les fils directs sont ajoutés à la *LIP* ou à la *LSP* en fonction de leur signifiance.
- Si au moins un élément de l'ensemble des autres *descendants* est *significatif*, cet ensemble est séparé en quatre ensembles *insignifiants* ajoutés à la *LIS*.

Le fait de traiter les coefficients par groupes de quatre permet d'effectuer un codage entropique efficace par la suite. L'algorithme SPIHT utilise les mêmes concepts de EZW : codage progressif par plan de bits et utilisation des dépendances hiérarchiques entre les coefficients des différents *sous-bandes*. Cependant, un nouveau protocole de

dépendance entre les coefficients est défini [BOU14][OUA08] . Les coefficients de la *sous-bande* de plus basse fréquence (*A3* en figure II.22) sont regroupés par quatre et pour chaque groupe la *descendance* est comme suit :

- Un des quatre coefficients (*A3* marqué par '×' en figure II.22) n'admet pas des *descendants*, alors que les trois autres (en gris en figure II.22) ont quatre *descendants* chacun.
- Pour les autres *sous-bandes*, comme dans le cas de EZW, chaque coefficient admet quatre *descendants*.

L'algorithme SPIHT permet de générer directement un flux binaire à la place des symboles de signifiance utilisés dans l'EZW. Cet algorithme permet d'avoir de performances meilleures que celles obtenues par l'EZW.

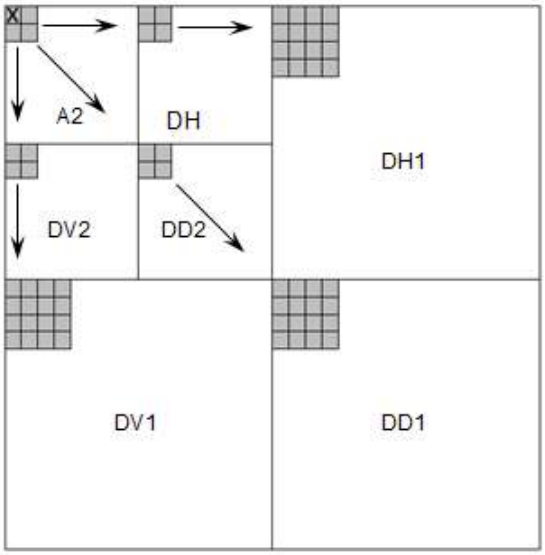


FIGURE II.22 – Exemples de descendances *parent-enfants* dans le cas SPIHT, le pixel désigné par (×) n'accepte pas de *descendants*.

II.6 Conclusion

Dans ce chapitre, on a dressé un état de l'art des techniques les plus connues dans le domaine de la compression d'images. Cela a permet d'appréhender les différentes méthodes compression *avec* et *sans pertes*. Les techniques *sans pertes* sont celles qui permettent d'avoir des données reconstruites exactes, mais elles proposent rarement des taux de compression importants. Sur ce point les techniques les plus efficaces sont celle *avec pertes*. Les approches utilisant les transformées sont actuellement les plus réputées. Différentes transformations telles que la KLT et la transformation en *ondelettes* ont

été détaillées. Nous verrons dans le chapitre suivant la *transformée en cosinus discrète* DCT en détail puisqu'elle est au cœur l'approche proposée.

Chapitre III

La DCT en compression d'images fixes

III.1 Introduction

La DCT (*Discrete Cosine Transform*) est une transformation très importante dans la compression des signaux. Elle existe en deux versions, une version *unidimensionnelle* traitant les signaux (1D) mais le plus souvent, elle est utilisée en version *bidimensionnelle* traitant les images pour diverses applications.

Ce chapitre sera consacré à la description de la DCT en exposant ses principales propriétés. Celle-ci est utilisée dans la majorité des standards de compressions d'images notamment par le format JPEG et de vidéos telles que MPEG, H264 et HEVC. Le célèbre format JPEG est largement utilisé pour les images pour Internet et constitue un standard pour la compression d'images. C'est pour cette raison que la dernière partie de ce chapitre sera dédiée à la description des différentes étapes de cet algorithme.

III.2 Définition

La DCT depuis son introduction en 1974 par *Ahmed* et al [AHM74] a fait l'objet de beaucoup d'études et d'applications telles que la compression, la détection de formes, le tatouage numérique (*watermarking*), l'estimation de mouvement et stéganographie. C'est une transformation mathématique qui transforme un ensemble de données (*image*, *son*, etc.) d'un domaine spatial en un spectre de fréquence et inversement.

Pour un signal d'entrée de taille *N*, le signal transformé est constitué d'un coefficient continu *DC* et *N-1* coefficients appelés coefficients *AC*. Le coefficient DC (*Direct Component*) représente correspondant la *valeur moyenne* et correspondant à la fréquence nulle. Tandis que les coefficients AC (*Alternative Component)* représentent les hautes fréquences (les détails) du signal d'entrée [BEA97][BAT12].

La forme *unidimensionnelle* DCT-1D est donnée par :

$$\begin{aligned} X(u) = \;& C(u) \sum_{n=0}^{N-1} x(n) \cdot cos\left[\frac{(2n+1)u\pi}{2N}\right] \\ & C(u) = \begin{cases} \sqrt{\frac{1}{N}} & \text{si} \quad u = 0 \\ \sqrt{\frac{2}{N}} & \text{sinon} \end{cases} \end{aligned} \tag{III.1}$$

où :

- n représente le temps discret,
- $x(n)$ représente le signal discret,
- $X(u)$ représente les coefficients de la DCT,
- N représentent la taille du signal.

Par contre, la DCT *bidimensionnelle* (DCT-2D) est un processus séparable qui peut être implémenté par deux DCT-1D, une dans la direction des *colonnes* et l'autre dans la direction des *lignes* (ou inversement). Pour un bloc de taille $N \times M$ pixels la DCT-2D est définie comme suit [KHA03][OUE12] :

$$\begin{aligned} F(u,v) = \;& \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} C(u) \cdot C(v) \cdot f(i,j) \cdot cos\left[\frac{(2j+1)u\pi}{2M}\right] \cdot cos\left[\frac{(2i+1)u\pi}{2N}\right] \\ & C(u) = \begin{cases} \sqrt{\frac{1}{N}} & \text{si } u = 0 \\ \sqrt{\frac{2}{N}} & \text{sinon} \end{cases} \end{aligned} \tag{III.2}$$

où :

- i, j représentent les indices des fréquences spatiales,
- $f(i,j)$ représente la valeur du pixel (i,j) de l'image originale,
- $F(u,v)$ représente les coefficients de la DCT,
- N et M représentent la taille du bloc image, le plus souvent on a $N = M$.

Dans le reste de ce chapitre on va prendre $N = M$, donc l'équation (III.2) sera remplacer par :

$$F(u,v) = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} C(u)\cdot C(v)\cdot f(i,j)\cdot cos\left[\frac{(2j+1)v\pi}{2N}\right]\cdot cos\left[\frac{(2i+1)u\pi}{2N}\right]$$
$$C(u) = \begin{cases} \sqrt{\frac{1}{N}} & \text{si } u = 0 \\ \sqrt{\frac{2}{N}} & \text{sinon} \end{cases} \quad \text{(III.3)}$$

Il est clair d'après les équations (III.1) et (III.3) que pour $n = 0$ $(i = 0, j = 0)$ le premier coefficient de la transformée est la *valeur moyenne* du signal ou de l'image respectivement.

$$X(0) = \frac{1}{\sqrt{N}}\sum_{i=0}^{N-1} x(n) \quad \text{(III.4)}$$

$$F(0,0) = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} f(i,j) \quad \text{(III.5)}$$

Le calcul des coefficients de la DCT peut se faire avec une combinaison linéaire d'un ensemble de fonctions $G_n(u)$ appelées *fonctions de base* définies par :

$$G_n(u) = C(u)\cdot cos\left[\frac{(2n+1)u\pi}{2N}\right] \quad \text{(III.6)}$$

Pour le cas de DCT-2D par :

$$G_{i,j}(u,v) = C(u)\cdot C(v)\cdot cos\left[\frac{(2i+1)u\pi}{2N}\right]\cdot cos\left[\frac{(2j+1)v\pi}{2N}\right] \quad \text{(III.7)}$$

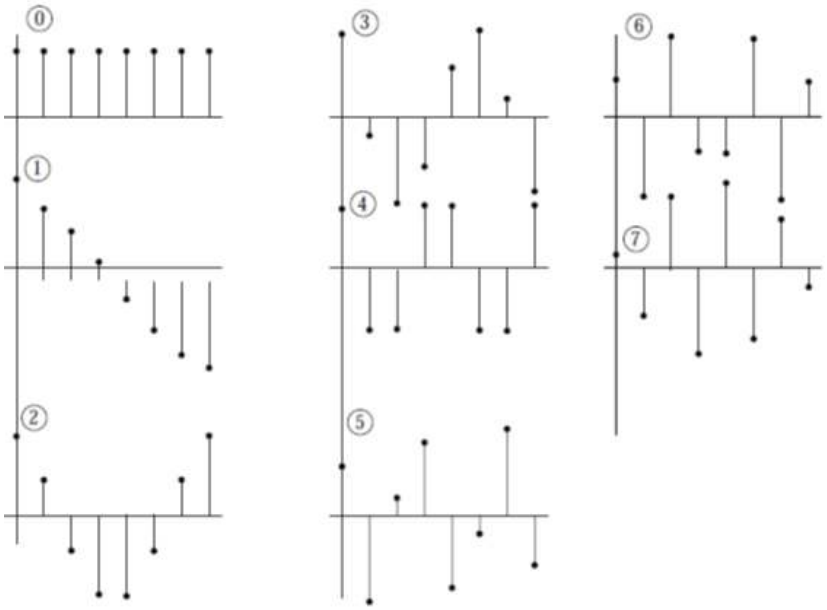
L'avantage d'une telle décomposition réside dans le fait que ces fonctions ; indépendantes du signal ou de l'image d'entrée ; peuvent être pré-calculées à l'avance. Ainsi, le calcul des coefficients DCT se réduit à la combinaison linéaire du le signal d'entrée et ces fonctions de base.Par conséquent, les équations (III.1) et (III.3) peuvent s'écrire sous la forme respectivement :

$$X(u) = \sum_{i=0}^{N-1} G_n(u)\cdot x(n) \quad \text{(III.8)}$$

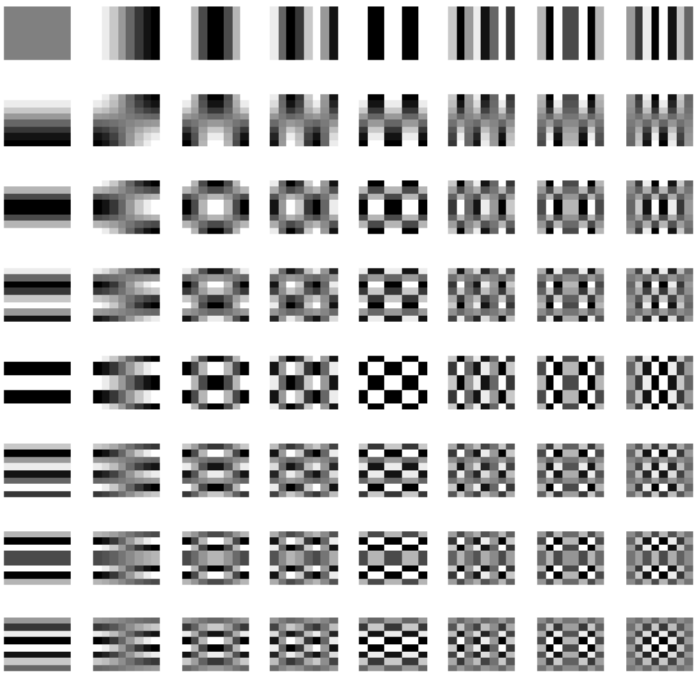
$$F(u,v) = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} G_{u,v}(i,j)\cdot f(i,j) \quad \text{(III.9)}$$

La figure III.1 représente les *fonctions de base* de la DCT-1D et DCT-2D. A noter que toutes ces fonctions sont *orthogonales*. Par conséquent, le produit scalaire point par point de deux fonctions différentes est nul. Ainsi, aucune de ces fonctions ne peut

être obtenue par une combinaison linéaire des autres fonctions.



(a)



(b)

FIGURE III.1 – Les *Fonctions de base* de la DCT pour *N=8* (a) : *unidimensionnelle* et (b) *bidimensionnelle*.

Il est possible à partir de ces coefficients DCT de retrouver le signal ou l'image d'origine en appliquant la transformation inverse définie par les formules (III.10) et (III.11) pour le cas 1D et 2D respectivement.

$$x(n) = \sum_{u=0}^{N-1} C(u) \cdot X(u) \cdot cos\left[\frac{(2n+1)u\pi}{2N}\right] \tag{III.10}$$

$$f(i,j) = \sum_{u=0}^{N-1}\sum_{v=0}^{N-1} C(u) \cdot C(v) \cdot F(u,v) \cdot cos\left[\frac{(2i+1)u\pi}{2N}\right] \cdot cos\left[\frac{(2j+1)v\pi}{2N}\right] \tag{III.11}$$

Pour le bidimensionnel, les coefficients DCT comportant l'information globale de l'image sont localisés dans les *basses* fréquences (en haut à gauche du bloc) et les coefficients situés en bas à droite représentent les *hautes* fréquences et représentent les détails de l'image. La figure III.2 illustre la répartition fréquentielle de la DCT-2D.

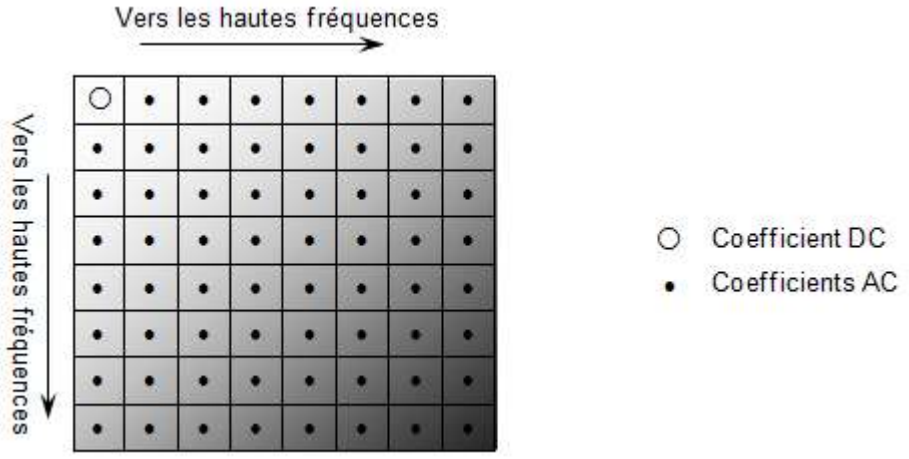


FIGURE III.2 – Répartition fréquentielle de la DCT-2D d'un bloc de 8×8.

III.3 Les types de la DCT

Il y a quatre chemins pour choisir n angles équitablement-espacés pour générer des vecteurs orthogonaux de cosinus. Cela produit quatre variantes de la *transformée discrète en cosinus* nommées DCT−I, DCT−II, DCT−III et DCT−IV. Les plus utilisées sont la DCT−II appelée simplement DCT et son inverse la DCT−III appelée IDCT [SAL04]. La définition des quatre types de DCT-1D est donnée par :

$$\begin{cases} DCT_I(k,j) & = \sqrt{\frac{2}{N}} C_k \cdot C_j \cdot cos\left[\frac{kj\pi}{N}\right] & k,j = 0,1,...N, \\ DCT_{II}(k,j) & = \sqrt{\frac{2}{N}} C_k \cdot cos\left[\frac{k(j+\frac{1}{2})\pi}{N}\right] & k,j = 0,1,...N-1, \\ DCT_{III}(k,j) & = \sqrt{\frac{2}{N}} C_j \cdot cos\left[\frac{(k+\frac{1}{2})j\pi}{N}\right] & k,j = 0,1,...N-1, \\ DCT_{IV}(k,j) & = \sqrt{\frac{2}{N}} \cdot cos\left[\frac{(k+\frac{1}{2})(j+\frac{1}{2})\pi}{N}\right] & k,j = 0,1,...N-1. \end{cases} \tag{III.12}$$

Avec le facteur d'échelle :

$$C_x = \begin{cases} \frac{1}{\sqrt{N}} & \text{si } x = 0 \quad \text{ou} \quad x = N \\ 1 & \text{sinon} \end{cases} \tag{III.13}$$

Il est à noter que la DCT−III est la transposée de la DCT−II et DCT−IV est la version décalée de DCT−I. Une autre remarque est que la DCT−I contient $N + 1$ vecteurs. Les angles Θ de la DCT−I et DCT−II pour N=*8* sont données par la matrice :

$$\Theta = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{\pi}{16} & \frac{3\pi}{16} & \frac{5\pi}{16} & \frac{7\pi}{16} & \frac{9\pi}{16} & \frac{11\pi}{16} & \frac{13\pi}{16} & \frac{15\pi}{16} \\ \frac{2\pi}{16} & \frac{6\pi}{16} & \frac{10\pi}{16} & \frac{14\pi}{16} & \frac{18\pi}{16} & \frac{22\pi}{16} & \frac{26\pi}{16} & \frac{30\pi}{16} \\ \frac{3\pi}{16} & \frac{9\pi}{16} & \frac{15\pi}{16} & \frac{21\pi}{16} & \frac{27\pi}{16} & \frac{33\pi}{16} & \frac{39\pi}{16} & \frac{45\pi}{16} \\ \frac{4\pi}{16} & \frac{12\pi}{16} & \frac{20\pi}{16} & \frac{28\pi}{16} & \frac{36\pi}{16} & \frac{44\pi}{16} & \frac{52\pi}{16} & \frac{60\pi}{16} \\ \frac{5\pi}{16} & \frac{15\pi}{16} & \frac{25\pi}{16} & \frac{35\pi}{16} & \frac{45\pi}{16} & \frac{55\pi}{16} & \frac{65\pi}{16} & \frac{75\pi}{16} \\ \frac{6\pi}{16} & \frac{18\pi}{16} & \frac{30\pi}{16} & \frac{42\pi}{16} & \frac{54\pi}{16} & \frac{66\pi}{16} & \frac{78\pi}{16} & \frac{90\pi}{16} \\ \frac{7\pi}{16} & \frac{21\pi}{16} & \frac{35\pi}{16} & \frac{49\pi}{16} & \frac{63\pi}{16} & \frac{77\pi}{16} & \frac{91\pi}{16} & \frac{105\pi}{16} \end{bmatrix} \tag{III.14}$$

III.4 Propriétés de la DCT

Dans la partie précédente, nous avons développé la base mathématique de la DCT. Cette section expose avec des exemples quelques propriétés de la DCT qui sont particulièrement importantes à la compression d'image. L'avantage majeur de la DCT, et c'est pourquoi elle est utilisée dans de nombreux algorithmes de compression, est qu'elle permet d'avoir une représentation fréquentielle du signal à l'aide de coefficients réels (contrairement à la *transformée de Fourier* qui utilise des coefficients complexes). Cela permet de privilégier certaines fréquences (généralement les basses fréquences) via l'utilisation de tables de quantification adaptées [BAT12] ou d'un processus de seuillage.

Malgré ses caractéristiques, cette transformée a su montrer ses faiblesses au fil du temps. Le principal reproche va aux artefacts ou effet de blocs générés lors d'une compression *avec pertes*. De plus, les fonctions sinusoïdales étant à valeurs numériques réelles, le calcul de la DCT génère des pertes d'informations du fait de les arrondir à des entiers [TAQ11].

III.4.1 Compactage d'énergie

Le *compactage* d'énergie des données d'entrée en un ensemble réduit de coefficients caractérise l'efficacité d'une transformation. Ceci permet d'écarter les coefficients de faibles amplitudes sans trop altérer la qualité de l'image reconstruite. Pour illustrer le *compactage* d'énergie pour la DCT, appliquons la DCT sur l'image typique *Lena*

(figure III.3.a). La figure III.3.b montre le très bon compactage d'énergie dans la région de basse fréquence (coins supérieur à gauche) de l'image transformée (coefficients DCT). La figure III.3.c, générée par le code de l'*Annexe A*, illustre l'image reconstruite avec seulement 5% de coefficients DCT. Ce *compactage* permet à la DCT d'être le processus fondamental du standard de compression des images JPEG.

(a) (b) (c)

FIGURE III.3 – Compactage d'énergie de la DCT, (a) image *Lena*, (b) coefficients DCT et (c) image *Lena* reconstitué avec 5% de coefficients DCT.

III.4.2 Décorrélation

L'avantage majeur d'une transformation d'images est l'élimination de la redondance entre les pixels voisins. Ceci engendre des coefficients de transformation *non corrélés* qui peuvent être codés indépendamment sans perdre en termes d'efficacité de la compression [KHA03] [ARO09][BRA11].

Pour illustrer cette propriété, prenons l'exemple de la figure III.3.a. Même avec un grand degré de détail dans beaucoup de régions de cette image, la valeur d'un pixel tend à être semblable à ses pixels voisins. La *corrélation* des points de l'image avant l'application de la DCT est claire dans figure III.4.a sur le tracé des valeurs du niveau de gris des paires de pixels adjacents. Ce tracé, générée par le code de l'*Annexe B*, correspond aux points $P(x, y)$ tel que x présente le niveau de gris d'un pixel et l'ordonnée y présente le niveau de gris de son voisin droit. La forte corrélation se voit par un nuage de points dans la direction de la droite $y = x$. La *corrélation* des coefficients DCT est très faible selon la figure III.4.b.

III.4.3 Séparabilité

La DCT-2D appartient à la classe des transformations à noyaux *séparables*. Ce qui permet de calculer la DCT-2D par une double application de la DCT-1D. D'après l'équation (III.3) la DCT-2D peut être écrite comme suit :

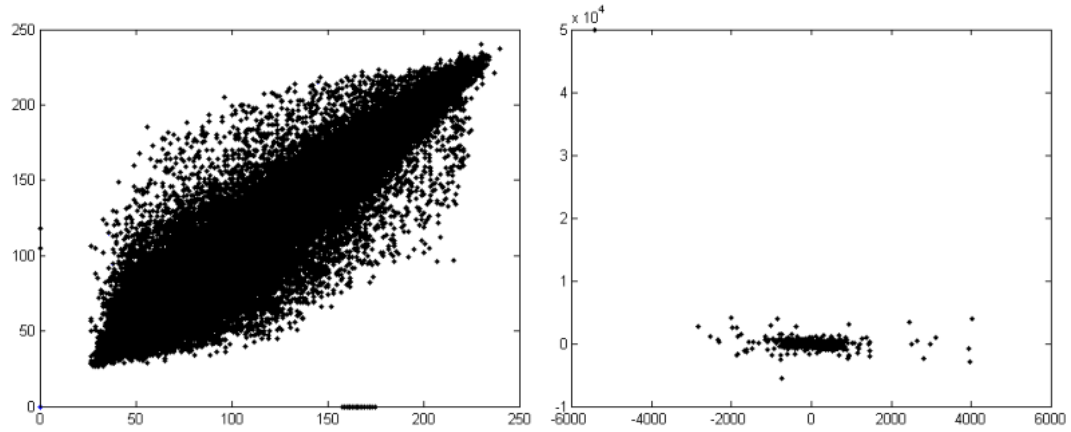


FIGURE III.4 – Les valeurs du niveau de gris des paires de pixels adjacents : (a) de l'image *Lena* et (b) des coefficients DCT.

$$F(u,v) = C(u) \cdot \sum_{i=0}^{N-1} cos\left[\frac{(2i+1)u\pi}{2N}\right] \cdot C(v) \cdot \sum_{j=0}^{N-1} f(i,j) \cdot cos\left[\frac{(2j+1)v\pi}{2N}\right] \quad \text{(III.15)}$$

De cette dernière équation, on voit qu'on peut calculer le coefficient $F(u,v)$ en calculant la sommation selon i (*linge* par *ligne*) ou calculant la sommation selon j (*colonne* par *colonne*). Cette propriété est appelée la *séparabilité* de la transformation. De ce fait, la transformation DCT-2D peut être calculée par deux transformations DCT-1D successives [ARO09]. Cette idée est illustrée graphiquement sur la figure III.5.

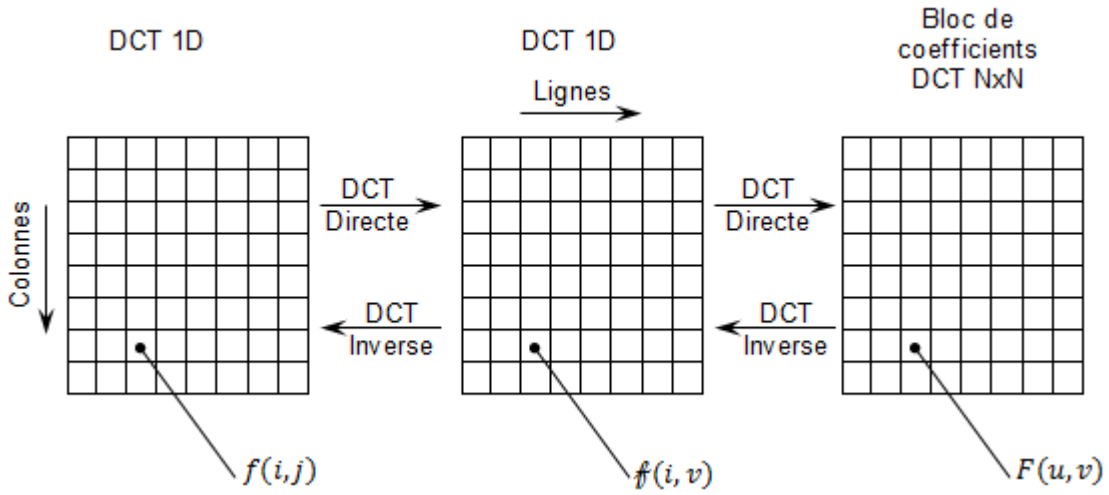


FIGURE III.5 – Calcul de la DCT-2D en utilisant la propriété de *séparabilité*.

III.4.4 Symétrie

L'équation (III.11) peut être vue comme un produit d'une matrice par les *fonctions de base* de la DCT. Cela nécessite les mêmes opérations pour les *lignes* que pour les *colonnes*, une telle transformation s'appelle une transformation *symétrique*. Une transformation *séparable* et *symétrique* peut être exprimée sous la forme :

$$F = T_N \times f \times T_N^T \tag{III.16}$$

$$T_N(i,j) = \begin{cases} \sqrt{\frac{1}{N}} & \text{si} \quad i = 0 \\ \sqrt{\frac{2}{N}} \cdot cos\left[\frac{(2j+1)v\pi}{2N}\right] & \text{sinon} \end{cases} \tag{III.17}$$

où T_N^T est la matrice transposée de T_N.

En traitement d'images, la matrice T_N est appelée *noyau* de la transformation. On peut remarquer d'après l'équation (III.17) que la matrice T_N peut être *pré-calculée* ce qui permet une amélioration en temps de calcul.

III.4.5 Orthogonalité

De l'équation (III.17) on remarque que la matrice T_N de la DCT est orthogonale :

$$T_N \times T_N^T = I_N \tag{III.18}$$

où I_N représente la matrice d'identité de taille $N \times N$.
Ce qui implique que :

$$T_N^{-1} = T_N^T \tag{III.19}$$

Par conséquent, et en plus de la caractéristique de *décorrélation*, cette propriété fournie une certaine réduction dans la complexité de pré-calcul de T_N [KHA03]. En exploitant l'*orthogonalité* de T_N, l'inverse de la transformation donnée par l'équation (III.11) est calculé par :

$$f = T_N^T \times F \times T_N \tag{III.20}$$

III.5 Algorithmes rapides de la DCT

Le calcul direct de la DCT nécessite un nombre important d'opérations *arithmétiques* $(N \times N - 1)$ *additions* et N^2 *multiplications* [AHM74]. Afin de proposer une architecture efficace, plusieurs algorithmes rapides ont été proposés dans la littérature,

tels que les algorithmes de *Chen*[CHE77], *Wang*[WAN84], *Lee*[LEE84], *Vetterli*[VIT84], *Suehiro*[SUE86] et *Loeffler*[LEO89].

En effet, le calcul de la DCT peut se faire de plusieurs méthodes. *Hou* [HOU87] a développé une méthode basée sur la récursivité pour la factorisation de la matrice DCT. La récursivité est utilisée ici pour calculer les coefficients DCT de taille N à partir de DCT de taille $N/2$.

Pour un vecteur de *8* points ($N=8$), cet algorithme permet de calculer les coefficients de la DCT en *12 multiplications* et *29 additions*. *Vitterli* [VIT84] a proposé une méthode de calcul indirect en utilisant la *Transformée de Fourier Rapide* FFT et la *Transformé Walsh Hadamard* (WHT) pour le calcul des coefficients DCT. Ces derniers peuvent être trouvés par une méthode basée sur la convolution. Celle-ci s'appuie sur le principe de la reformulation des nombres premiers en un certain nombre de cycliques. Dans cette optique, Chen [CHE05] a développé une architecture de calcul de la DCT à base deux convolutions cycliques similaires.

Une autre méthode pour le calcul des coefficients DCT a été adoptée par un grand nombre de chercheurs est celle utilisant la factorisation directe qui exploite la propriété de symétrie de la matrice de transformation. L'avantage principal de cette méthode réside dans l'utilisation d'un nombre réduit d'*additions* ou de *multiplications*.

En effet, Wang [WAN84] a proposé un algorithme utilisant en *13 multiplications* et *29 additions*. En plus des coefficients DCT, il permet aussi de calculer la *Transformée en Sinus Discrète* (DST), la *transformée en ondelettes discrète* (DWT) et la *Transformée de Fourier discrète (DFT)*. Sur la même démarche, *Suehiro* [SUE86] a développé un algorithme optimisé qui a permet de réduire le nombre de *multiplications* nécessaire pour le calcul des coefficients DCT ainsi que pour les autres transformées déduites.

Les algorithmes de *Chen* [CHE77] et de *Leoffler* [LEO89] appartenant à cette méthode de calcul sont très utilisés dans la littérature. La table III.1 présente une comparaison entre les différents algorithmes utilisés pour le calcul des coefficients DCT selon le nombre de *multiplications* et *additions* [SAL04][OUE12].

III.6 Le standard JPEG

Le standard JPEG (*Joint Photographic Experts Group*) est une norme définie en 1993 par les comités ISO et CCITT, pour la compression des images fixes couleurs ou

TABLE III.1 – Comparaison entre les différents algorithmes selon le nombre d'opérations.

Auteur	*Chen* [CHE77]	*Wang* [WAN84]	*Lee* [LEE84]	*Vetterli* [VET84]	*Suehiro* [SUE86]	*Hou* [HOU87]	*Loeffler* [LEO89]
Multiplication	16	13	12	12	12	12	11
Addition	26	29	29	29	29	29	29

en niveaux de gris. La norme JPEG est actuellement très utilisée pour coder les images numériques que l'on trouve dans notre vie quotidienne (mobile, Internet, appareils photos numériques, etc.).

Cet algorithme spécifie deux modes de compression, notamment la compression *sans pertes* et la compression *avec pertes* [BAT12] :

- Un mode de compression *sans pertes* basé sur la prédiction du pixel courant à partir de son voisinage (*Lossless* JPEG).
- Un mode de compression *avec pertes* basé sur la DCT et permettant de choisir la qualité de l'image reconstruite via l'utilisation d'un paramètre dédié.

III.6.1 La compression sans pertes (JPEG-LS)

Le mode de fonctionnement *sans pertes* (JPEG-LS), est en effet une simple méthode basée sur le codage *prédictif* de la méthode DPCM en utilisant les valeurs des pixels voisins. Le schéma d'un tel codeur est donné par la figure III.6. Le *prédicteur* est de type simple DPCM où chaque pixel de chaque composante couleur est codé de manière différentielle. La *prédiction* d'un pixel d'entrée x est effectuée à partir de ses trois pixels voisins de la même composante couleur. Ce *prédicteur* ; connu sous le nom *Median Edge Detector* (MED) ; est illustré sur la figure III.7.

La *prédiction* est ensuite soustraite de la valeur réelle du pixel à la position x, et la différence est codée en entropie soit par la technique *Huffman* soit par le codage *arithmétique*. Les spécifications du tableau d'entropie (figure III.6) déterminent les caractéristiques de la méthode choisie pour le codage entropique [EMR12].

III.6.2 La compression avec pertes du JPEG

Le principe de l'algorithme JPEG *avec pertes* pour une image en niveaux de gris est simple. L'image est décomposée séquentiellement en blocs de 8×8 subissant le même traitement. Une transformée en DCT-2D est appliquée à chaque bloc. Les coefficients DCT sont ensuite quantifiés uniformément en association avec une table de 64 éléments

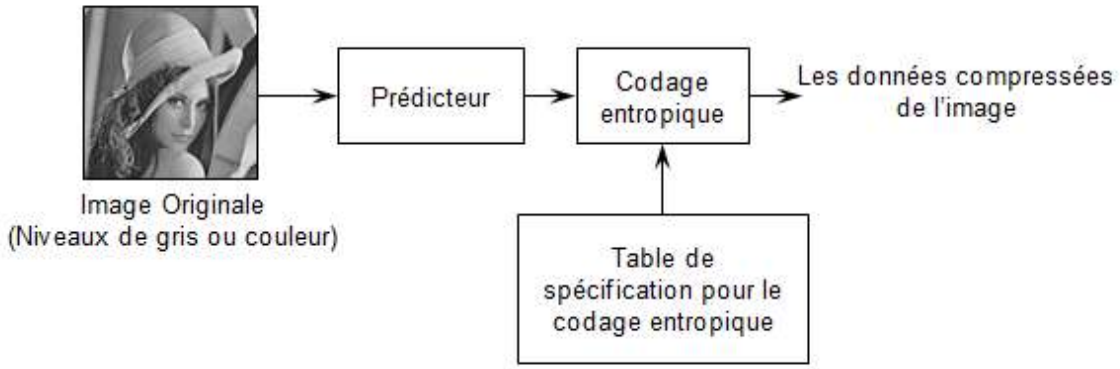


FIGURE III.6 – Le schéma de bloc du codeur JPEG en mode *sans pertes* (codeur JPEG-LS).

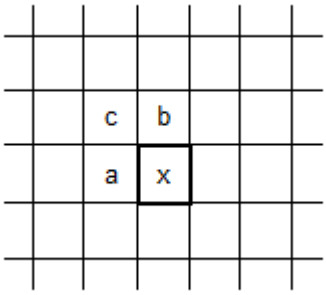


FIGURE III.7 – Le voisinage de la prédiction à trois échantillons du *prédicteur* (MED).

définissant les pas de quantification. Cette table permet de choisir un pas de quantification important pour certaines composantes jugées peu significatives visuellement.

La compression d'une image couleur peut être vue comme la compression de plusieurs images en niveaux de gris. Deux variantes existent pour la compression d'une image en couleurs [ZIT13] :

- Composante par composante (la compression de la deuxième composante débute lorsque celle de la première est terminée.)
- Par bloc de 8×8 pixels en alternant les composantes de l'image (premier bloc de la première composante, puis premier bloc de la deuxième composante, etc.).

L'algorithme JPEG *avec pertes* fonctionne conformément au schéma de la figure III.8. L'image originale subit une transformation colorimétrique suivie d'un sous-échantillonnage des composantes de *chrominance*. L'image résultante est découpée en blocs de 8×8. Une transformation DCT *bidimensionnelle* est appliquée sur chaque bloc, suivie par l'étape de quantification. Enfin, un codage entropique *sans pertes* est appliqué sur l'ensemble des coefficients quantifiés.

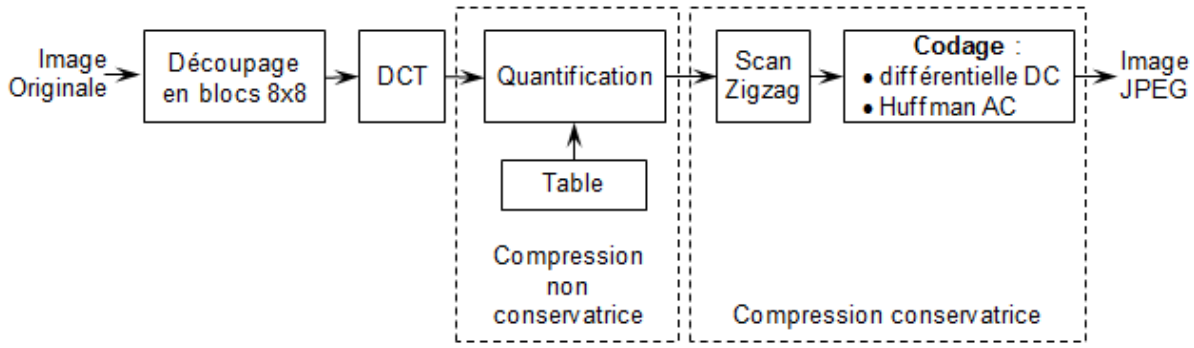


FIGURE III.8 – Chaîne de codage de l'algorithme JPEG.

III.6.3 Prétraitement de l'image

La phase de *prétraitement* de l'image est une phase optionnelle ne faisant pas partie de la norme JPEG et qui est effectuée seulement pour les images couleurs. Elle permet la conversion d'une image dans l'espace colorimétrique adéquat et une première compression par un sous-échantillonnage sur les composantes de *chrominances*.

III.6.3.1 Changement de l'espace colorimétrique

Classiquement les images à compresser sont codées dans l'espace colorimétrique RGB qui contient des redondances *inter plans* importantes. L'œil étant particulièrement sensible aux faibles variations de *luminance*, l'espace de couleur YCbCr est particulièrement adapté à la compression. Une transformation de l'espace RGB vers l'espace de couleur YCbCr est effectuée pour séparer la composante *luminance* (Y) et les composantes *chrominances* (Cb pour la *chrominance bleue*, et Cr pour la *chrominance rouge*).

Il est possible de dégrader la *chrominance* d'une image tout en gardant une bonne qualité. Cela permet de réduire la taille de l'image [ZIT13]. L'espace de couleur YCbCr est choisi essentiellement car :

- Il garantit une décorrélation de l'information sur les composantes Cb et Cr vis-à-vis de la composante Y.

- l'œil humain étant moins sensible aux changements de *chrominance*, les deux composantes Cb et Cr sont sous-échantillonnées pour réduire la quantité d'informations sans pour autant altérer la qualité globale perçue de l'image.

Le passage de l'espace de couleur RGB vers YCbCr est effectué selon la formule :

$$\begin{cases} Y = & 0.299 \cdot R & +0.587 \cdot G & +0.114 \cdot B \\ C_b = & -0.169 \cdot R & -0.331 \cdot G & +0.500 \cdot B \\ C_r = & 0.500 \cdot R & -0.418 \cdot G & -0.081 \cdot B \end{cases} \quad \text{(III.21)}$$

Afin de représenter *Cb* et *Cr* par des nombres entiers de 8 bits non signés, ils sont décalés en ajoutant 128 à chaque échantillon suivi d'une saturation de la valeur dans la gamme [0,255]. Par conséquent, la transformation ci-dessus peut être exprimée comme :

$$\begin{cases} Y = & 0.299 \cdot R & +0.587 \cdot G & +0.114 \cdot B & \\ C_b = & -0.169 \cdot R & -0.331 \cdot G & +0.500 \cdot B & +128 \\ C_r = & 0.500 \cdot R & -0.418 \cdot G & -0.081 \cdot B & +128 \end{cases} \quad \text{(III.22)}$$

Lors du passage d'un espace colorimétrique à un autre, les seules pertes possibles sont donc celles relatives aux arrondis. Différents espaces colorimétriques peuvent être utilisés tels que YCbCr, YUV, CIELUV, CIELAB, etc. L'espace colorimétrique utilisé est spécifié dans l'entête de l'image compressée.

III.6.3.2 Le sous-échantillonnage

Conformément à l'assertion (1) de la *section* II.4 du *chapitre* II, l'œil étant moins sensible aux changements de *chrominance*. On va pouvoir donc *sous-échantillonner* les deux composantes *Cb* et *Cr* pour réduire de façon conséquente le volume d'informations à compresser sans trop toucher à l'aspect global de l'image. La figure III.9 décrit les différents modes du *sous-échantillonnage* des composantes de *chrominance* proposés par JPEG suivant les *lignes* et/ou les *colonnes*.

- 4 :4 :4 : Pas de *sous-échantillonnage* des *chrominances*. Ce format est utilisé lorsque toute la qualité de l'image doit être gardée,
- 4 :2 :2 : On ne garde qu'une ligne sur deux pour les composantes couleurs. Seulement la moitié de l'information couleur est gardée ce qui réduit la taille de l'image résultante à 2/3 $(\frac{1}{3} + \frac{2}{4} \cdot \frac{1}{3} + \frac{2}{4} \cdot \frac{1}{3})$ de l'image originale,
- 4 :1 :1 : Dans ce mode la taille de l'image originale est réduite à la moitié $(\frac{1}{3} + \frac{1}{4} \cdot \frac{1}{3} + \frac{1}{4} \cdot \frac{1}{3})$,
- 4 :2 :0 : Ce mode réduit la taille de l'image résultante à la moitié $(\frac{1}{3} + \frac{2}{4} \cdot \frac{1}{3})$ de l'image originale,

Il est à noter que la composante de *luminance* n'est jamais *sous-échantillonnée*.

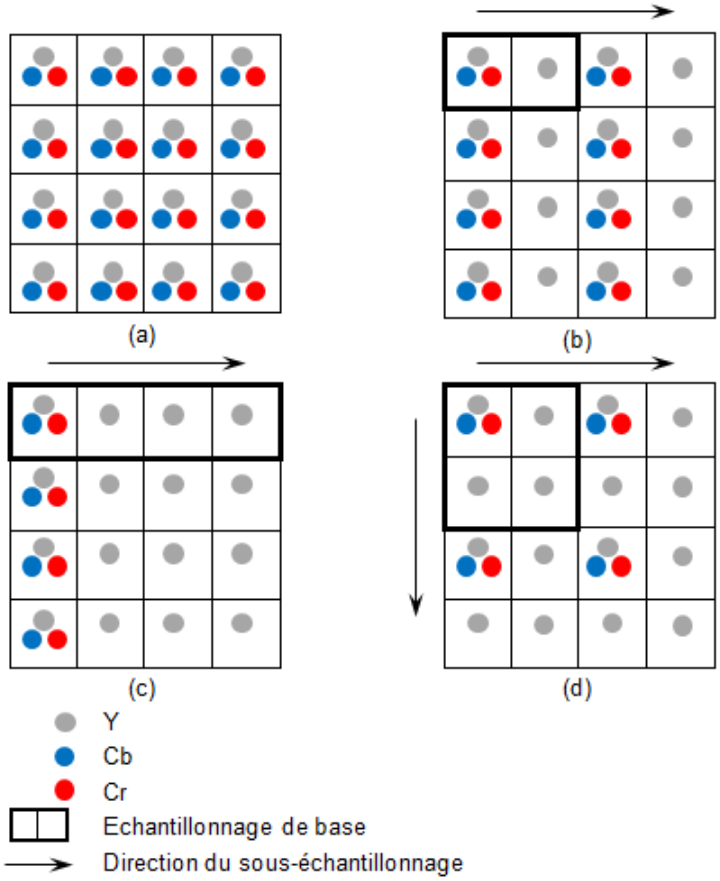


FIGURE III.9 – Représentation des modèles du sous-échantillonnage dans l'espace YCbCr : (a) 4 :4 :4 pas de *sous-échantillonnage*, (b) 4 :2 :2, (c) 4 :1 :1 et (d) 4 :2 :0.

III.6.3.3 Découpage en blocs

Les pixels de chaque plan de l'espace YCbCr sont groupés en blocs de 8×8. Dans le cas où la *largeur* ou la *longueur* de l'image n'est pas multiple de 8, la dernière *ligne* ou *colonne* de l'image est dupliquée autant de fois que nécessaire jusqu'à atteindre un multiple de 8. Les blocs seront traités de gauche à droite et de haut en bas [ZIT13]. Le *découpage* a plusieurs raisons, dont on peut citer :

- L'application de la DCT à l'image entière produit une meilleure compression mais ralentit énormément la compression. Par contre l'application de la DCT sur de petits blocs donne des taux de compression réduits mais la compression est plus rapide,

- D'une façon générale, les corrélations entre les pixels d'une image sont de court terme. La corrélation entre un pixel et ses proches voisins (*horizontaux* et *verticaux*) est forte, tandis qu'elle s'affaiblit pour les voisins lointains.

III.6.4 Application de la DCT bidimensionnelle

L'étape qui suit est la transformation DCT (transposition *amplitude/fréquence*). Elle consiste à passer du domaine spatial au domaine fréquentiel. Cette transformation est appliquée sur chaque bloc de 8×8 suivant l'équation (III.3) par :

$$\begin{aligned} F(u,v) = \;& C(u) \cdot C(v) \cdot \sum_{i=0}^{7} \sum_{j=0}^{7} f(i,j) \cdot cos\left[\frac{(2j+1)v\pi}{16}\right] \cdot cos\left[\frac{(2i+1)u\pi}{16}\right] \\ & C(u) = \begin{cases} \frac{1}{2\sqrt{2}} & \text{si} \quad u = 0 \\ \frac{1}{2} & \text{sinon} \end{cases} \end{aligned} \tag{III.23}$$

D'après l'équation (III.17), la matrice de transformation de la DCT T_N pour $N = 8$ est donnée par :

$$T_8 = \frac{1}{2} \cdot \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ cos\frac{\pi}{16} & cos\frac{3\pi}{16} & cos\frac{5\pi}{16} & cos\frac{7\pi}{16} & cos\frac{9\pi}{16} & cos\frac{11\pi}{16} & cos\frac{13\pi}{16} & cos\frac{15\pi}{16} \\ cos\frac{2\pi}{16} & cos\frac{6\pi}{16} & cos\frac{10\pi}{16} & cos\frac{14\pi}{16} & cos\frac{18\pi}{16} & cos\frac{22\pi}{16} & cos\frac{26\pi}{16} & cos\frac{30\pi}{16} \\ cos\frac{3\pi}{16} & cos\frac{9\pi}{16} & cos\frac{15\pi}{16} & cos\frac{21\pi}{16} & cos\frac{27\pi}{16} & cos\frac{33\pi}{16} & cos\frac{39\pi}{16} & cos\frac{45\pi}{16} \\ cos\frac{4\pi}{16} & cos\frac{12\pi}{16} & cos\frac{20\pi}{16} & cos\frac{28\pi}{16} & cos\frac{36\pi}{16} & cos\frac{44\pi}{16} & cos\frac{52\pi}{16} & cos\frac{60\pi}{16} \\ cos\frac{5\pi}{16} & cos\frac{15\pi}{16} & cos\frac{25\pi}{16} & cos\frac{35\pi}{16} & cos\frac{45\pi}{16} & cos\frac{55\pi}{16} & cos\frac{65\pi}{16} & cos\frac{75\pi}{16} \\ cos\frac{6\pi}{16} & cos\frac{18\pi}{16} & cos\frac{30\pi}{16} & cos\frac{42\pi}{16} & cos\frac{54\pi}{16} & cos\frac{66\pi}{16} & cos\frac{78\pi}{16} & cos\frac{90\pi}{16} \\ cos\frac{7\pi}{16} & cos\frac{21\pi}{16} & cos\frac{35\pi}{16} & cos\frac{49\pi}{16} & cos\frac{63\pi}{16} & cos\frac{77\pi}{16} & cos\frac{91\pi}{16} & cos\frac{105\pi}{16} \end{bmatrix} \tag{III.24}$$

III.6.5 Quantification

Une fois la transformation DCT est effectuée, on peut procéder à l'étape de *quantification*. Conformément à l'assertion (2) de la *section* II.4 du *chapitre* précédent, la répartition statistique de l'amplitude des fréquences spatiales $F(u,v)$ dépend fortement de son rang (u,v) et elle décroît rapidement quand (u,v) s'éloigne de point (0,0). L'étape de *quantification* va permettre d'ignorer les hautes fréquences (*détails*) afin d'atteindre des taux de compression plus élevés sans une importante dégradation de la qualité de reconstruction.

L'étape de *quantification* se base essentiellement sur des tables de quantification définies pour la *luminance* B_Y et la *chrominance* B_{CrCb} par :

$$B_Y = \begin{vmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{vmatrix} \tag{III.25}$$

$$B_{CrCb} = \begin{vmatrix} 17 & 18 & 24 & 47 & 99 & 99 & 99 & 99 \\ 18 & 21 & 26 & 66 & 99 & 99 & 99 & 99 \\ 24 & 26 & 56 & 99 & 99 & 99 & 99 & 99 \\ 47 & 66 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \end{vmatrix} \tag{III.26}$$

La *quantification* utilise un facteur de quantification Q et un facteur de qualité q défini par l'utilisateur. Ce dernier aura des valeurs entre *1* et *100* selon la qualité désirée, la valeur *100* étant la qualité d'image maximale. Tandis que le facteur de quantification Q sera calculé par l'équation (III.27) à partir des deux matrices de quantification (III.25) et (III.26) [BAT12] :

$$Q(i,j) = \begin{cases} ((\frac{5000}{q}) \times B(i,j) + 50)/100 & \text{si} \quad q < 50 \\ ((200 - 2q) \times B(i,j) + 50)/100 & \text{sinon} \end{cases} \tag{III.27}$$

On remarque clairement que le facteur de quantification associé aux hautes fréquences est plus important permettant ainsi de négliger les coefficients DCT correspondants. La valeur quantifiée Fq du coefficient F est calculée par :

$$F_q = \lceil \frac{F}{Q} \rceil \tag{III.28}$$

où $\lceil \cdots \rceil$ représente un arrondit à l'entier supérieur.

A la fin de cette étape, la matrice des coefficients DCT quantifiés contiendra très peu de coefficients *non nuls* essentiellement concentrés dans la région gauche supérieure. La phase suivante tentera de compresser ces coefficients *non nuls* par un codage *entropique* pour avoir plus d'efficacité en ce qui concerne le taux de compression.

III.6.6 Codage entropique

Dans cette étape, on parcourt chaque bloc afin de coder les 64 coefficients qui le constitue. D'après les tables de quantification précédentes, il est facile de voir que plus l'on va aller dans les hautes fréquences (coin inférieur droit du bloc), plus on a de chances de trouver des valeurs nulles. Afin de minimiser le code globale et ainsi réaliser une compression plus efficace, on doit utiliser des parcours pour constituer des suites de valeurs *nulles* les plus longues possibles.

Pour cela, la meilleure solution consiste à utiliser un parcours *Zigzag*. Celui-ci lit les différentes fréquences en partant du coefficient DC (emplacement $(0,0)$) jusqu'aux fréquences les plus hautes orthogonalement à la *diagonale*, maximisant ainsi le nombre de valeurs *nulles* consécutives en fin de parcours [BAT12]. Une fois la procédure de parcours terminée, on dispose d'une suite de 64 coefficients (1 coefficient DC et 63 coefficients AC).

Ces coefficients sont codés séparément, en effet, la composante DC d'un bloc 8×8 étant fortement corrélée avec la composante DC du bloc précédent, ces coefficients vont être codés différemment des coefficients AC, en utilisant un codage *différentiel* de type DPCM. C'est la différence E_k qui sera codée ; et non pas le coefficient lui-même ; selon l'équation [ZIT13] :

$$E_k = DC_k - DC_{k-1} \tag{III.29}$$

La figure III.10 représente le codage des coefficients DC. Par contre, les 63 coefficients AC commençant à partir du coefficient $AC(0,1)$ sont codés par une combinaison du codage RLE et du codage *Huffman* ou *arithmétique* après un scan *Zigzag*. L'idée est que la séquence des coefficients AC contient juste quelques nombres *non nuls*, avec une séquence de *zéros* entre eux.

L'algorithme JPEG utilise des tables de correspondances entre les différentes valeurs de coefficients et le la taille de la séquence de *zéros* pour coder efficacement un bloc (table III.2). Seuls les coefficients *non nuls* sont codés, mais d'une façon qui prend en compte le nombre de zéros qui les précèdent.

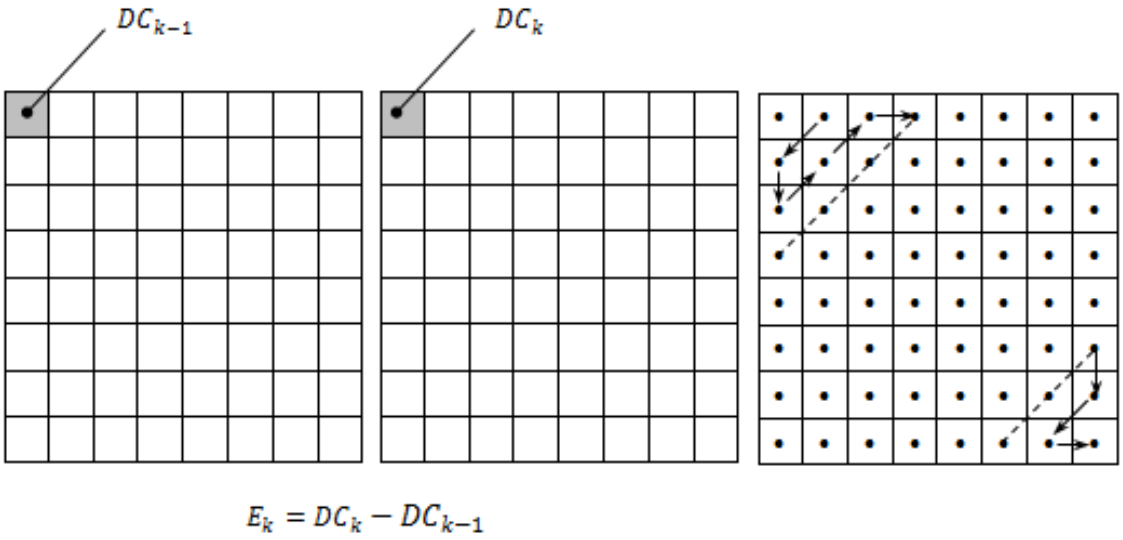


FIGURE III.10 – Préparation des coefficients quantifiés pour le codage entropique. (a) : Codage différentiel des coefficients *DC* et (b) : Lecture en *Zigzag* des coefficients *AC*.

Un coefficient *AC* non nul est caractérisé par :

1. Le nombre des zéros qui le séparent de son prédécesseur *non-nul*,
2. Sa catégorie i qui donne le nombre de bits nécessaires pour le codage du coefficient,
3. Son numéro dans la catégorie (ligne de la table III.2) (les numéros commencent par zéro).

TABLE III.2 – Table de codage des valeurs de coefficients de la norme JPEG.

Catégorie	**Valeur du coefficient**	
1	-1	1
2	-3, -2	2, 3
3	-7, -6, -5, -4	4, 5, 6, 7
4	-15, -14, ..., -9, -8	8, 9, ..., 14, 15
5	-31, -30, ..., -17, -16	16, 17, ..., 30, 31
6	-63, -62, ..., -33, -32	32, 33, ..., 62, 63
7	-127, -126, ..., -65, -64	64, 65, ..., 126, 127
8	-255, -254, ..., -129, -128	128, 129, ..., 254, 255
9	-511, -510, ..., -257, -256	256, 257, ..., 510, 511
10	-1023, -1022, ..., -513, -512	512, 513, ..., 1022, 1023

Parce que les séquences de zéros peuvent être longues, leurs tailles sont codées par des nombres binaires *hexadécimaux*. Si le nombre la séquence est d'une taille supérieure

à 16, on rajoute un symbole de *Zero Run List* pour chaque bloc de 16 zéros et on compte la suite de zéros.

Pour illustrer cela d'une façon claire prenant l'exemple suivant :
Soit la suite des 63 coefficients AC :
0 -46 0 0 0 0 0 0 -4 0 0 0 0 0 0 0 0 0 0 -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0.

Le code de ce bloc est donnée à la table III.3.

TABLE III.3 – Codage des coefficients AC.

Coefficient	**Code**	**Remarque**
-46	(1/6) [17]	1 zéro qui le précède, 6 bits pour coder 46, 17 l'indice ou le numéro de -46 dans la ligne 6 de la table III.2.
-4	(6/3) [3]	6 zéros qui le précèdent, 3 bits pour coder 4, 3 l'indice ou le numéro de -4 dans la ligne 3 de la table III.2.
-1	(A/1) [0]	A 10 zéros qui le précèdent, 1 bit pour coder 1, 0 l'indice ou le numéro de -1 dans la ligne 1 de la table III.2.
00...0	(EOB)	Code de la fin du bloc.

III.7 Conclusion

Dans ce chapitre, nous avons présenté dans un premier temps la transformation DCT et ses propriétés. On a vu que cette transformation exploite les redondances inter-pixels de l'image pour produire des coefficients non corrélés. Elle permet aussi de subdiviser l'image en basses et hautes fréquences ce qui permet de compacter l'énergie contenue dans l'image dans la partie des basses fréquences. La dernière partie du chapitre a été consacrée à l'étude des différentes parties de la norme JPEG. Ces parties seront présentent d'une façon ou autre dans n'importe quel algorithme de compression. Le chapitre suivant fera l'objet d'une étude détaillée de l'approche proposée ainsi qu'aux tests de simulation.

Chapitre IV

Technique élaborée pour la compression d'images

IV.1 Introduction

Dans ce chapitre, nous allons présenter la méthode mise au point pour la compression *avec pertes* des images naturelles (*non synthétiques*) fixes couleurs. Ce chapitre est composé de deux parties principales.

Dans la première partie nous détaillerons le fonctionnement de l'algorithme proposé. Tandis que dans la seconde partie nous présenterons, analyserons et commenterons les résultats obtenus par notre approche. Cette dernière a fait l'objet en 2016 d'une publication (*Messaoudi et al.* [MES16]) dans le journal *ELECTRONICS LETTERS*.

IV.2 Critères d'évaluation de la compression

Lorsqu'on utilise des méthodes de compression *non réversibles*, trois critères sont généralement retenus pour l'évaluation de la compression : la *distorsion*, le *taux de compression* et la *charge de calcul* impliqué par la méthode. La pertinence des méthodes de compression *avec pertes* ne dépend pas seulement du taux de compression obtenu, mais aussi de la perte de qualité entre l'image originale et l'image restituée.

D'une façon générale, il existe deux types de critères permettant d'évaluer cette qualité : les critères *subjectifs* et les critères *objectifs*. Le critère *subjectif* est tout simplement le critère visuel. L'œil est un outil essentiel pour apprécier la qualité, la netteté,

les contours et le contenu d'une image. L'évaluation de la qualité visuelle d'une image n'est, pas évidente [MAN74]. Dans ce type de critère ; basé sur l'évaluation de la qualité par des observateurs humains ; l'image est classée dans des catégories de qualité : *excellente*, *bonne*, *acceptable*, *mauvaise* et *inacceptable*.

Le critère consiste à attribuer une note de qualité ; entre *5* (qualité *excellente*) et *1* (qualité *inacceptable*) ; par un ensemble d'observateurs. Cette note est obtenue en calculant la moyenne des résultats d'une série de tests standards où les observateurs donnent leurs avis sous la forme de notes pour évaluer la qualité de l'image. Ces tests standards exigent que les observateurs examinent les images dans les mêmes conditions, telles que la taille de l'image, la durée d'exposition et l'environnement lumineux dans lequel se déroule l'expérience [MAR03][CAL04][CAL06][CHR06].

Cependant, cette évaluation ne peut être que *subjective* puisqu'il n'existe aucune mesure correcte pouvant traduire fidèlement la perception de l'œil. Sans pour autant négliger ce critère, il est préférable d'introduire des critères plus *objectifs* [GOU02]. Les critères *objectifs* sont basés sur des critères mathématiques pour évaluer la qualité des images. Généralement, les mesures utilisées en compression sont l'*erreur moyenne quadratique* (MSE) et le *rapport signal à bruit* (SNR) (*Signal to Noise Ratio*).

IV.2.1 Erreur quadratique moyenne

La distorsion est l'erreur introduite par l'opération de compression, due au fait qu'éventuellement l'image reconstruite n'est pas exactement identique à l'image originale. L'*erreur quadratique moyenne* (MSE) est la plus simple mesure de la qualité de l'image reconstruite. Elle est définit par :

$$MSE = \frac{1}{N \times M} \times \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} (I(i,j) - \hat{I}(i,j))^2 \tag{IV.1}$$

où : I : Image originale et $\hat{I}$: Image reconstruite.

IV.2.2 Rapport signal à bruit en pic

La mesure la plus couramment utilisée par la communauté internationale est la mesure du *rapport signal à bruit* SNR. Elle est donné par :

$$SNR = 10 \times \log_2 \frac{\sum_{i=0}^{N-1} \sum_{j=0}^{M-1} I(i,j)^2}{\sum_{i=0}^{N-1} \sum_{j=0}^{M-1} (I(i,j) - \hat{I}(i,j))^2} \tag{IV.2}$$

Il existe différentes variantes à cette mesure, la plus utilisée est le *Peak Signal to Noise Ratio* (*PSNR*), évaluant le *rapport signal à bruit en pic* et se mesure en *décibel* (dB). Pour les images en niveaux de gris le *PSNR* est calculé par :

$$PSNR = 10 \times \log_2 \frac{255^2}{MSE} \tag{IV.3}$$

Pour les images couleurs le *PSNR* est définit par [DOU11][BOU12] :

$$PSNR = 10 \times \log_2 \frac{255^2 \times 3}{MSE(R) + MSE(G) + MSE(B)} \tag{IV.4}$$

où $MSE(\cdot)$ représente les *erreurs quadratiques moyennes* dans les trois plans R, G et B.

IV.2.3 Taux de compression

Le *taux de compression* (CR) (*Compression Ratio*) caractérise un facteur incontournable dans le domaine spécifique de la compression. Il donne une mesure de la réduction de la quantité d'informations à stocker. Plusieurs variantes de cette mesure existent et la couramment employée est [BEA97] [GOU02][DOU11] :

$$CR = \frac{La\,taille\,de\,l'image\,originale\,en\,bits}{La\,taille\,de\,l'image\,compressée\,en\,bits} \tag{IV.5}$$

Une autre variante est largement utilisée qui est le *bpp* (*bit per pixel*). Pour mesurer le pouvoir de compactage d'une méthode de compression, on utilise la notion de *débit binaire*. Ce qui donne une mesure en *bit par pixel* [DHA07] :

$$bpp = \frac{Nombre\,de\,bits\,nécessaires\,pour\,coder\,l'image}{Nombre\,de\,pixels\,de\,l'image} \tag{IV.6}$$

Les équations (IV.7) et (IV.8) permettent de calculer le *bpp* pour une image en niveaux de gris et couleurs respectivement :

$$bpp = \frac{8\,bits}{CR} \tag{IV.7}$$

$$bpp = \frac{3 \times 8\,bits}{CR} \tag{IV.8}$$

IV.2.4 Temps de calcul

La complexité d'un algorithme ou d'une méthode de compression est difficile à évaluer. Elle recouvre divers aspects tels que sa rapidité, l'existence de composants intégrés, la *charge de calcul* etc [VAL06]. La complexité fait également intervenir

les contraintes spécifiques à l'application, les performances des processeurs et le coût économique .

La complexité d'un algorithme peut se décrire par le temps nécessaire pour son exécution. Le plus souvent, les machines sont à architectures différentes et par conséquent des vitesses d'exécution différentes. C'est pour cela qu'on préfère exprimer la *charge de calcul* par une quantité absolue qui est le nombre d'opérations mathématiques. Ceci permet d'établir une différence entre les algorithmes *rapides* et les algorithmes *lents*, entre les algorithmes fonctionnant en *temps réel* et ceux ne pouvant pas le faire [TCH07].

En effet, il serait dommage, dans une application de transmission, que le temps gagné par une réduction de la taille des données à transmettre soit inférieur au temps passé à la *compression/décompression*. Cela sera cependant moins crucial dans des applications visant l'archivage de données [LIN04]. Il est important de noter que la charge de calcul n'est pas forcément identique entre le codeur et le décodeur. On parlera alors de compression *asymétrique*.

IV.3 Description de la technique élaborée

Dans cette section, nous présentons en détail notre algorithme dédié à la compression des images couleurs. Le schéma bloc de la technique proposée [MES16] est représenté à la figure IV.1. Il est composé des parties suivantes :

1. Conversion RGB / YCbCr,
2. Transformation DCT,
3. Seuillage,
4. Quantification,
5. Encodage (Encodeur proposé).

IV.3.1 Conversion RGB / YCbCr

Habituellement, les images à compresser sont représentées dans l'espace de couleur RGB. Or cet espace contient de fortes redondances *inter-plans* ce qui câble la compression directe de ces images [LUK06]. Cela est dû essentiellement à la concentration d'énergie dans les plans du modèle (table IV.1). Un changement d'espace de couleurs RGB vers autre moins corrélé comme YCbCr se voit nécessaire pour une compression

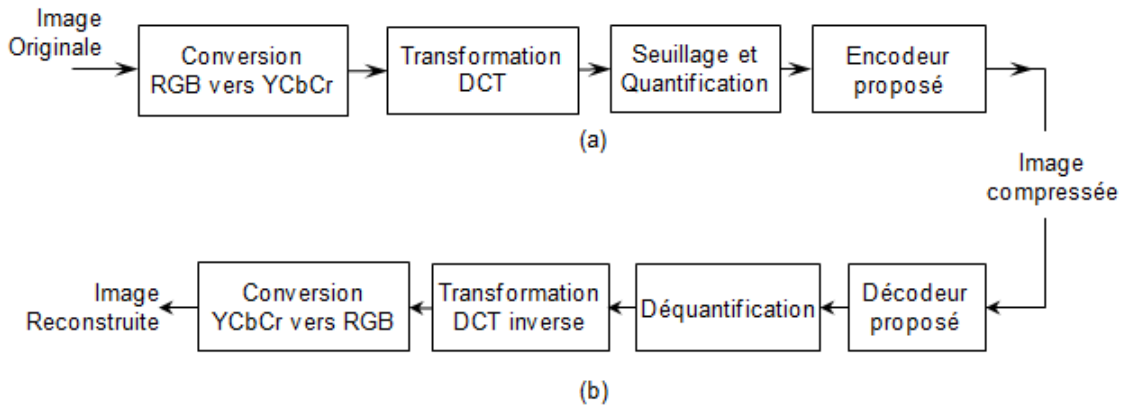


FIGURE IV.1 – Schéma bloc de la technique proposée [MES16] : (a) phase de *compression* et (b) phase de *décompression*.

efficace de ce type d'images. Ce changement permet une compression plus efficace car l'information est principalement condensée dans le plan Y. La conversion est réalisée par l'équation (IV.9) [RAB02] :

$$\begin{cases} Y &= 0.299 \cdot R &+0.587 \cdot G &+0.114 \cdot B \\ C_b &= -0.169 \cdot R &-0.331 \cdot G &+0.500 \cdot B \\ C_r &= 0.500 \cdot R &-0.418 \cdot G &-0.081 \cdot B \end{cases} \tag{IV.9}$$

Tandis que la conversion inverse est garantie par :

$$\begin{cases} R = 1 \cdot Y &+ &0 \cdot Cb &+ &1.402 \cdot Cr \\ G = 1 \cdot Y &- &0.344 \cdot Cb &- &0.714 \cdot Cr \\ B = 1 \cdot Y &+ &1.772 \cdot Cb &+ &0 \cdot Cr \end{cases} \tag{IV.10}$$

La table IV.1 récapitule la répartition des énergies de chaque plan pour l'ensemble des images de test. Pour l'espace RGB, les énergies sont calculées selon les équations (IV.11) à (IV.14) :

$$E_R = 100 \times \frac{\sum_{i=0}^{N-1} \sum_{j=0}^{M-1} R_{ij}^2}{T_{RGB}} \tag{IV.11}$$

$$E_G = 100 \times \frac{\sum_{i=0}^{N-1} \sum_{j=0}^{M-1} G_{ij}^2}{T_{RGB}} \tag{IV.12}$$

$$E_B = 100 \times \frac{\sum_{i=0}^{N-1} \sum_{j=0}^{M-1} B_{ij}^2}{T_{RGB}} \tag{IV.13}$$

$$T_{RGB} = \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} R_{ij}^2 + \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} G_{ij}^2 + \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} B_{ij}^2 \tag{IV.14}$$

TABLE IV.1 – Distribution d'énergies dans les espaces RGB et YCbCr.

Image	E_R (%)	E_G (%)	E_B(%)	E_Y (%)	E_{Cb} (%)	E_{Cr} (%)
Airplane	31,901	32,668	35,431	99,253	0,477	0,270
Peppers	49,109	38,080	12,811	86,007	5,614	8,380
Lena	58,373	21,095	20,533	89,530	1,503	8,967
Girl	32,361	32,926	34,713	99,587	0,052	0,361
Couple	46,763	28,888	24,349	93,029	2,310	4,661
House	33,354	30,901	35,744	96,507	1,044	2,449
Zelda	48,265	28,757	22,978	90,936	2,603	6,461
Moyenne	**42,875**	**30,474**	**26,651**	**93,55**	**1,943**	**4,507**

Tandis que les énergies sont données pour l'espace YCbCR, selon les équations (IV.15) à (IV.18) :

$$E_Y = 100 \times \frac{\sum_{i=0}^{N-1}\sum_{j=0}^{M-1} Y_{ij}^2}{T_{YCbCr}} \tag{IV.15}$$

$$E_{Cb} = 100 \times \frac{\sum_{i=0}^{N-1}\sum_{j=0}^{M-1} Cb_{ij}^2}{T_{YCbCr}} \tag{IV.16}$$

$$E_{Cr} = 100 \times \frac{\sum_{i=0}^{N-1}\sum_{j=0}^{M-1} Cr_{ij}^2}{T_{YCbCr}} \tag{IV.17}$$

$$T_{YCbCr} = \sum_{i=0}^{N-1}\sum_{j=0}^{M-1} Y_{ij}^2 + \sum_{i=0}^{N-1}\sum_{j=0}^{M-1} Cb_{ij}^2 + \sum_{i=0}^{N-1}\sum_{j=0}^{M-1} Cr_{ij}^2 \tag{IV.18}$$

De la table IV.1, il est clair que la concentration des énergies dans l'espace YCbCr est nettement déséquilibrée au profit du plan Y justifiant ainsi le choix de cet espace dans notre méthode de compression.

IV.3.2 Transformation DCT

Dans cette étape, on applique la DCT sur chaque plan de l'espace YCbCr. On commence par découper le plan Y en blocs de 8×8, 16×16 ou 32×32. Dans le cas où la taille de l'image n'est pas multiple de 8, 16 ou 32 respectivement, on duplique la dernière ligne autant de fois que cela est nécessaire. La transformation se fait plan par plan. Pour une image $I(Lignes \times Colonnes)$, le nombre de blocs $Nblocs$ pour chaque plan sera :

$$Nblocs = n \times m \quad \text{avec } n = \lceil \frac{Lignes}{N} \rceil \text{ et } m = \lceil \frac{Colonnes}{N} \rceil \quad N = 8, 16 \text{ où } 32 \tag{IV.19}$$

où $\lceil \cdots \rceil$ désigne l'arrondit à l'entier supérieur.

La transformée et la transformée inverse de chaque bloc sont calculées selon les équations (III.3) et (III.11) du chapitre précédent qui sont rappelées ici par :

$$\begin{cases} F(u,v) = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} C(u)\cdot C(v)\cdot f(i,j)\cdot cos\left[\frac{(2j+1)v\pi}{2N}\right]\cdot cos\frac{(2i+1)u\pi}{2N} \\ f(i,j) = \sum_{u=0}^{N-1}\sum_{v=0}^{N-1} C(u)\cdot C(v)\cdot F(u,v)\cdot cos\left[\frac{(2j+1)v\pi}{2N}\right]\cdot cos\frac{(2i+1)u\pi}{2N} \\ \qquad C(u) = \begin{cases} \sqrt{\frac{1}{N}} & \text{si } u = 0 \\ \sqrt{\frac{2}{N}} & \text{sinon} \end{cases} \end{cases} \tag{IV.20}$$

IV.3.3 Seuillage

En exploitant les caractéristiques du système visuel humain qui est moins sensible aux distorsions présentes sur les hautes fréquences, on peut employer un *seuillage* des coefficients appartenant à cette gamme de fréquences. En effet le *seuillage* consiste à éliminer les coefficients de faibles valeurs. Cette élimination permet d'avoir un gain en matière du taux de compression au détriment de la qualité de reconstitution.

On peut contrôler cette procédure on recherchant le seuil qui garantit un *PSNR* souhaité (FPSNR *Fixed PSNR*). Du fait que la mesure de qualité utilisée est monotone en fonction du seuil, cette recherche peut se faire par le bais des méthodes numériques usuelles telle que la méthode de *bissection* (*dichotomie*) en résolvant l'équation [BEN03][BEN05] :

$$PSNR(Thr) - FPSNR = 0 \tag{IV.21}$$

Habituellement, on cherche une solution Thr^* qui satisfait :

$$|PSNR(Thr^*) - FPSNR| \leqslant \epsilon \tag{IV.22}$$

où ϵ est la tolérance choisie généralement $\leqslant 10\%$ ($\epsilon = 0.5\%$).

Le *PSNR* est contrôlé en utilisant la *dichotomie*. Les étapes de cette dernière sont les suivantes [BEN05] :

- ETAPE 1 : *Initialisation*
  - Fixer en avance une valeur désirée dite *FPSNR* à atteindre par le *PSNR*,
  - Sélectionner l'intervalle de recherche $[Thr_{min}, Thr_{max}]$,
    Thr_{min} et Thr_{max} représentent les valeurs *minimale* et *maximale* de chaque plan (Y, Cb et Cr).
  - Choisir la précision de convergence ϵ,

  - Calculer $Thr = \frac{Thr_{min} + Thr_{max}}{2}$,

- ETAPE 2 : *Seuillage*

  - Mettre à zéro tous les coefficients inférieurs ou égaux à *Thr*,
    $C_{ij} = 0$, Si $|C_{ij}| \leqslant Thr$,

- ETAPE 3 : *Calculer le PSNR*

  - Calculer le *PSNR* de l'image seuillée avec la valeur actuelle de *Thr*,

- ETAPE 4 : *Mise à jour de Thr*

  - **Si** $PSNR < FPSNR$ **Alors** $Thr_{min} = Thr$,
    Sinon $Thr_{max} = Thr$,
  - Calculer $Thr = \frac{Thr_{min} + Thr_{max}}{2}$,

- ETAPE 5 : *Condition d'arrêt*

  - **Si** $|PSNR - FPSNR| > \epsilon$ **Alors** Aller à ETAPE 2,
    Sinon *FIN*.

- FIN

Suite à ce seuillage, on aperçoit généralement que la majorité des coefficients DCT *non nuls* se trouvent au coin supérieur à gauche (voir exemple ci-dessous). La figure IV.2 représente l'évolution du *Thr* en fonction du *PSNR* de l'image *Lena*.

IV.3.4 Quantification

En réalité, la transformation des pixels en domaine DCT n'aboutit pas à une compression. Un bloc de taille 64 pixels est transformé en 64 coefficients. En raison de l'orthonormalité de la DCT, l'énergie dans les deux domaines ne change pas, donc il n'y a pas de compression effectuée. Toutefois, la transformation entraine la concentration de l'énergie dans les composantes de basses fréquences. La majorité des coefficients qui en restent, représentent une faible énergie. Ce sont les étapes de *seuillage*, de *quantification* et du *codage* qui réduisent réellement le débit binaire.

Les coefficients DCT étant réels, une étape de *quantification* selon une résolution de bits est essentielle pour avoir des coefficients entiers. Ces derniers peuvent être codés et compressés efficacement par la suite. Le quantificateur utilisé dans notre méthode est

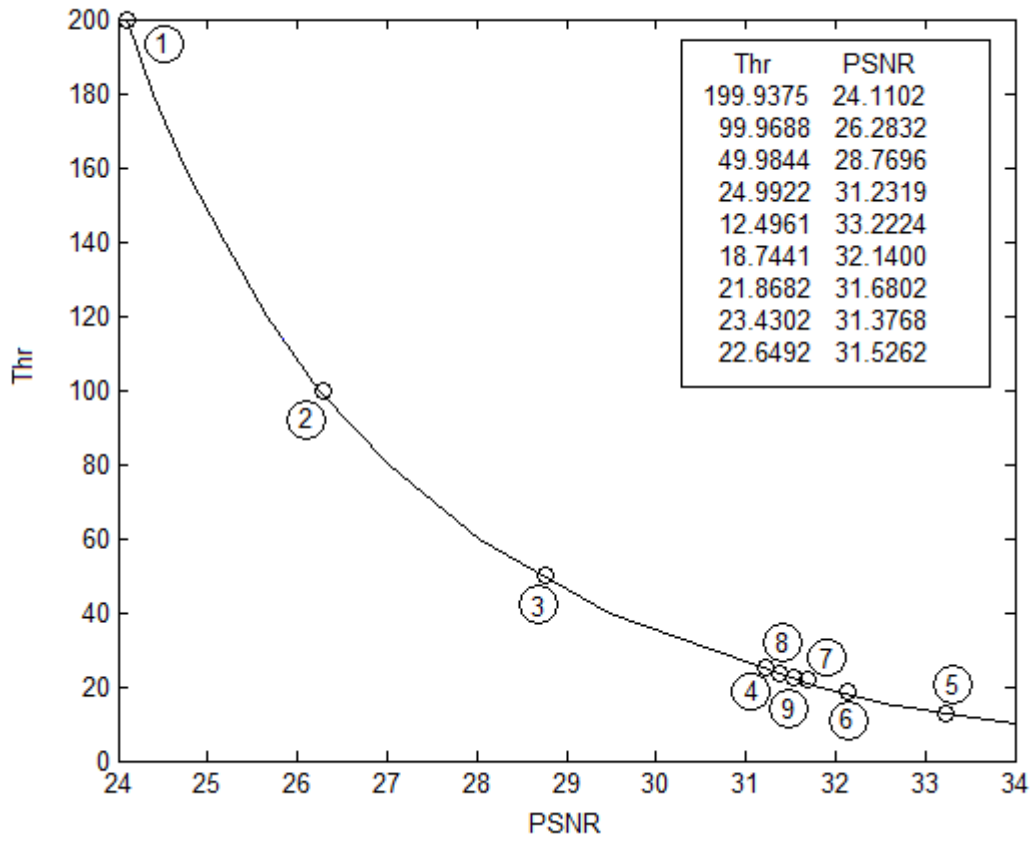


FIGURE IV.2 – Exemple d'une recherche par bissection UPSNR = 31,50 ϵ =0.1% Thr^*=22,65 UPSNR = 31,53 de l'image *Lena*.

un quantificateur *scalaire uniforme* définit par les équations (IV.23) et (IV.24) pour la *quantification* et la *déquantification* respectivement.

$$B_q = (2^Q - 1) \times \frac{B - B_{min}}{B_{max} - B_{min}} \tag{IV.23}$$

$$B_{dq} = \frac{B_{max} - B_{min}}{2^Q - 1} \times B_q + B_{min} \tag{IV.24}$$

où : - B représente le bloc d'image seuillé,
- B_q désigne le bloc quantifié,
- B_{dq} est le bloc déquantifié,
- B_{min} et B_{max} désignent les valeurs *minimale* et *maximale* du plan concerné.

Les étapes du *seuillage* et de la *quantification* sont responsables de la perte d'informations irrécupérables ce qui qualifie cette compression par une compression *avec pertes*.

IV.3.5 Encodage (Encodeur proposé) [MES16]

La dernière étape du processus de compression est le *codage*. C'est durant cette étape que va être généré le train de bits constituant l'image compressée. En effet, le bloc de coefficients DCT quantifié Bq est arrangé pour de produire un vecteur 1D contenant les différents coefficients quantifiés. L'approche utilisée pour le parcours du bloc 2D est présentée plus bas.

Le *codeur* proposé s'articule autour de quatre points essentiels :

- Établir la table LUT (*LookUp Table*) d'indices des coefficients *non nuls* NN de chaque bloc,
- Calcul de la table $dLUT$ contenant la *différence* d'indices du vecteur NN de chaque bloc à partir de la table LUT,
- Choisir le meilleur scan qui permet coder la table $dLUT$ avec le minimum de bits permettant ainsi de réduire la séquence de bits de chaque bloc,
- Coder les différents vecteurs nécessaires pour la compression du bloc.

IV.3.5.1 Codage

Pour encoder rentablement les blocs issus des étapes précédentes, nous avons proposé un *encodeur* simple et efficace. Le format de ce dernier est décrit à la figure IV.3. L'*encodeur* proposé est constitué des champs suivants :

- *Header* : le vecteur d'*entête* du bloc de taille fixe égale à P+2+P bits,
  - L (P bits) : définit le nombre de coefficients quantifié *non nuls* ;
  - AS (2 bits) : définit le mode de *scan* retenu (le meilleur *scan*) ;
  - q (P bits) : définit le nombre de bits nécessaire pour encoder la table $dLUT$;
- $dLUT$ ($q \times P$ bits) : le vecteur $dLUT$ contenant la *différence* d'indices du vecteur NN,
- NN ($q \times P$ bits) : le vecteur contenant les coefficients *non nuls* quantifié sur Q bits.

L'ordonnancement des coefficients constituant un bloc DCT peut se faire par plusieurs méthodes. La littérature spécialisée du domaine évoque plusieurs courbes de *scan* à savoir le scan *Zigzag*, *Hilbert*, *Vertical*, *Horizontal*, *Regazzoni*, *lexicographique* etc. Pour améliorer le taux de compression CR, *Douak et al* [DOU11] ont proposés un scan *adaptatif* afin d'avoir une *suite de zéros* la plus longue possible à la fin du

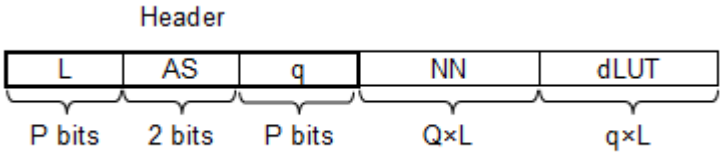


FIGURE IV.3 – Format du *codeur* proposé [MES16].

vecteur construit à partir du bloc de coefficients suivant quatre types de scan *Zigzag*, *Horizontal*, *Vertical* et *Hilbert* de la figure IV.4. Pour notre *encodeur*, nous avons retenu ce *scan adaptatif* mais pour un autre but.

Contrairement à la méthode [DOU11], notre *codeur* ne code que les coefficients *non nuls*. Les indices de ces coefficients sont stockés dont une table appelée LUT (*LookUp Table*). La méthode de lecture (*scan*) de ces coefficients utilise le *scan adaptatif* dans le but d'avoir le minimum nombre de bits q nécessaire pour encoder la *table de différence d'indices dLUT*. Cette table ($dLUT$) est utilisée pour enregistrer les différences d'indices des coefficients *non nuls* (LUT) de chaque bloc. En effet, l'utilisation de la différence d'indices au lieu des indices eux-mêmes permet un gain en plus. Le vecteur $dLUT$ est obtenu du vecteur LUT par le code suivant [MES16] :

$dLUT(1) = LUT(1)$
Pour $i = 2$ à L **Faire**
$\quad dLUT(i) = LUT(i) + LUT(i-1)$
Fin

Le *meilleur scan* parmi les quatre scans est codé sur 2 bits par le champ AS du codeur selon les cas suivants :

- $AS = 00$ pour le scan *Zigzag*,
- $AS = 01$ pour le scan *Horizontal*,
- $AS = 10$ pour le scan *Vertical*,
- $AS = 11$ pour le scan *Hilbert*.

A l'issue de cette étape, on a appliqué un codage *entropique* (codage *arithmétique*) pour comprimer davantage les vecteurs résultants. Un aperçu approfondi sur des vecteurs résultants (*Header*, *NN* et $dLUT$) permet de voir que la nature de leurs valeurs diffère les unes des autres. Cela nous a conduits à coder les vecteurs *Headers*, les vecteurs *NN* et les vecteurs $dLUT$ séparément dans le but de bien pousser le taux de compression plus loin (voir le Test 5 concernant l'effet du *codage séparé* plus loin).

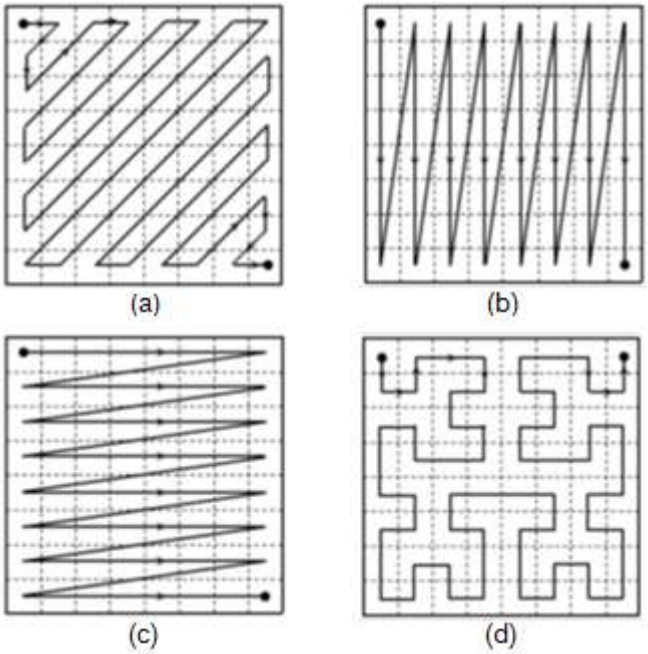


FIGURE IV.4 – Les différents types de scan : (a) *Zigzag*, (b) *Horizontal*, (c) *Vertical* et (d) *Hilbert*.

IV.3.5.2 Décodage

Au niveau du décodeur, l'*entête* du bloc est décodé en premier lieu pour extraire la valeur L qui représente la taille des vecteurs NN et $dLUT$. En effet, les P+2+P premiers bits forment le *Header* du bloc. Les P premiers bits de ce *Header* représentent la taille du vecteur NN qui représente les coefficients *non nuls*. Le scan adopté pour le bloc présent est donné par les deux bits qui suivent (AS), tandis que le nombre de bits du codage du vecteur $dLUT$ est obtenu par les P derniers bits du *Header*.

Les adresses des coefficients *non nuls* LUT peut être obtenu à partir du vecteur $dLUT$ par le pseudo code suivant [MES16] :

$LUT(1) = dLUT(1)$
Pour $i = 2$ à L **Faire**
    $LUT(i) = LUT(i-1) + dLUT(i)$
Fin

Pour éclaircir les idées, un exemple *illustratif* est donné ci-dessous comprenant les différents points du codeur proposé.

A. Exemple

Prenons comme exemple le premier bloc 8×8 du plan Y de l'image couleur *Lena* donné par :

$$B = \begin{matrix} 155 & 155 & 155 & 154 & 155 & 150 & 156 & 154 \\ 155 & 155 & 155 & 154 & 155 & 150 & 156 & 154 \\ 155 & 155 & 155 & 154 & 155 & 150 & 156 & 154 \\ 155 & 155 & 155 & 154 & 155 & 150 & 156 & 154 \\ 155 & 155 & 155 & 154 & 155 & 150 & 156 & 154 \\ 157 & 157 & 151 & 149 & 154 & 153 & 152 & 153 \\ 154 & 154 & 156 & 152 & 154 & 155 & 153 & 150 \\ 152 & 152 & 149 & 150 & 152 & 152 & 150 & 151 \end{matrix}$$

B. Codage

L'application de la transformation DCT et d'un arrondi à l'entier le plus proche donne le bloc suivant :

$$B_{DCT} = \begin{matrix} 1229 & 3 & 7 & -3 & 10 & 0 & 0 & -1 \\ 0 & 7 & -4 & 6 & -4 & 4 & -2 & 5 \\ 0 & 0 & -3 & 2 & 1 & 1 & 0 & -2 \\ -1 & 0 & 0 & 0 & 4 & 1 & 1 & 0 \\ 8 & -2 & 6 & -9 & 3 & -3 & 4 & 0 \\ -2 & 3 & -5 & 4 & -2 & 2 & -5 & 0 \\ 2 & 0 & 1 & -3 & 1 & -2 & 1 & 1 \\ 2 & -1 & 1 & -3 & 1 & -1 & -1 & 2 \end{matrix}$$

Après un seuillage (en valeur absolue) avec un *Thr=3*, on obtient le bloc $B_{DCT}TH$:

$$B_{DCT}TH = \begin{matrix} 1229 & 0 & 7 & 0 & 10 & 0 & 0 & 0 \\ 0 & 7 & -4 & 6 & -4 & 4 & 0 & 5 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 4 & 0 & 0 \\ 8 & 0 & 6 & -9 & 0 & 0 & 4 & 0 \\ 0 & 0 & -5 & 4 & 0 & 0 & -5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$$

La quantification du bloc $B_{DCT}TH$ produit le bloc des coefficients seuillés et quantifiés $B_{DCT}TH_q$:

$$B_{DCT}TH_q = \begin{matrix} 101 & 26 & 0 & 0 & 0 & 0 & 25 & 26 \\ 26 & 0 & 0 & 25 & 0 & 0 & 0 & 0 \\ 25 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$$

La figure IV.5 présente les différents scans et la table IV.2 récapitule les vecteurs des coefficients *non nuls NN*, LUT et $dLUT$. La valeur de q est calculée par l'équation :

$$q = \lceil \log_2(max(dLUT)) \rceil \tag{IV.25}$$

où $\lceil \cdots \rceil$ représente l'arrondis à l'entier supérieur.

ZG HZ Coef DCT VT HB							
1 1 **101** 1 1	2 2 **26** 9 2	6 3 **0** 17 15	7 4 **0** 25 16	15 5 **0** 33 17	16 6 **0** 41 20	28 7 **25** 49 21	29 8 **26** 57 22
3 9 **26** 2 4	5 10 **0** 10 3	8 11 **0** 18 14	14 12 **25** 26 13	17 13 **0** 34 18	27 14 **0** 42 19	30 15 **0** 50 24	43 16 **0** 58 23
4 17 **25** 3 5	9 18 **0** 11 8	13 19 **0** 19 9	18 20 **0** 27 12	26 21 **0** 35 31	31 22 **0** 43 30	42 23 **0** 51 25	44 24 **0** 59 26
10 25 **0** 4 6	12 26 **0** 12 7	19 27 **0** 20 10	25 28 **0** 28 11	32 29 **0** 36 32	41 30 **0** 44 29	45 31 **0** 52 28	54 32 **0** 60 27
11 33 **0** 5 59	20 34 **0** 1 58	24 35 **0** 21 55	33 36 **0** 29 54	40 37 **0** 37 33	46 38 **0** 45 36	53 39 **0** 53 37	55 40 **0** 61 38
21 41 **0** 6 60	23 42 **0** 14 57	34 43 **0** 22 56	39 44 **0** 30 53	47 45 **0** 38 34	52 46 **0** 46 35	56 47 **0** 54 40	61 48 **0** 62 39
22 49 **0** 7 61	35 50 **0** 15 62	38 51 **0** 23 51	48 52 **0** 31 52	51 53 **0** 39 47	57 54 **0** 47 46	60 55 **0** 55 41	62 56 **0** 63 42
36 57 **0** 8 64	37 58 **0** 16 63	49 59 **0** 24 50	50 60 **0** 32 49	58 61 **0** 40 48	59 62 **0** 48 45	63 63 **0** 56 44	64 64 **0** 64 43

FIGURE IV.5 – Coefficients DCT avec les différents scans.

TABLE IV.2 – Les vecteurs des coefficients *non nuls NN*, LUT et $dLUT$, les valeurs en **gras** représentent les valeurs maximales de chaque colonne.

Zigzag AS = 00			Horizontal AS = 01			Vertical AS = 10			Hilbert AS =11		
NN	LUT	dLUT	NN	LUT	dLUT	NN	LUT	dLUT	NN	LUT	dLUT
101	1	1	101	1	1	101	1	1	101	1	1
26	2	1	26	2	1	26	2	1	26	2	1
26	3	1	25	7	**5**	25	3	1	26	4	2
25	4	1	26	8	1	26	9	6	25	7	3
25	14	10	26	9	1	25	26	**17**	25	15	8
25	28	**14**	25	12	3	25	49	13	25	61	**46**
26	29	1	25	17	**5**	26	57	8	26	64	3
q=4			q=3			q=4			q=6		

Pour ce bloc on a :

- Pour le scan *Zigzag* : $q = \lceil \log_2(14) \rceil = 4$ bits,
- Pour le scan *Horizontal* : $q = \lceil \log_2(5) \rceil = 3$ bits,
- Pour le scan *Vertical* : $q = \lceil \log_2(17) \rceil = 4$ bits,
- Pour le scan *Hilbert* : $q = \lceil \log_2(46) \rceil = 6$ bits.

La valeur minimale de q ($q = 3$) correspond au scan *Horizontal* ($AS = 01$). Donc, ce scan est le meilleur scan qui produira la séquence binaire minimale.

La figure IV.6.a représente les vecteurs du *Header*, de NN et de $dLUT$ pour $Q = 7$ et $P = log_2(8 \times 8) = 6$ bits.
La taille de la séquence binaire résultante S est calculée par :
$S = P + AS + P + q \times L + Q \times L = 6 + 2 + 6 + 4 \times 7 + 7 \times 7 = 91$ bits.

La méthode CDABS [DOU11] est proche notre méthode mais elle utilise le quantificateur TRE [BEN08] et un encodeur de *Huffman*. Le format du codeur de cette méthode est décrit à la figure IV.7. Cet encodeur est constitué de :

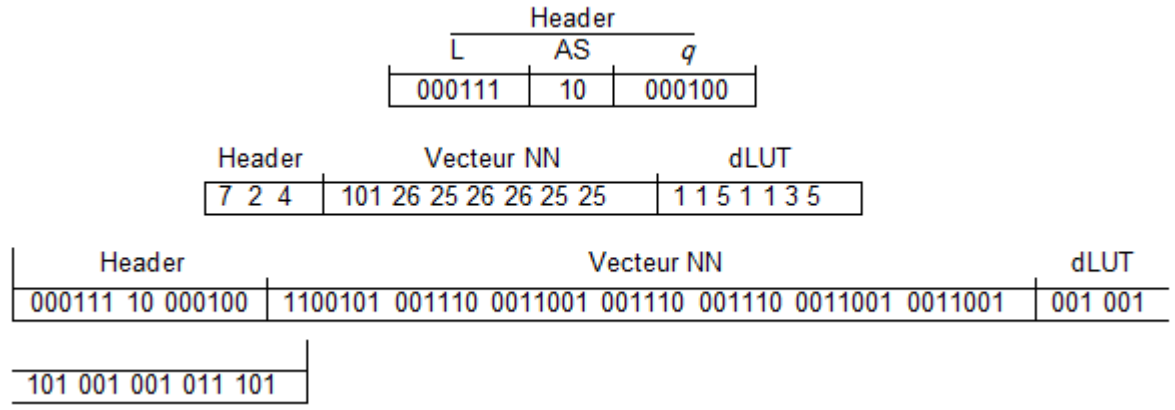
- A (1 bit) : représente l'activité du bloc, si tous les coefficients du bloc sont *nuls*, ce bit est mis à *0*.
- $PLNZ$ (P bits) : représente la position du dernier coefficient *non nul*.
  Avec $P = log_2(8 \times 8) = 6$ bits.
- AS (2 bits) : définit le mode de scan retenu (le meilleur scan).
- L ($(Q+1) \times (PLNZ+1)$ bits) : Vecteur qui contient tous les coefficients quantifiés.

L'application de la méthode CDABS [DOU11] sur le même bloc produit pour son meilleur scan ; qui est lui aussi le scan (*Horizontal*) ; la séquence S :
S={224 155 155 3 155 155 155 2 155 1 155 155 155 155 33 155}.

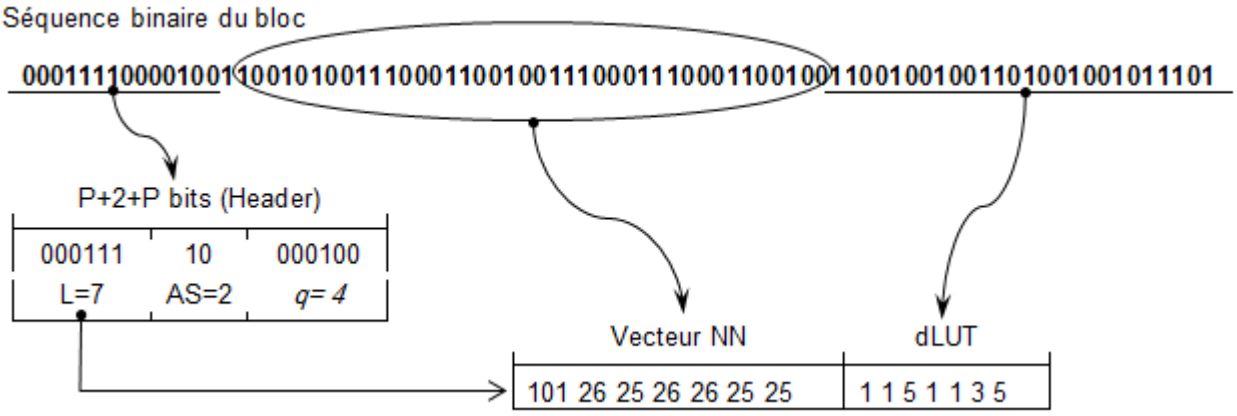
Il est à noter que pour le quantificateur TRE [BEN08], les coefficients sont codés avec $(Q+1)$ bits et les valeurs inférieurs à 2^Q représentent les longueurs des séquences de zéros successifs.
Dans ce cas :

- $L = 16$ valeurs et $P = log_2(8 \times 8) = 6$ bits,
- La taille de la séquence est :
  $S = A + AS + P + L \times (Q + 1) = 1 + 2 + 6 + 16 \times (7 + 1) = 137$ bits.

Une réduction de **33%** est obtenue sur ce bloc ce qui prouve l'efficacité du codage proposé. Les résultats de comparaison de notre méthode sur l'intégrité des images de test avec cette méthode et d'autres sont donnés plus loin dans ce chapitre.



(a)



(b)

FIGURE IV.6 – Exemple de codage et de décodage d'un bloc : (a) codage et (b) décodage.

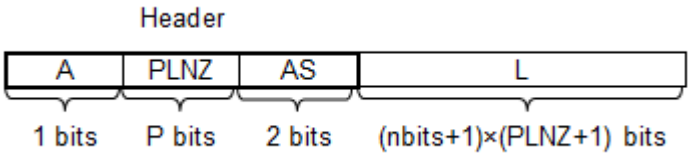


FIGURE IV.7 – Format du codeur de la méthode CDABS [DOU11].

C. Décodage

Les bits de l'*entête* du bloc permettent le décodage correct du bloc. C'est pour cela que le *Header* est décodé en premier lieu pour extraire les vecteurs *NN* et *dLUT* de

taille L. Le *Header* du bloc (figure IV.6.b) est formé par les P+2+P premiers bits de la séquence binaire du bloc. On aura donc les trois vecteurs suivants :

1. Le vecteur *Header* (P+2+P bits) (L, AS et q) : 000111 10 000100 décodé 3 2 4.
2. Le vecteur NN (L valeurs codées avec 7 bits) : 101 26 25 26 26 25 25.
3. Le vecteur $dLUT$ (L valeurs codées avec q bits) : 1 1 5 1 1 3 5.

Ce qui donne le vecteur : 1 2 7 8 9 12 17 qui est le même vecteur LUT de départ.

En utilisant le scan *Horizontal* (AS=2) et la table LUT, on remplira le bloc décodé B_{DEC} par les valeurs du vecteur NN. Les autres valeurs du bloc B_{DEC} sont initialement mises à zéro. Cela permet de restitué exactement le bloc $B_{DCT}TH_q$ de départ.

IV.4 Résultats expérimentaux

La simulation numérique est un outil d'analyse, d'étude et de mise en œuvre important en traitement et compression d'images. Par conséquent, elle permet de valider les études théoriques d'une part et l'optimisation du système par des tests sur ses divers paramètres. Cette partie du chapitre est consacrée aux expérimentations numériques et a pour but de valider la pertinence de la méthode proposée.

Les résultats de simulation ont été obtenus en utilisant le logiciel *MATLAB 2009b* installé sous *Windows 7* fonctionnant sur un PC portable *i3 234M 2.30GHz* et *4Go* de *RAM*. Les images couleurs utilisées dans les différents tests de simulation sont celles utilisées par plusieurs références [DHA07][DOU11][BOU12].

L'ensemble de ces images sont issues de la base de données *USC-SIPI* (*University of Southern California - Signal and Image Processing Institute*) [USC16]. Cette base a été créée dans le but de soutenir la recherche en permettant aux chercheurs de comparer leurs algorithmes sur les mêmes images, dans des domaines tels que le traitement et la compression d'images. La base de données *USC-SIPI* a été utilisée à de nombreuses reprises dans la littérature. Il s'agit d'un ensemble d'images au format TIFF, de différentes tailles (256×256, 512×512 et 1024×1024), en *noir* et *blanc*, en niveaux de gris ou en couleurs, et dont la plupart sont des images scannées. Les images sont réparties en différentes catégories : des images texturées, des vues aériennes, des portraits ou encore des images provenant de vidéos.

Dans le but de comparer notre méthode [MES16] avec d'autres méthodes, nous avons choisi l'ensemble des images de test utilisés dans les articles [DHA07][DOU11]

[BOU12]. Ces images sont représentatives et de nature différentes ce qui permet de tester efficacement notre algorithme.

L'ensemble des images de test contient les images en couleurs sans compression suivantes : *Aireplane.tiff*, *Peppers.tiff*, *Lena.tiff* et *Girl.tiff* de taille 512×512 et les images *Couple.tiff*, *House.tiff* et *Zelda.tiff* de taille 256×256. Ces images sont représentées à la figure IV.8.

Pour voir l'impact de chaque composante de la méthode proposée, un ensemble de tests est suggéré. Cet ensemble est composé de :

- Test 1 : Effet de la *transformation* RGB-YCbCr,
- Test 2 : Effet de *la taille du bloc* de découpage,
- Test 3 : Effet de *la taille du quantificateur* Q,
- Test 4 : Effet du *scan adaptatif*,
- Test 5 : Effet du *codage séparé* des vecteurs *Header*, *NN* et $dLUT$.

Chaque test est composé de plusieurs expériences. Dans la première expérience, la méthode proposée est appliquée en premier lieu sur chaque image avec un *PSNR* choisi dans un intervalle d'utilisation ordinaire (qualité d'image acceptable par l'ensemble des usagers). Ce *PSNR* varie selon la nature d'image. Nous visons par cette expérience le comportement de notre algorithme vis à vis la nature de chaque image en plus une comparaison avec d'autres algorithmes.

Une deuxième expérience vise la robustesse de notre méthode pour une diversité d'images. Cela est réalisé en prenant la moyenne des performances de toutes les images pour différentes valeurs du *PSNR*. Nous appliquerons notre méthode sur une image de l'ensemble de test comme une troisième expérience pour voir l'effet visuel de l'image compressée. L'amélioration en moyenne en termes de *CR* et *bpp* pour pratiquement le même *PSNR* est calculée par :

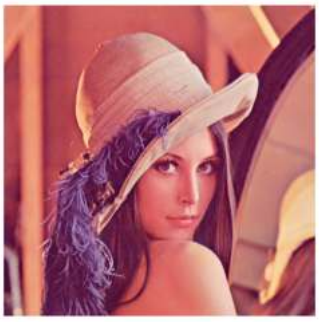
$$G_{CR} = 100 \times \frac{Nouveau\,CR - Ancien\,CR}{Ancien\,CR} \tag{IV.26}$$

et l'amélioration (*réduction*) pour le *bpp* est calculée par :

$$G_{bpp} = 100 \times \frac{Ancien\,bpp - Nouveau\,bpp}{Ancien\,bpp} \tag{IV.27}$$

(a)

(b)



(c)

(d)

(e)

(f)

(g)

FIGURE IV.8 – Images de test : (a) *Aireplane*, (b) *Peppers*, (c) *Lena*, (d) *Girl*, (e) *Couple*, (f) *House* et (g) *Zelda*[USC16].

IV.4.1 Test 1 : Effet de la transformation RGB-YCbCr

Nous souhaitons dans ce test voir l'intérêt de la conversion colorimétrique pour notre approche. Pour cela, nous avons appliqué notre algorithme sur l'ensemble d'images de

test dans l'espace RGB et dans l'espace YCbCr après la conversion colorimétrique suivant les équations (IV.9) et (IV.10).

Expérience 1 :

De la table IV.3, on peut voir les résultats en termes de *PSNR* et *bpp* pour chacune des images de test dans les deux espaces colorimétriques. Il est à remarquer que pour l'espace YCbCr, le meilleur résultat est pour $Q{=}8$ et une taille du bloc 16×16 par contre celui de l'espace RGB est pour $Q{=}7$ et une taille du bloc 16×16. Cela est reporté à la table IV.4 qui résume la table IV.3.

TABLE IV.3 – Performances de la compression de chaque image de test dans les espaces RGB et YCbCr.

	Espace	RGB						YCbCr					
	Bloc	8×8			16×16			8×8			16×16		
		PSNR	**bpp**	**CR**	**PSNR**	**bpp**	**CR**	**PSNR**	**bpp**	**CR**	**PSNR**	**bpp**	**CR**
$Q{=}7$	Airplane	31,16	1,00	23,92	31,16	0,73	32,77	31,14	0,66	36,58	31,16	0,48	49,79
	Peppers	31,27	1,16	20,74	31,20	0,90	26,59	31,18	1,04	23,17	31,12	0,97	24,66
	Lena	32,78	1,33	18,10	32,73	1,03	23,19	32,78	1,01	23,78	32,83	0,90	26,68
	Girl	35,98	0,85	28,15	35,94	0,66	36,37	35,88	0,45	53,93	35,89	0,38	63,11
	Couple	32,90	1,46	16,48	33,00	1,26	19,09	32,90	0,93	25,76	32,80	0,82	29,45
	House	32,15	1,16	20,63	32,00	0,94	25,40	32,14	0,90	26,53	32,11	0,78	30,79
	Zelda	32,01	1,44	16,71	32,01	1,18	20,33	32,08	0,99	24,21	31,94	0,86	27,93
	Moyenne	**32,61**	**1,20**	**20,67**	**32,58**	**0,96**	**26,25**	**32,59**	**0,85**	**30,56**	**32,55**	**0,74**	**36,06**
$Q{=}8$	Airplane	31,23	1,14	21,13	31,19	0,80	29,94	31,17	0,74	32,23	31,12	0,51	46,62
	Peppers	31,25	1,27	18,87	31,24	0,97	24,86	31,20	1,13	21,30	31,19	0,88	27,31
	Lena	32,77	1,46	16,49	32,79	1,06	22,74	32,70	1,03	23,29	32,65	0,74	32,43
	Girl	35,80	0,95	25,20	35,94	0,72	33,40	35,90	0,51	46,65	35,87	0,38	63,58
	Couple	32,79	1,61	14,89	32,92	1,34	17,96	32,84	1,02	23,44	32,85	0,82	29,25
	House	32,12	1,32	18,19	32,02	1,03	23,31	32,16	1,02	23,64	32,08	0,77	31,26
	Zelda	31,96	1,62	14,86	32,02	1,26	19,06	32,03	1,11	21,69	32,01	0,82	29,15
	Moyenne	**32,56**	**1,34**	**18,52**	**32,59**	**1,02**	**24,47**	**32,57**	**0,94**	**27,46**	**32,54**	**0,70**	**37,09**
$Q{=}9$	Airplane	31,14	1,27	18,91	31,14	0,90	26,56	31,04	0,84	28,67	31,08	0,57	41,90
	Peppers	31,27	1,44	16,72	31,19	1,08	22,28	31,24	1,28	18,77	31,20	0,97	24,84
	Lena	32,65	1,60	14,99	32,74	1,18	20,40	32,76	1,17	20,57	32,65	0,81	29,63
	Girl	35,83	1,11	21,63	35,87	0,81	29,63	35,93	0,62	38,64	36,00	0,44	54,40
	Couple	32,80	1,84	13,01	32,80	1,48	16,22	32,88	1,19	20,18	32,98	0,96	25,04
	House	32,14	1,54	15,59	32,08	1,18	20,29	32,20	1,19	20,15	32,02	0,86	27,78
	Zelda	31,98	1,86	12,89	31,94	1,41	17,07	32,07	1,28	18,68	32,11	0,96	24,94
	Moyenne	**32,55**	**1,52**	**16,25**	**32,54**	**1,15**	**21,78**	**32,59**	**1,08**	**23,67**	**32,58**	**0,80**	**32,65**

On remarque clairement que le l'espace YCbCr offre une nette amélioration en performance pour chaque image. Une amélioration en moyenne de **41,26%** pour le *CR* et de **27,08%** concernant le *pbb* (*réduction*) pour pratiquement le même *PSNR*. Cela justifie l'utilisation de cette transformation dans notre algorithme. Les graphes de la figure IV.9 illustrent en claire l'amélioration apportée par cette conversion colo-

rimétrique.

Expérience 2 :

Pour une comparaison judicieuse, on a utilisé l'ensemble des images de test pour différentes valeurs de *PNSR* et différentes valeurs de Q dans les deux espaces de couleur. La table IV.5 relève les résultats de cette expérience. Les graphes de la figure IV.10 montrent les différentes comparaisons entre les deux espaces de couleur. On remarque que l'espace YCbCr avec $Q=8$ montre sa supériorité sur les autres situations.

Expérience 3 :

La table IV.6, quant à elle, expose les résultats de la compression de l'image *Airplane* pour différentes valeurs du *PSNR* dans les espaces RGB et YCbCr. Les graphes de la figure IV.11 confirment la supériorité de l'utilisation de l'espace YCbCr par rapport à l'espace RGB. La figure IV.12 représente les images reconstruites dans l'espace RGB et YCbCr pour différentes valeurs du *PSNR* et *bpp*.

Conclusion

Comme résultat de ce test, la conversion colorimétrique sera adoptée dans notre algorithme.

IV.4.2 Test 2 : Effet de la taille du bloc de découpage

Le découpage de l'image en blocs est primordial pour la compression par DCT. Le présent test a pour fin de rechercher s'il existe une valeur de la taille du bloc qui donne des résultats meilleure que les autres tailles. Pour cela, nous avons utilisé trois valeurs de découpage 8×8, 16×16 et 32×32.

Expérience 1 :

La table IV.7 représente les résultats de la compression de chaque image de l'ensemble de test dans l'espace YCbCr pour $Q=8$ et $Q=9$. Une comparaison est faite à la figure IV.13 pour les différentes tailles du bloc de découpage. On remarque une amélioration en moyenne de **35,03%** (8×8) et **4,51%** (32×32) pour le *CR* et de **25,53%** (8×8) et **4,11%** (32×32) concernant le *bpp* pour pratiquement le même *PSNR*.

TABLE IV.4 – Performances de la compression de chaque image de test dans les espaces RGB et YCbCr avec la taille de bloc 16 × 16 (résumé de la table IV.3).

Espace	RGB (Q=7)			YCbCr (Q=8)		
Image	**PSNR**	**bpp**	**CR**	**PSNR**	**bpp**	**CR**
Airplane	31,16	0,73	32,77	31,12	0,51	46,62
Peppers	31,20	0,90	26,59	31,19	0,88	27,31
Lena	32,73	1,03	23,19	32,65	0,74	32,43
Girl	35,94	0,66	36,37	35,87	0,38	63,58
Couple	33,00	1,26	19,09	32,85	0,82	29,25
House	32,00	0,94	25,40	32,08	0,77	31,26
Zelda	32,01	1,18	20,33	32,01	0,82	29,15
Moyenne	**32,58**	**0,96**	**26,25**	**32,54**	**0,70**	**37,08**

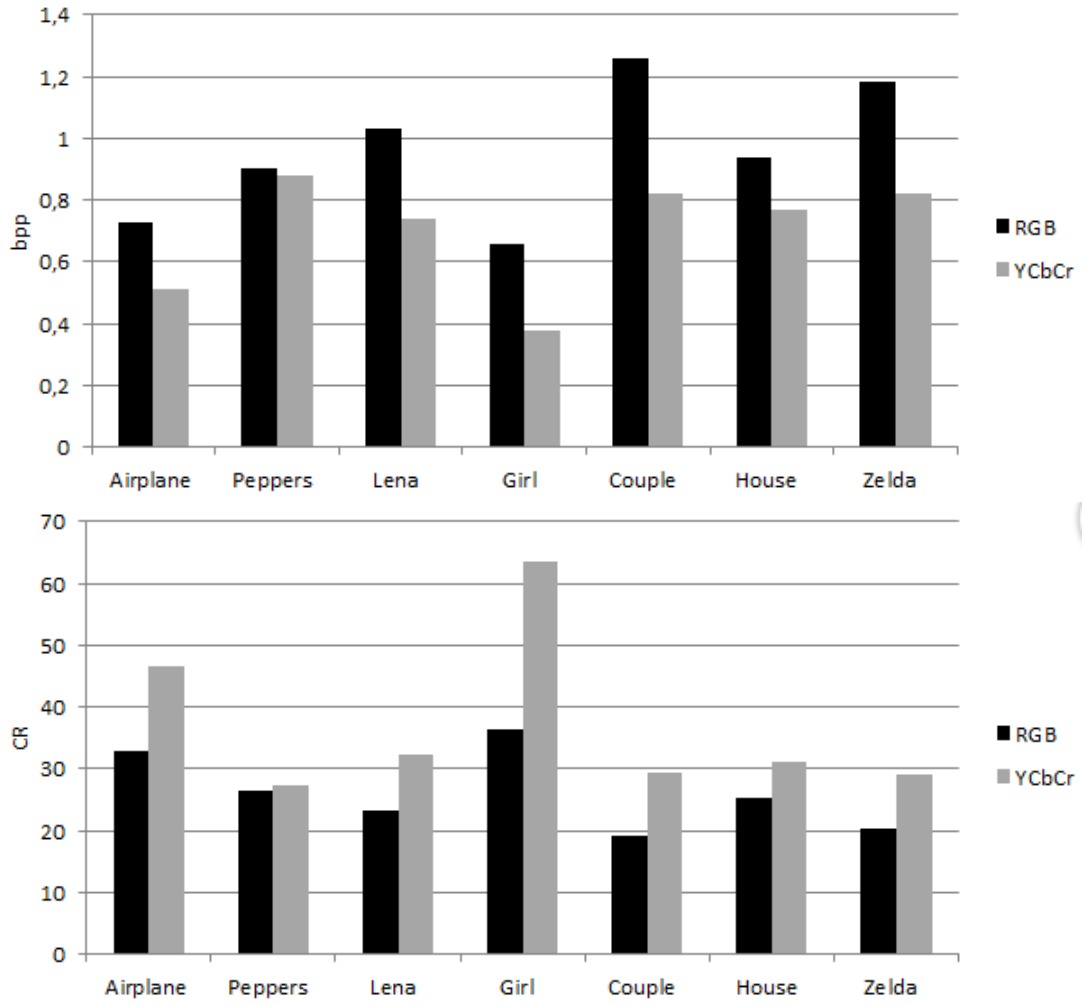


FIGURE IV.9 – Comparaison des performances de la compression de chaque image de test dans les espaces RGB et YCbCr (table IV.4).

TABLE IV.5 – Performances de la compression de l'ensemble des images de test dans les espaces RGB et YCbCr avec la taille du bloc 16 × 16.

Espace	RGB						YCbCr					
Q	7		8		9		7		8		9	
	PSNR	**bpp**	**PSNR**	**bpp**	**PSNR**	**bpp**	**PSNR**	**bpp**	**PSNR**	**bpp**	**PSNR**	**bpp**
	29,51	0,56	29,45	0,49	29,53	0,65	30,15	0,38	30,34	0,43	30,38	0,51
	30,27	0,65	30,18	0,57	30,29	0,75	30,83	0,44	31,08	0,51	31,15	0,59
	31,03	0,75	30,90	0,65	31,06	0,87	31,45	0,51	31,80	0,59	31,90	0,68
	31,81	0,88	31,62	0,77	31,86	1,01	31,97	0,59	32,46	0,68	32,58	0,78
	32,59	1,02	32,31	0,90	32,66	1,17	32,38	0,68	33,02	0,77	33,20	0,89
	33,36	1,20	32,97	1,06	33,47	1,38	32,84	0,81	33,71	0,93	34,01	1,06
	34,15	1,41	33,59	1,24	34,30	1,61	33,13	0,95	34,27	1,09	34,67	1,24
	34,92	1,67	34,13	1,46	35,14	1,90	33,35	1,14	34,93	1,31	35,46	1,49
Moyenne	**32,21**	**1,02**	**31,89**	**0,89**	**32,29**	**1,17**	**32,01**	**0,69**	**32,70**	**0,79**	**32,92**	**0,90**

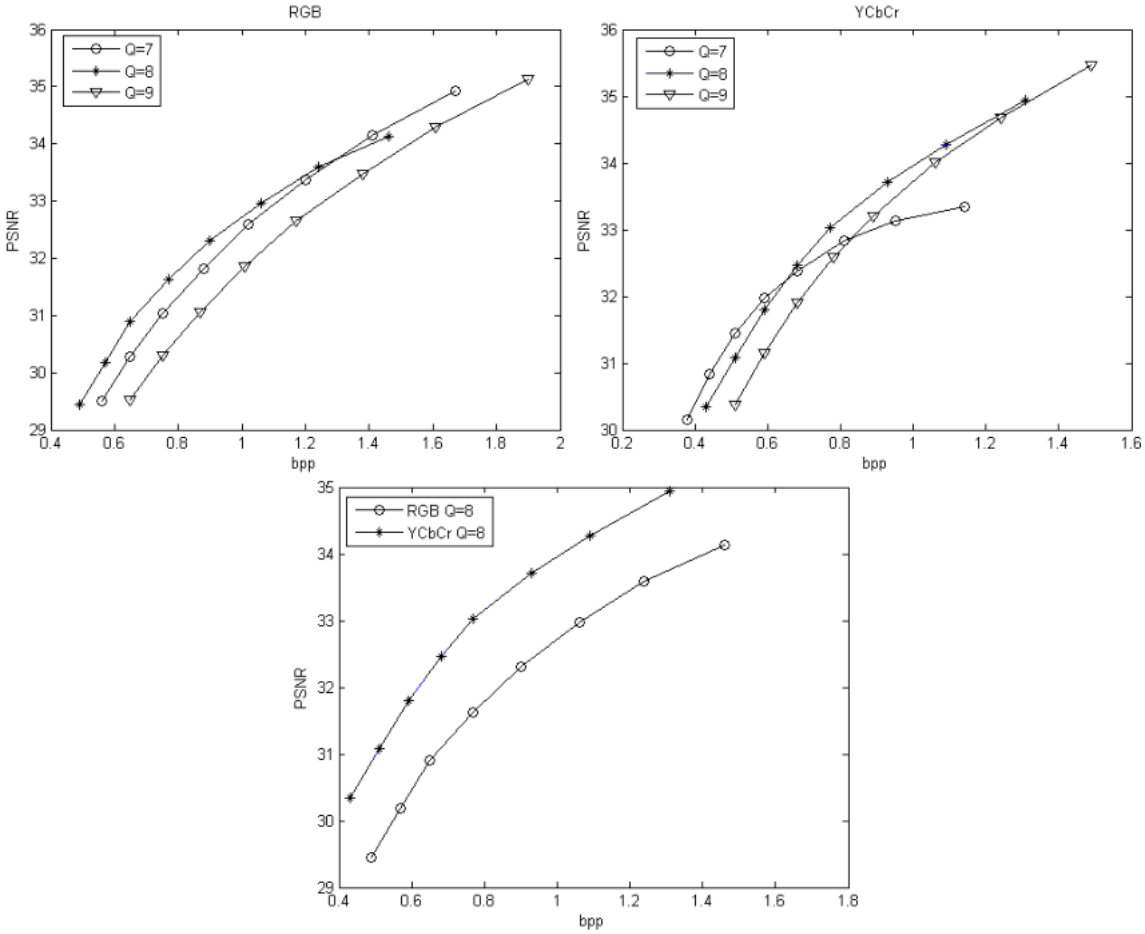


FIGURE IV.10 – Comparaison des performances de la compression de l'ensemble des images de test dans les espaces RGB et YCbCr (table IV.5).

TABLE IV.6 – Performances de la compression de l'image *Airplane* dans les espaces RGB et YCbCr avec la taille du bloc 16×16.

Espace	RGB						YCbCr					
Q	7		8		9		7		8		9	
	PSNR	**bpp**	**PSNR**	**bpp**	**PSNR**	**bpp**	**PSNR**	**bpp**	**PSNR**	**bpp**	**PSNR**	**bpp**
	27,70	0,39	27,75	0,45	27,76	0,52	29,13	0,33	29,36	0,38	29,41	0,44
	28,55	0,46	28,62	0,52	28,64	0,60	29,90	0,38	30,15	0,43	30,23	0,50
	29,39	0,53	29,50	0,61	29,53	0,70	30,64	0,43	30,96	0,50	31,08	0,57
	30,24	0,62	30,40	0,70	30,44	0,81	31,27	0,49	31,80	0,58	31,93	0,66
	30,96	0,71	31,19	0,80	31,25	0,92	31,69	0,57	32,56	0,65	32,75	0,75
	31,82	0,82	32,15	0,94	32,23	1,07	32,07	0,67	33,30	0,76	33,69	0,87
	32,56	0,94	33,02	1,08	33,15	1,23	32,22	0,77	34,01	0,90	34,54	1,00
	33,19	1,10	33,87	1,25	34,07	1,43	32,30	0,87	34,58	1,01	35,19	1,13
Moyenne	**30,55**	**0,69**	**30,81**	**0,79**	**30,89**	**0,91**	**31,15**	**0,56**	**32,09**	**0,65**	**32,35**	**0,74**

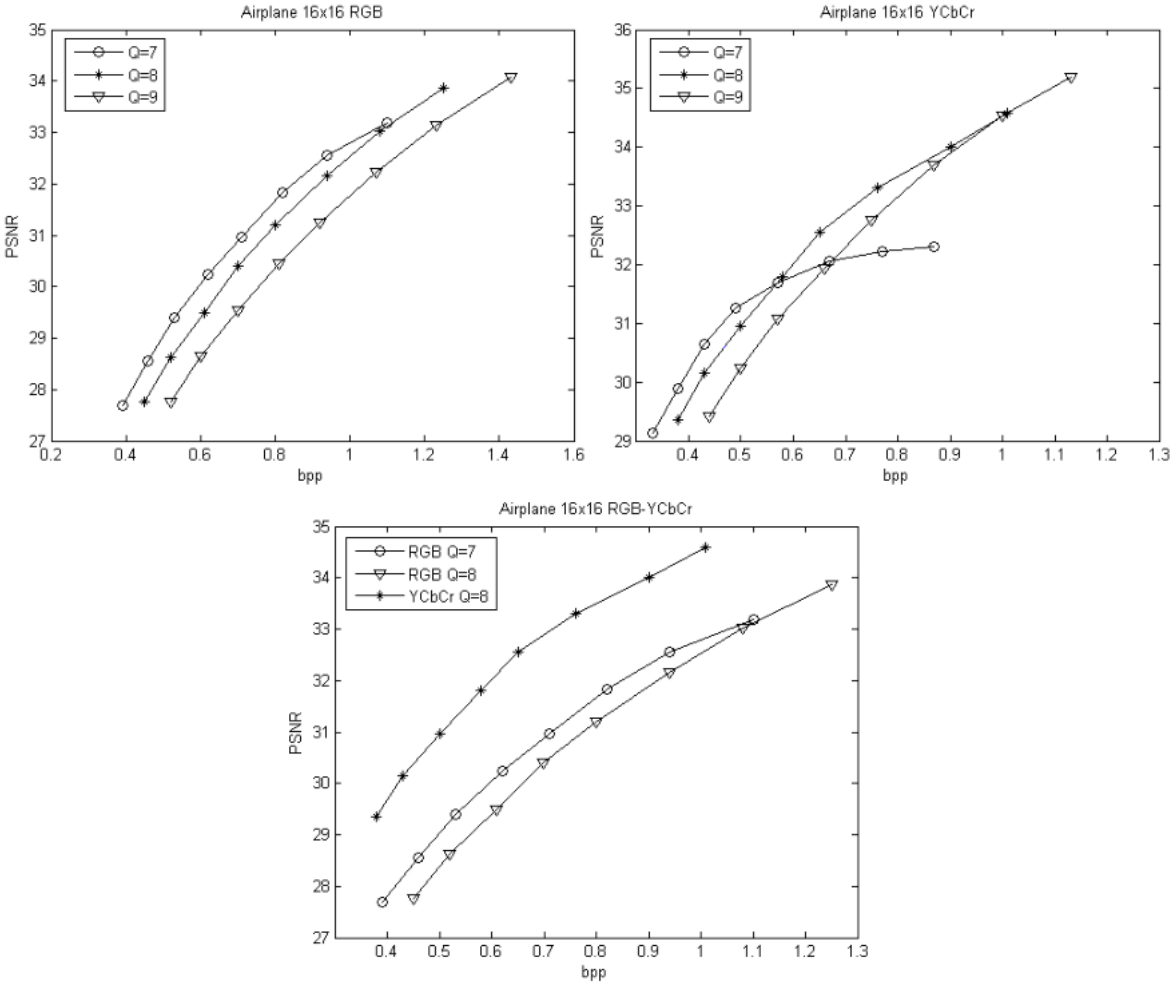
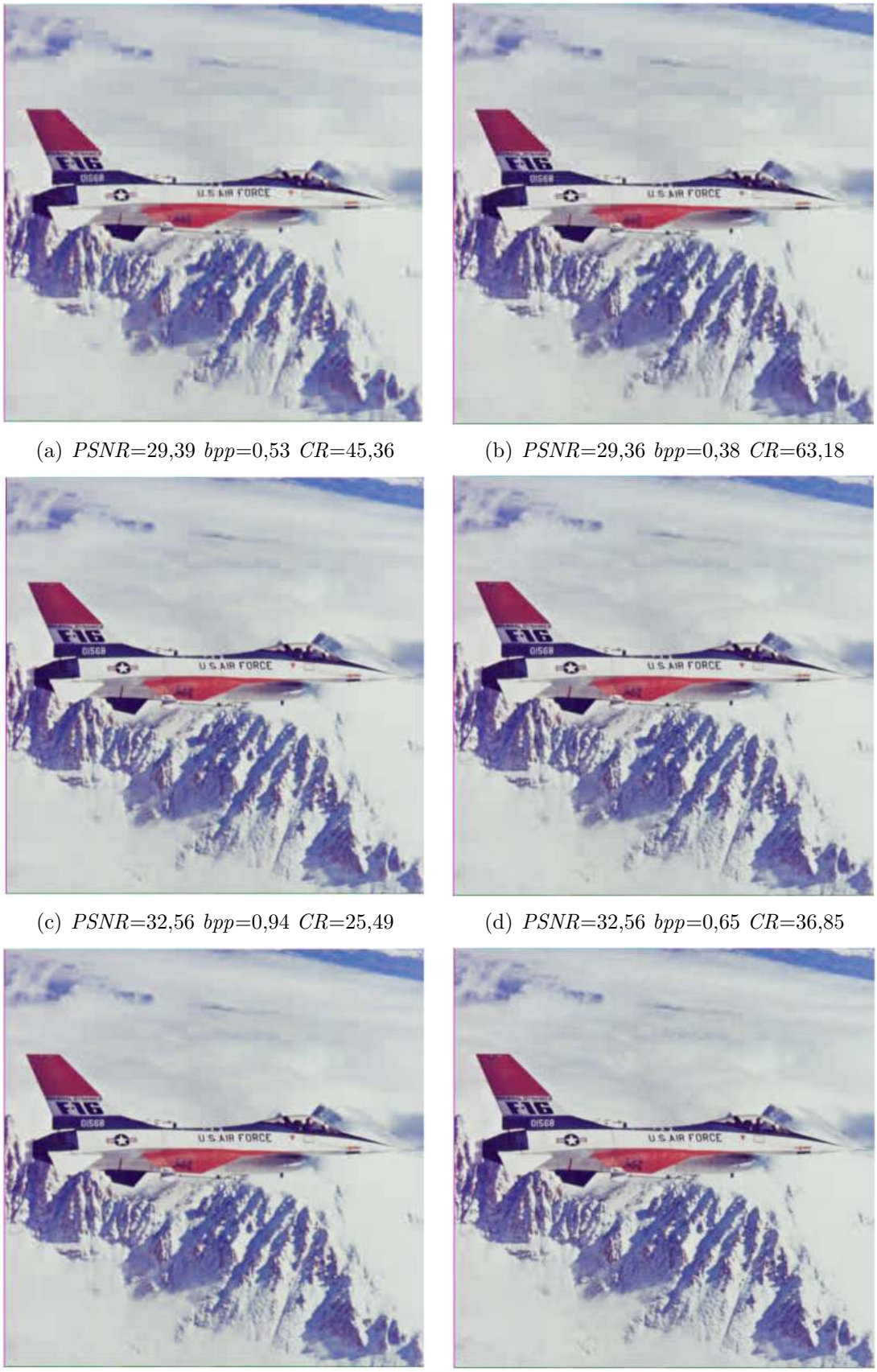


FIGURE IV.11 – Comparaison des performances de la compression de l'image *Airplane* dans les espaces RGB et YCbCr (table IV.6).



(a) *PSNR*=29,39 *bpp*=0,53 *CR*=45,36

(b) *PSNR*=29,36 *bpp*=0,38 *CR*=63,18

(c) *PSNR*=32,56 *bpp*=0,94 *CR*=25,49

(d) *PSNR*=32,56 *bpp*=0,65 *CR*=36,85

(e) *PSNR*=32,56 *bpp*=0,94 *CR*=25,49

(f) *PSNR*=34,01 *bpp*=0,90 *CR*=26,75

FIGURE IV.12 – Comparaison des performances de la compression de l'image *Airplane* dans les espaces RGB et YCbCr : (a), (c) et (e) RGB (Q=7) et (b), (d) et (f) YCbCr (Q=8) (table IV.6).

TABLE IV.7 – Performances de la compression de chaque image de test dans l'espace YCbCr pour les différentes tailles du bloc de découpage.

	Taille	8×8			16×16			32×32		
	Image	**PSNR**	**bpp**	**CR**	**PSNR**	**bpp**	**CR**	**PSNR**	**bpp**	**CR**
Q=8	Airplane	31,17	0,74	32,23	31,12	0,51	46,62	31,19	0,47	51,16
	Peppers	31,20	1,13	21,30	31,19	0,88	27,31	31,14	1,65	14,56
	Lena	32,70	1,03	23,29	32,65	0,74	32,43	32,84	0,85	28,33
	Girl	35,90	0,51	46,65	35,87	0,38	63,58	36,08	0,41	58,34
	Couple	32,84	1,02	23,44	32,85	0,82	29,25	33,05	0,88	27,32
	House	32,16	1,02	23,64	32,08	0,77	31,26	32,07	0,81	29,68
	Zelda	32,03	1,11	21,69	32,01	0,82	29,15	31,94	0,81	29,60
	Moyenne	**32,57**	**0,94**	**27,46**	**32,54**	**0,70**	**37,08**	**32,62**	**0,84**	**34,14**
Q=9	Airplane	31,04	0,84	28,67	31,08	0,57	41,90	31,23	0,51	46,83
	Peppers	31,24	1,28	18,77	31,20	0,97	24,84	31,09	0,94	25,43
	Lena	32,76	1,17	20,57	32,65	0,81	29,63	32,76	0,73	32,96
	Girl	35,93	0,62	38,64	36,00	0,44	54,40	36,09	0,41	58,07
	Couple	32,88	1,19	20,18	32,98	0,96	25,04	32,82	0,89	26,97
	House	32,20	1,19	20,15	32,02	0,86	27,78	32,19	0,82	29,31
	Zelda	32,07	1,28	18,68	32,11	0,96	24,94	31,94	0,83	28,81
	Moyenne	**32,59**	**1,08**	**23,67**	**32,58**	**0,80**	**32,65**	**32,59**	**0,73**	**35,48**

Conformément à la table IV.7, on observe que le découpage en bloc de 16×16 avec $Q{=}8$ donne les meilleurs résultats pour chaque image. Ce qui signifie que ce découpage est le mieux adapté indépendamment de la nature de l'image.

Expérience 2 :

Un autre essai cette fois-ci sur l'ensemble des images de test est effectué pour les différentes valeurs du *PSNR* dont les résultats sont affichés à la table IV.8. La supériorité du découpage $16{\times}16$ avec $Q{=}8$ est facilement repérable sur le graphe de la figure IV.14.

Expérience 3 :

La table IV.9 regroupe les résultats de l'application de la compression avec les différentes tailles de bloc sur l'image *Lena* avec plusieurs valeurs du *PSNR* dans l'espace YCbCr. Ces résultats sont traduits à la figure IV.15 qui montre clairement l'efficacité du découpage $16{\times}16$. La figure IV.16 représente les images reconstruites pour les différentes tailles du bloc de découpage et les différentes valeurs du *PSNR*.

Conclusion

Suite à ce test, le découpage en bloc de 16×16 sera retenu pour notre algorithme de compression et tout de long des tests qui suivent.

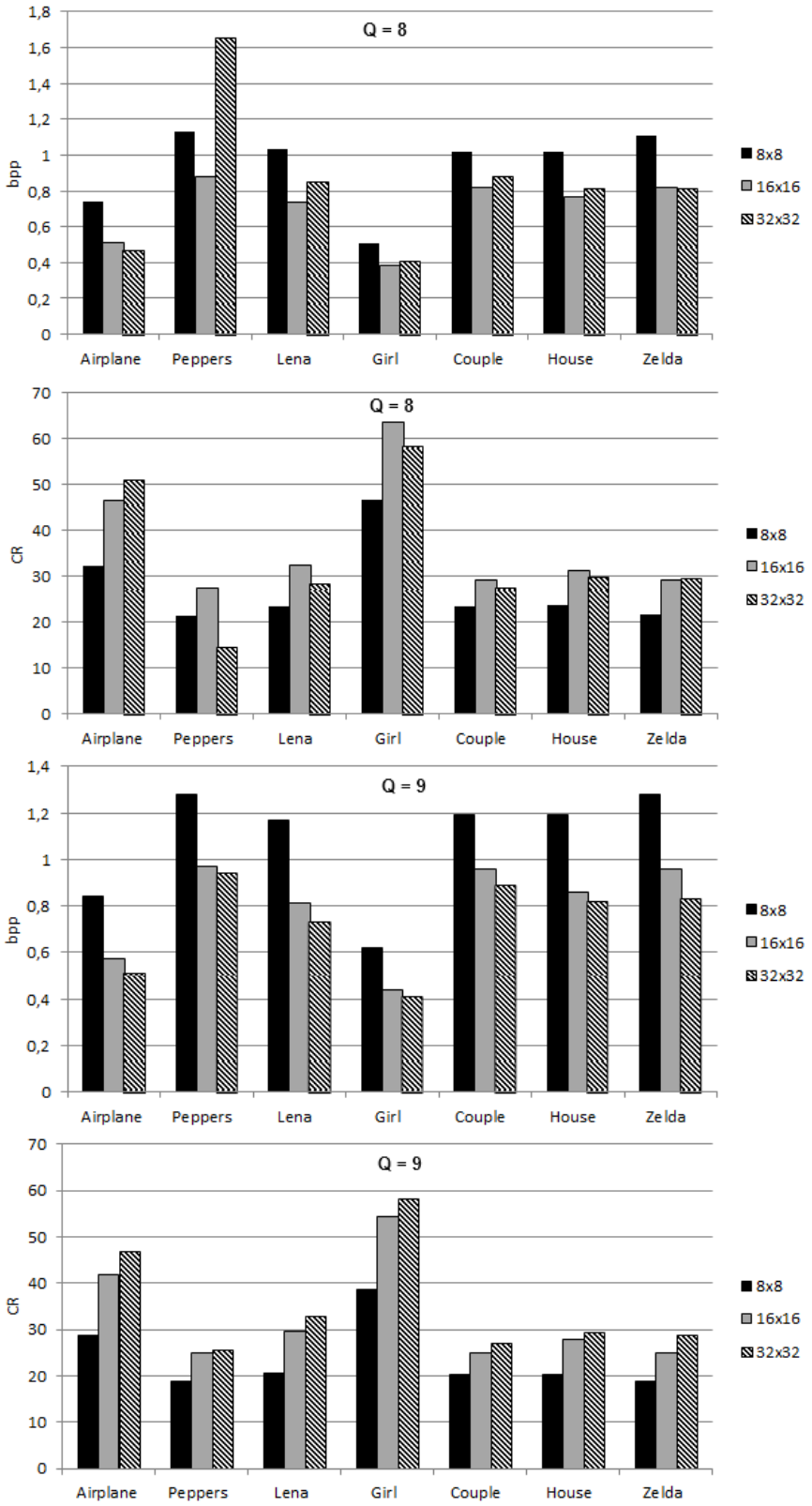


FIGURE IV.13 – Comparaison des performances de la compression de chaque image de test dans l'espace YCbCr pour différentes tailles du bloc de découpage (table IV.7).

TABLE IV.8 – Performances de la compression de l'ensemble des images de test dans l'espace YCbCr pour différentes valeurs du *PSNR* et avec différentes tailles du bloc.

Bloc	$8 \times 8\,(Q=7)$			$16 \times 16\,(Q=8)$			$32 \times 32\,(Q=9)$		
	PSNR	**bpp**	**CR**	**PSNR**	**bpp**	**CR**	**PSNR**	**bpp**	**CR**
	30,45	0,61	47,21	30,22	0,47	66,79	29,93	0,65	78,36
	30,80	0,65	44,79	30,54	0,51	62,52	30,30	0,69	72,22
	31,15	0,69	42,24	30,90	0,55	58,08	30,65	0,73	66,35
	31,46	0,74	40,09	31,24	0,61	53,89	31,02	0,78	61,17
	32,08	0,85	35,93	31,93	0,70	47,16	31,66	0,88	52,72
	32,70	0,98	31,82	32,58	0,82	41,21	32,33	1,00	44,92
	33,34	1,17	27,88	33,26	1,00	35,26	33,03	1,18	38,32
	33,93	1,35	24,52	33,85	1,17	30,62	33,64	1,35	32,85
Moyenne	**31,99**	**0,88**	**36,81**	**31,82**	**0,73**	**49,44**	**31,57**	**0,91**	**55,86**

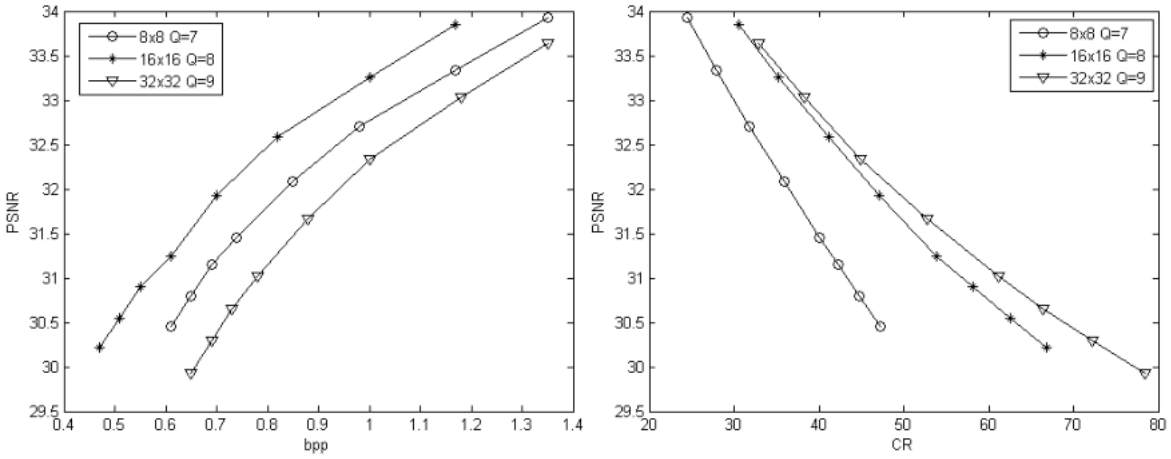


FIGURE IV.14 – Comparaison des performances de la compression de l'ensemble des images de test dans l'espace YCbCr pour les différentes tailles du bloc de découpage (table VI.8).

IV.4.3 Test 3 : Effet de la taille du quantificateur Q

Sans doute le nombre de bits du quantificateur Q d'un schéma de compression est un paramètre essentiel, important et décisif pour la qualité de l'image reconstruite. Une grande valeur de Q donne une qualité d'image élevée au prix d'un faible taux de compression CR tandis qu'un taux de compression élevé ne nécessite qu'une faible valeur de Q mais ceci produit une image reconstruite de qualité médiocre.

Suite à cela, une question qui se pose :

Existe-t-il une valeur de Q qui permet d'avoir une bonne qualité reconstruction avec un taux de compression acceptable ?

TABLE IV.9 – Performances de la compression de l'image *Lena* dans l'espace YCbCr pour les différentes tailles du bloc de découpage.

Bloc	$8 \times 8\,(Q=7)$			$16 \times 16\,(Q=8)$			$32 \times 32\,(Q=9)$		
	PSNR	**bpp**	**CR**	**PSNR**	**bpp**	**CR**	**PSNR**	**bpp**	**CR**
	29,03	0,45	53,33	28,95	0,28	85,71	29,86	0,42	57,14
	29,48	0,49	48,98	29,53	0,32	75,00	30,21	0,45	53,33
	29,99	0,54	44,44	30,06	0,36	66,67	30,56	0,49	48,98
	30,56	0,60	40,00	30,50	0,41	58,54	30,81	0,51	47,06
	31,03	0,66	36,36	30,95	0,48	50,00	31,50	0,60	40,00
	31,56	0,73	32,88	31,47	0,55	43,64	32,02	0,69	34,78
	31,98	0,81	29,63	31,92	0,62	38,71	32,25	0,73	32,88
	32,44	0,91	26,37	32,53	0,78	30,77	32,61	0,79	30,38
Moyenne	**30,76**	**0,65**	**39,00**	**30,74**	**0,48**	**56,13**	**31,36**	**0,61**	**42,15**

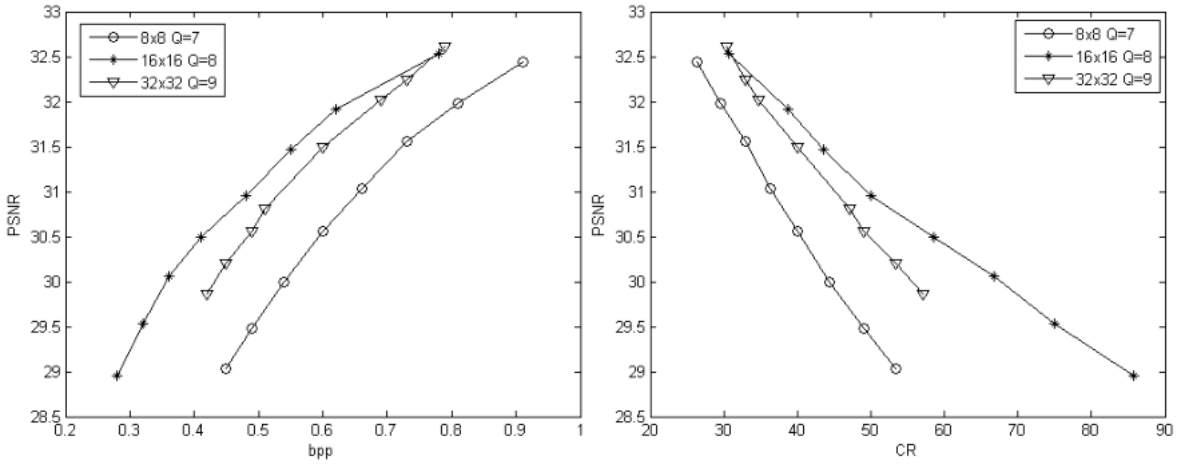


FIGURE IV.15 – Comparaison des performances de la compression de l'image *Lena* dans l'espace YCbCr pour les différentes tailles du bloc de découpage (table IV.15).

Pour répondre à cette question, un test pour la recherche d'une telle valeur est indispensable. D'après les tests précédents, on a vu que l'espace YCbCr est préféré sur l'espace RGB et que le découpage 16×16 est favorisé sur les autres tailles de bloc. Donc, le présent test sera effectué dans l'espace YCbCr et avec une taille du bloc 16×16.

Expérience 1 :

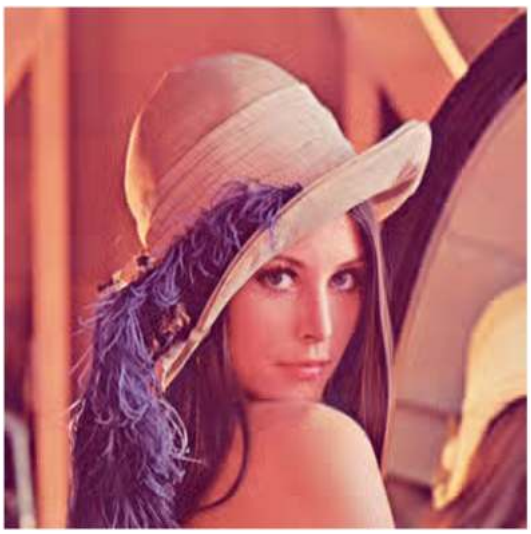
Dans cette expérience, on comprime chaque image suivant son *PSNR* approprié suivant différentes valeurs de Q. Les résultats obtenus ; qui sont répertoriés à la table IV.10 et illustrés à la figure IV.17 ; témoignent que le nombre de bits du quantificateur Q=*8* donne un résultat meilleur que les autres valeurs. Ce qui traduit une amélioration en moyenne de **2,86%** (Q=7), **13,60%** (Q=9) pour le *CR* et de **5,41%** (Q=7), **12,50%** (Q=9) concernant le *bpp* pour pratiquement le même *PSNR*.

(a) *PSNR*=30,56 *bpp*=0,60 *CR*=40,00

(b) *PSNR*=32,44 *bpp*=0,91 *CR*=26,37

(c) *PSNR*=30,50 *bpp*=0,41 *CR*=58,54

(d) *PSNR*=32,53 *bpp*=0,78 *CR*=30,77



(e) *PSNR*=30,56 *bpp*=0,49 *CR*=48,98

(f) *PSNR*=32,61 *bpp*=0,79 *CR*=30.38

FIGURE IV.16 – Comparaison des performances de la compression de l'image *Lena* dans l'espace YCbCr pour les différentes tailles du bloc de découpage : (a) et (b) 8×8 (Q=7), (c) et (d) 16×16 (Q=8), (e) et (f) 32×32 (Q=9) (table IV.9).

TABLE IV.10 – Performances de la compression de chaque image de test dans l'espace YCbCr avec la taille du bloc 16×16 pour différentes valeurs de Q.

Q	7			8			9			10		
	PSNR	**bpp**	**CR**	**PSNR**	**bpp**	**CR**	**PSNR**	**bpp**	**CR**	**PSNR**	**bpp**	**CR**
Airplane	31,16	0,48	49,79	31,12	0,51	46,62	31,08	0,57	41,90	31,10	0,66	36,34
Peppers	31,12	0,97	24,66	31,19	0,88	27,31	31,20	0,97	24,84	30,21	0,85	28,12
Lena	32,83	0,90	26,68	32,65	0,74	32,43	32,65	0,81	29,63	31,80	0,76	31,60
Girl	35,89	0,38	63,11	35,87	0,38	63,58	36,00	0,44	54,40	35,20	0,44	54,97
Couple	32,80	0,82	29,45	32,85	0,82	29,25	32,98	0,96	25,04	31,90	0,88	27,27
House	32,11	0,78	30,79	32,08	0,77	31,26	32,02	0,86	27,78	31,84	0,96	24,91
Zelda	31,94	0,86	27,93	32,01	0,82	29,15	32,11	0,96	24,94	31,38	0,96	25,05
Moyenne	**32,55**	**0,74**	**36,06**	**32,54**	**0,70**	**37,09**	**32,58**	**0,80**	**32,65**	**31,92**	**0,79**	**32,61**

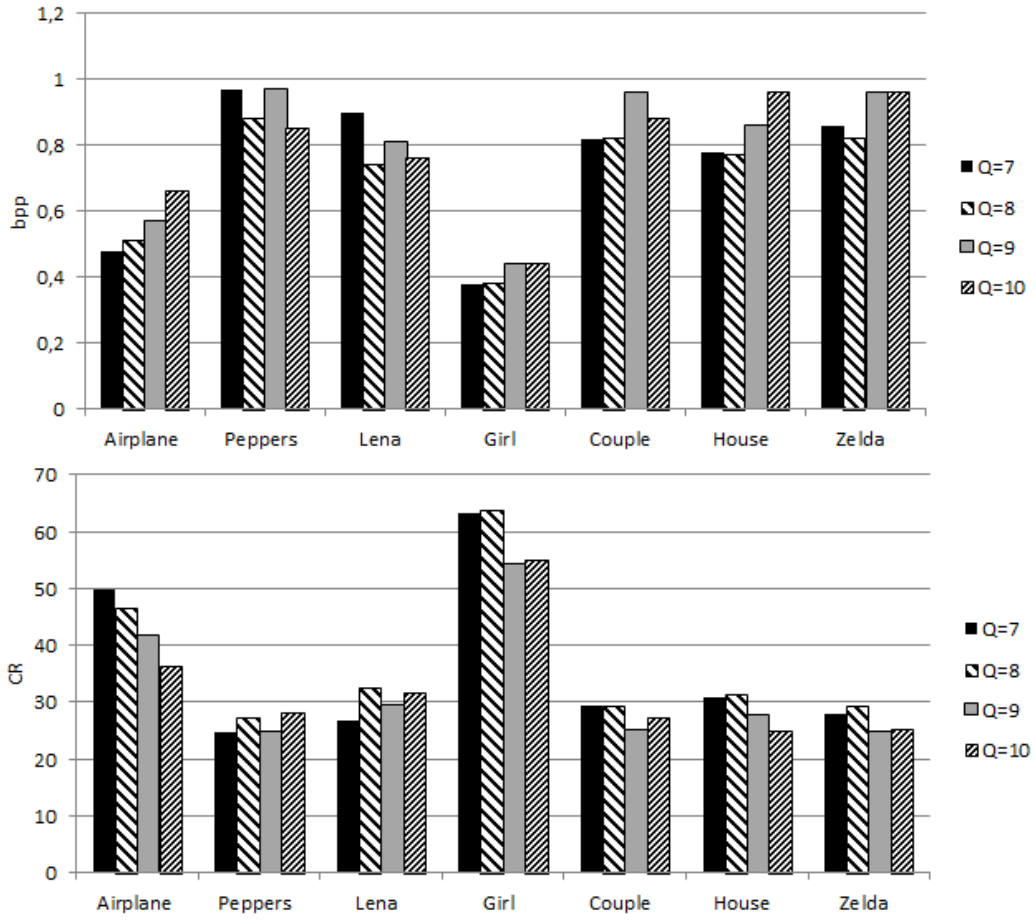


FIGURE IV.17 – Comparaison des performances de la compression de chaque image de test dans l'espace YCbCr avec la taille du bloc 16×16 pour différentes valeurs de Q (table IV.10).

Expérience 2 :

Le résultat de l'expérience précédente se confirme en appliquant la compression sur l'ensemble des images de test pour différentes valeurs du *PSNR* comme il claire dans la figure IV.18. Les résultats sont répertoriés à la table IV.11.

Cela permet de dégagé que la valeur $Q=8$ donne de bons résultats indépendamment de la qualité de l'image reconstruite.

TABLE IV.11 – Performances de la compression de l'ensemble des images de test dans l'espace YCbCr pour différentes valeurs du *PSNR* et avec différentes valeurs de Q.

Q	7			8			9			10		
	PSNR	**bpp**	**CR**	**PSNR**	**bpp**	**CR**	**PSNR**	**bpp**	**CR**	**PSNR**	**bpp**	**CR**
	31,55	0,52	49,08	31,40	0,54	47,57	31,30	0,61	42,39	31,16	0,68	37,53
	31,89	0,58	44,61	31,92	0,60	42,62	31,75	0,66	38,87	31,92	0,79	32,61
	32,19	0,63	40,57	32,77	0,72	35,64	32,28	0,73	34,98	32,27	0,85	30,59
	32,40	0,68	37,72	33,06	0,77	33,32	32,62	0,79	32,59	32,61	0,90	28,58
	32,67	0,76	34,18	33,45	0,86	30,16	32,91	0,84	30,67	32,93	0,96	26,88
	32,87	0,82	31,46	33,75	0,93	27,61	33,24	0,89	28,68	33,24	1,02	25,17
	33,13	0,94	27,83	33,96	1,00	26,17	33,69	0,99	26,24	33,66	1,12	23,28
	33,29	1,06	24,64	34,24	1,08	24,22	34,05	1,06	24,13	34,07	1,21	21,39
Moyenne	**32,50**	**0,75**	**36,26**	**33,07**	**0,81**	**33,41**	**32,73**	**0,82**	**32,32**	**32,73**	**0,94**	**28,25**

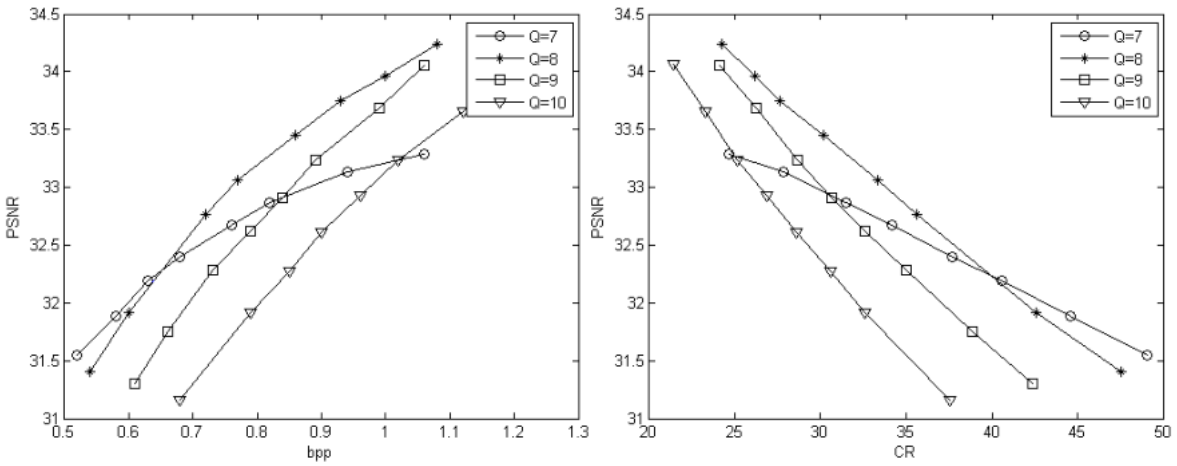


FIGURE IV.18 – Comparaison des performances de la compression de l'ensemble des images de test dans l'espace YCbCr avec la taille du bloc 16×16 pour différentes valeurs de Q (table IV.11).

Expérience 3 :

Comme pour les tests précédents, on applique l'algorithme proposé sur une image de l'ensemble de test (*Zelda*) pour différentes valeurs du *PSNR* avec différentes valeurs de Q. Les résultats de cette expérience sont regroupés dans la table IV.12 et tracés à la figure IV.19. La figure IV.20 représente les images reconstruites pour différentes valeurs du *PSNR* avec différentes valeurs de Q.

TABLE IV.12 – Performances de la compression de l'image *Zelda* dans l'espace YCbCr avec la taille du bloc 16 × 16 pour différentes valeurs de Q.

Q	7			8			9			10		
	PSNR	**bpp**	**CR**	**PSNR**	**bpp**	**CR**	**PSNR**	**bpp**	**CR**	**PSNR**	**bpp**	**CR**
	30,25	0,52	46,32	30,20	0,56	42,93	30,24	0,65	36,76	29,86	0,70	34,07
	30,70	0,57	42,15	30,45	0,59	40,40	30,50	0,69	34,69	30,68	0,82	29,13
	31,00	0,61	39,06	30,93	0,66	36,16	31,01	0,77	31,34	31,38	0,96	25,05
	31,40	0,68	35,28	31,28	0,72	33,31	31,36	0,82	29,16	32,14	1,11	21,67
	31,81	0,81	29,74	31,81	0,80	30,13	31,91	0,93	25,91	32,59	1,22	19,66
	32,23	0,99	24,30	32,42	0,91	26,45	32,56	1,06	22,65	33,43	1,47	16,29
	32,44	1,17	20,51	33,08	1,14	21,10	33,36	1,29	18,55	33,77	1,61	14,98
	33,05	1,70	14,12	33,62	1,36	17,64	34,03	1,54	15,63	34,11	1,76	13,67
Moyenne	**31,61**	**0,88**	**31,44**	**31,72**	**0,84**	**31,02**	**31,87**	**0,97**	**26,84**	**32,25**	**1,21**	**21,81**

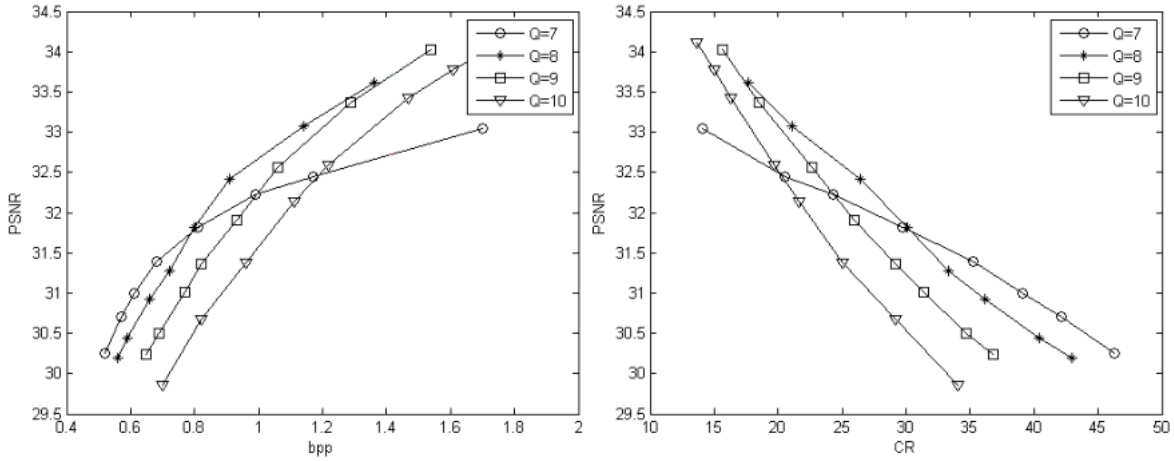


FIGURE IV.19 – Comparaison des performances de la compression de l'image *Zelda* dans l'espace YCbCr avec la taille du bloc 16 × 16 pour différentes valeurs de Q (table IV.12).

Conclusion

Selon les expériences de ce test, on peut dire que la valeur donnant les meilleures performances est la valeur Q=*8*. Par conséquence, cette valeur sera retenue pour notre approche.

IV.4.4 Test 4 : Effet du *scan adaptatif*

Certes le *scan Zigzag* permet de retrouver les coefficients basse fréquences en premier lieu dans le but d'avoir la séquence la plus longue possible de zéros. Cela n'empêche pas que dans certaines situations, il y a d'autres chemins qui retrouvent les coefficients *non nuls* en premier lieu mieux que le *scan Zigzag*.

Dans cette optique, les expériences de ce test donnent les répartitions des meilleurs scans dans le but de trouver une combinaison des différents scans améliorant plus la compression.

(a) *PSNR*=30,25 *bpp*=0,52 *CR*=46,32

(b) *PSNR*=32,23 *bpp*=0,99 *CR*=24,30

(c) *PSNR*=30,20 *bpp*=0,56 *CR*=42,93

(d) *PSNR*=32,42 *bpp*=0,91 *CR*=26,45

(e) *PSNR*=30,24 *bpp*=0,65 *CR*=36,76

(f) *PSNR*=32,56 *bpp*=1,06 *CR*=22,65

FIGURE IV.20 – Comparaison des performances de la compression de l'image *Zelda* dans l'espace YCbCr avec la taille du bloc 16×16 pour différentes valeurs de Q : (a) et (b) Q=7, (c) et (d) Q=8 et (e) et (f) Q=9 (table IV.12).

Expérience 1 :

Ci-dessous, on peut trouver les résultats de l'exécution de notre méthode sur chaque image de l'ensemble d'images de test pour le *scan Zigzag* seul et pour un *scan adaptatif*. La table IV.13 présente les performances de la compression pour les deux scans dans l'espace YCbCr pour $Q=8$ et un découpage 16×16. La figure IV.21 illustre la comparaison entre les deux modes de scan.
De cela, on peut affirmer que le *scan adaptatif* apporte une amélioration à la compression. Cette amélioration est en moyenne de **6,80%** pour le *CR* et de **6,67%** concernant le *bpp* pour le même *PSNR*.

Il est claire à remarquer que les résultats du *scan adaptatif* sont meilleurs que ceux du *scan Zigzag* seul pour chaque image. Cela signifie que le *scan adaptatif* est loin d'être dépendant du contenu de l'image. Cela justifie notre choix de retenir ce scan dans notre approche.

Expérience 2 :

Pour voir si la valeur du *PSNR* de l'image reconstruite influe sur l'avantage du *scan adaptatif*, on applique notre méthode sur l'ensemble des images de test pour différentes valeurs du *PSNR*. La table IV.14 et la figure IV.22 récapitulent les résultats de cette expérience. Une fois aussi, on remarque que les résultats du *scan adaptatif* sont meilleurs que ceux du *scan Zigzag* seul pour toutes les valeurs du *PSNR*. Cela permet de tirer que le *scan adaptatif* est totalement indépendant de la valeur du *PSNR* de l'image reconstruite.

Expérience 3 :

Suite aux deux expériences précédentes, des questions viennent à l'esprit :

1. *Y a-t-il un scan important ou meilleur ?*
2. *Quel est le scan le plus présent ?*
3. *Quelle est la contribution des différents scans ?*

Pour commencer, on définit la notion de *scan décisif* qui peut aider à répondre à la *première question*. Un scan est considéré comme *décisif* s'il est le seul scan qui donne le meilleur résultat. Dans le cas où plusieurs scans donnent le même meilleur résultat, ces scans ne seront pas considérés comme *décisifs*. L'importance d'un scan dans notre codeur est synonyme du nombre de fois où il est considéré comme *décisif*.

TABLE IV.13 – Performances de la compression de chaque image de test dans l'espace YCbCr avec la taille du bloc 16 × 16 et $Q = 8$ pour le *scan Zigzag* seul et le *scan adaptatif*.

	Scan	**Zigzag**		**Adaptatif**	
Image	**PSNR**	**bpp**	**CR**	**bpp**	**CR**
Airplane	31,12	0,56	42,92	0,51	46,62
Peppers	31,19	0,93	25,76	0,88	27,31
Lena	32,65	0,78	30,76	0,74	32,43
Girl	35,87	0,40	60,51	0,38	63,58
Couple	32,85	0,87	27,51	0,82	29,25
House	32,08	0,86	27,81	0,77	31,26
Zelda	32,01	0,86	27,77	0,82	29,15
Moyenne	**32,54**	**0,75**	**34,72**	**0,70**	**37,08**

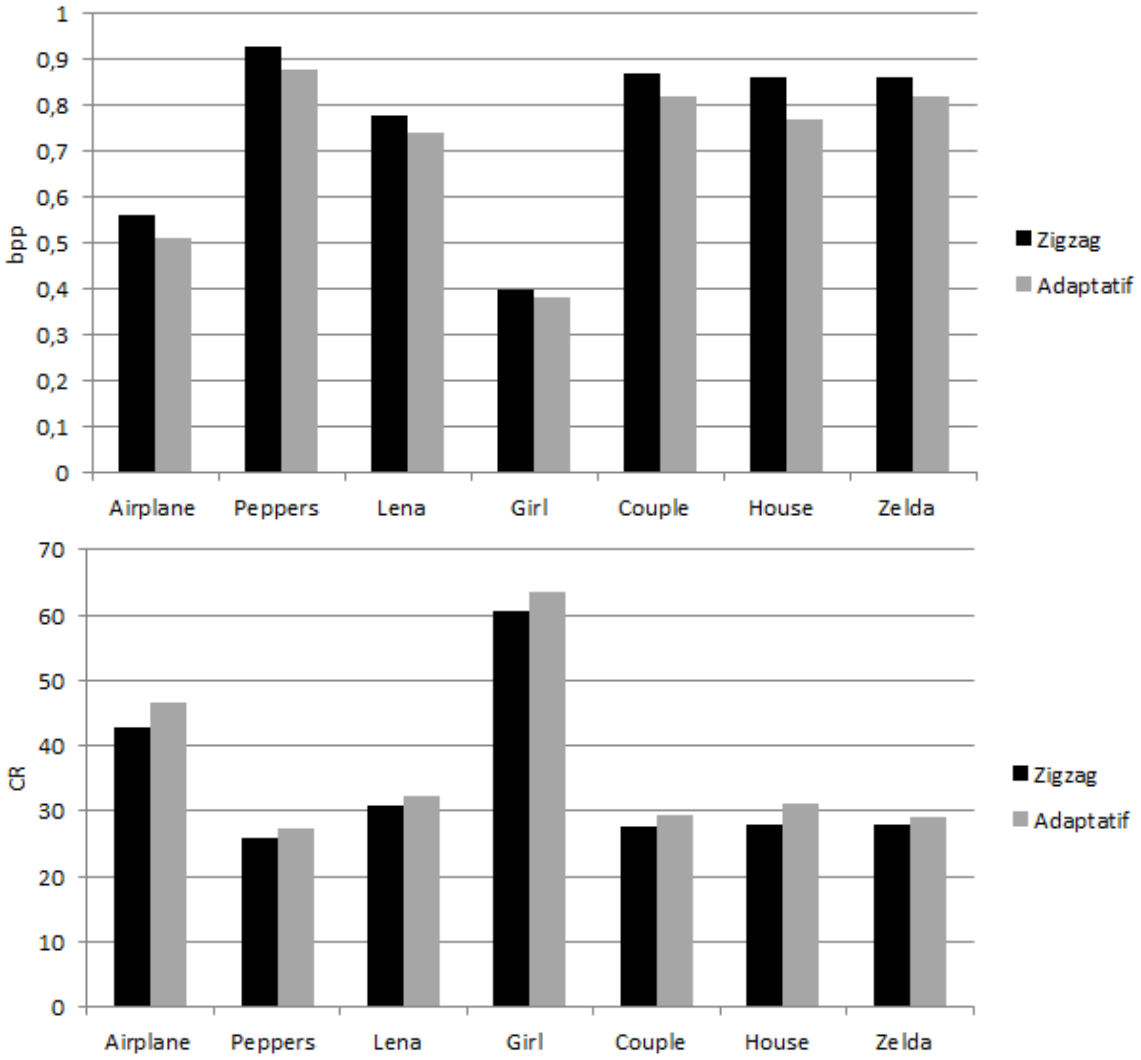


FIGURE IV.21 – Comparaison des performances de la compression de chaque image de test dans l'espace YCbCr avec la taille du bloc 16 × 16 et *Q=8* pour le *scan Zigzag* seul et le *scan adaptatif* (table IV.13).

TABLE IV.14 – Performances de la compression de l'ensemble des images de test dans l'espace YCbCr pour différentes valeurs du *PSNR* avec la taille du bloc 16×16 et Q=8 avec le *scan Zigzag* seul et le *scan adaptatif*

	Scan	**Zigzag**		**Adaptatif**	
	PSNR	**bpp**	**CR**	**bpp**	**CR**
	30,34	0,50	52,45	0,43	58,91
	31,08	0,57	45,24	0,51	50,73
	31,80	0,66	38,98	0,59	43,68
	32,46	0,76	34,13	0,68	38,16
	33,02	0,86	30,10	0,77	33,63
	33,71	1,03	25,21	0,93	28,09
	34,27	1,21	21,65	1,09	24,00
	34,93	1,44	18,27	1,31	20,17
Moyenne	**32,70**	**0,88**	**33,25**	**0,79**	**37,17**

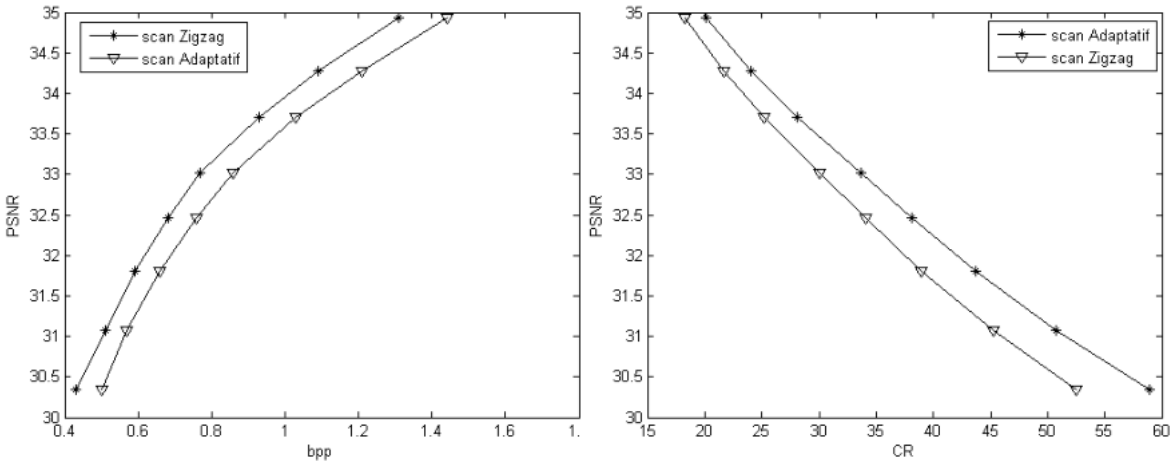


FIGURE IV.22 – Comparaison des performances de la compression de l'ensemble des images de test dans l'espace YCbCr avec la taille du bloc 16×16 et Q=*8* pour le *scan Zigzag* seul et le *scan adaptatif* pour différentes valeurs du *PSNR* (table IV.14).

Pour éclaircir les idées prenant comme exemple les 2 cas suivants (exemple réduit de la table IV.2) :

Cas 1 :

	Zigzag	*Horizontal*	*Horizontal*	*Hilbert*
Max *dLUT*	14	5	26	31
q	4	**3**	5	5

Dans ce cas le *scan HZ* est considéré comme *décisif* puisqu'il est le seul scan qui donne le meilleur résultat (minimum *q*, q=3).

Cas 2 :

	Zigzag	*Horizontal*	*Horizontal*	*Hilbert*
Max *dLUT*	14	12	26	31
q	**4**	**4**	5	5

Dans ce cas, les scans *Zigzag* et *Horizontal* donne donnent le meilleur résultat (minimum q, q=4). Ses deux scans ne sont considérés comme *décisifs*.

La table IV.15 et la figure IV.23, nous fournissent les pourcentages du *scan décisif* correspondants aux résultats de la table IV.14 pour chaque plan.

La réponse à *deuxième question* est simple, en comptant le nombre de fois où un scan donne le meilleur résultat (pas forcément le seul, *cas 2* de l'exemple précédent), on retrouve les pourcentages de la *présence* des différents scans dans le processus de compression. La table IV.16 et la figure IV.24 donnent une vue sur les pourcentages de *présence* des différents scans correspondants aux résultats de la table IV.14 pour chaque plan.

La réponse à la *dernière question* est intuitive en prenant simplement les rapports entre les *présences* des différents scans on peut retrouver la contribution de chaque scan dans le processus de compression. Cela est reporté à la table IV.17. La figure IV.25 illustre en claire la répartition de *présence* des différents scans correspondants aux résultats de la table IV.14 pour chaque plan.

Les données de la table IV.17 peuvent se voir d'un autre coté pour montrer l'avantage acquis par le *scan adaptatif* sur le *scan Zigzag* seul. La figure IV.26 illustre ce point de vue.

Conclusion :
Deux remarques peuvent être tirées de ce test :

1. Le *scan adaptatif* apporte certes une amélioration sur le *scan Zigzag* seul.
2. Dans le cas où on utilise un seul scan (par exemple pour réduire le temps de calcul) le *scan Zigzag* est de loin le scan à retenir puisqu'il est le scan le plus présent indépendamment de la nature de l'image et du *PSNR* exigé. Cette remarque confirme le choix du *scan Zigzag* pour la norme JPEG.

IV.4.5 Test 5 : Effet du codage séparé des vecteurs *Header*, *NN* et *dLUT*.

Comme déjà mentionné, la nature des vecteurs résultants du codeur (*Header*, *NN* et *dLUT*) est très différente. Un test sur l'intérêt de coder ces vecteurs séparément serait bien justifié. Dans cette optique, le test présent met en évidence le choix d'utiliser ou non un *codage séparé* à la sortie du codeur proposé. En effet, les intervalles de variations des différents vecteurs résultants sont donnés par :

TABLE IV.15 – Les pourcentages du scan le plus *décisif* de l'ensemble des images de test (table IV.14).

Espace	*Y*				*Cb*				*Cr*			
Scan	ZG	HZ	VC	HB	ZG	HZ	VC	HB	ZG	HZ	VC	HB
	39,71	24,86	19,71	16,00	36,83	24,50	18,83	20,17	31,71	28,00	19,43	21,00
	40,43	25,29	21,86	12,57	30,00	37,14	17,00	15,71	33,29	24,57	19,14	23,14
	39,86	24,86	23,29	11,71	29,29	36,43	15,71	18,71	35,43	26,14	16,86	21,86
	36,71	28,14	25,57	9,86	30,86	32,00	14,00	23,29	37,71	26,71	18,00	17,71
	33,86	28,29	27,43	9,86	34,14	28,29	13,71	23,86	36,86	26,43	19,43	17,14
	32,29	28,86	29,29	9,57	35,00	28,71	16,43	20,00	36,00	28,14	20,43	15,57
	31,14	30,00	30,14	8,71	31,43	29,43	18,43	20,71	35,57	27,86	23,29	13,57
	32,43	29,57	30,29	8,00	30,57	32,43	20,14	16,57	32,71	27,86	24,29	15,43
Moyenne	**35,80**	**27,48**	**25,95**	**10,79**	**32,26**	**31,12**	**16,78**	**19,88**	**34,91**	**26,96**	**20,11**	**18,18**

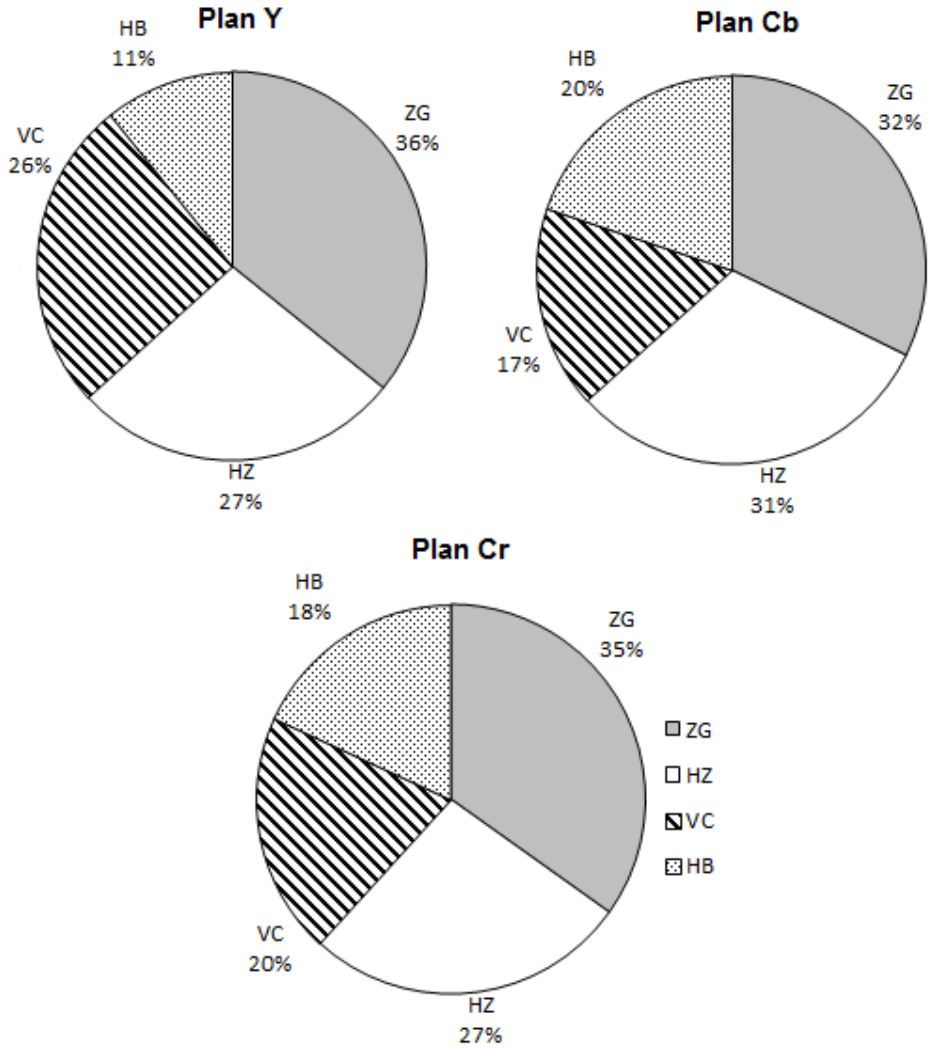


FIGURE IV.23 – La répartition du scan le plus *décisif* de l'ensemble des images de test (table IV.15).

TABLE IV.16 – Les pourcentages de la contribution de la *présence* de chaque scan de l'ensemble des images de test (table IV.14)

Espace	*Y*				*Cb*				*Cr*			
Scan	ZG	HZ	VC	HB	ZG	HZ	VC	HB	ZG	HZ	VC	HB
	32,00	25,00	21,29	21,86	26,14	24,71	24,00	25,00	26,86	24,14	23,71	24,86
	32,00	24,71	22,86	20,57	26,57	24,57	23,86	24,71	27,71	24,14	23,29	24,71
	32,43	25,00	23,43	19,00	27,43	24,29	23,29	24,86	28,14	24,14	23,00	24,29
	32,14	25,86	24,43	17,57	27,86	24,43	23,14	24,86	28,71	24,29	23,00	24,00
	31,43	26,29	25,14	16,86	28,43	24,71	22,57	24,43	28,57	24,29	23,00	24,00
	30,43	27,00	26,43	16,00	29,29	25,00	22,14	23,71	28,86	25,14	23,00	23,14
	29,86	27,29	27,14	15,57	29,14	25,29	22,43	23,14	28,71	25,14	23,86	22,43
	30,29	27,29	27,29	15,29	28,43	25,57	23,00	23,14	28,00	25,29	24,29	22,00
Moyenne	**31,32**	**26,05**	**24,75**	**17,84**	**27,91**	**24,82**	**23,05**	**24,23**	**28,20**	**24,57**	**23,39**	**23,68**

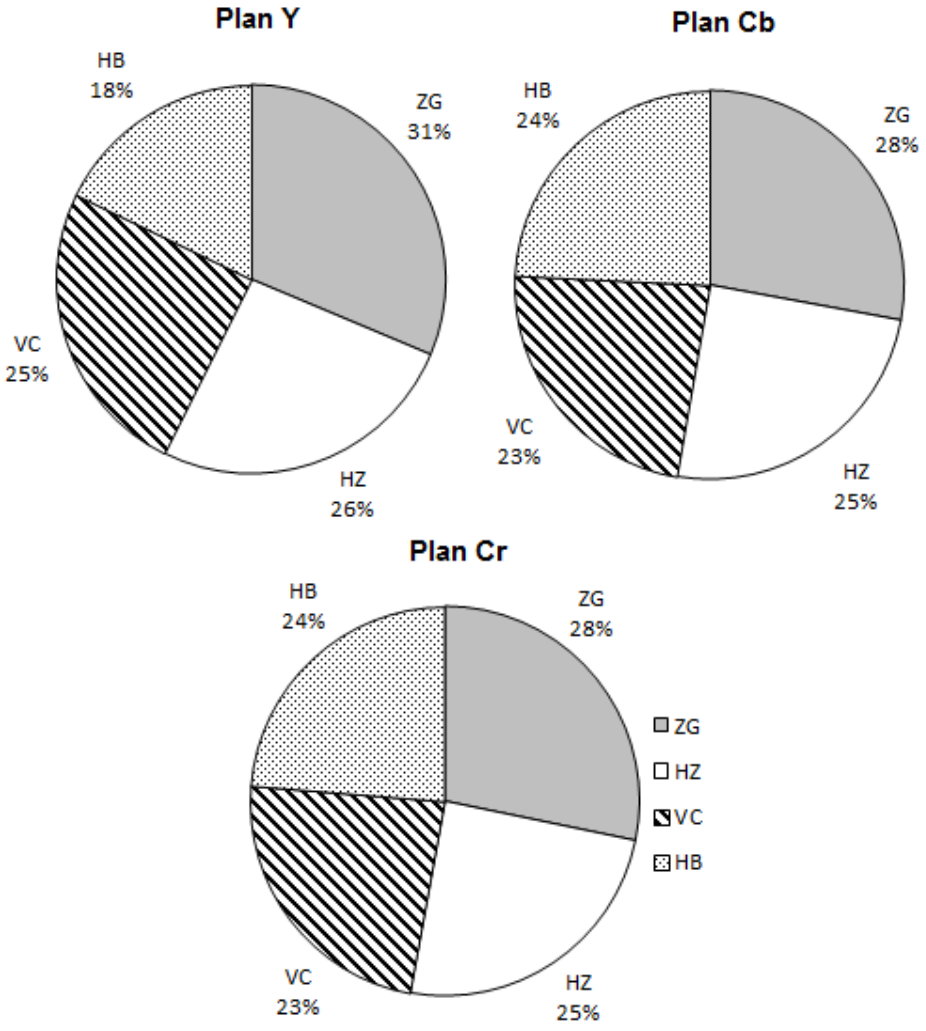


FIGURE IV.24 – La répartition de la *présence* des différents scans de l'ensemble des images de test (table IV.16).

TABLE IV.17 – Les pourcentages de la *présence* des scans de l'ensemble des images de test (table IV.14).

Espace	*Y*				*Cb*				*Cr*			
Scan	ZG	HZ	VC	HB	ZG	HZ	VC	HB	ZG	HZ	VC	HB
	70,14	55,57	49,57	49,71	84,57	80,00	78,29	80,57	82,57	75,86	73,57	77,14
	67,29	52,71	49,86	45,29	82,00	76,57	74,43	76,71	81,14	72,14	69,71	73,43
	65,71	51,14	49,29	40,29	80,43	72,86	70,00	73,86	79,43	69,14	65,71	70,14
	63,43	51,57	50,14	37,14	77,57	68,57	65,14	70,29	78,29	66,71	63,71	66,71
	61,00	51,57	50,14	34,86	75,71	66,29	60,71	65,86	74,86	64,29	61,29	63,57
	58,29	52,00	51,57	32,57	73,86	63,43	57,86	60,86	72,14	62,71	58,86	59,00
	57,00	52,14	52,43	31,57	70,29	60,71	55,43	57,00	68,86	59,71	57,43	54,43
	58,29	52,86	53,14	31,29	65,14	58,71	54,00	53,86	65,71	59,29	56,86	52,43
Moyenne	**62,64**	**52,45**	**50,77**	**37,84**	**76,20**	**68,39**	**64,48**	**67,38**	**75,38**	**66,23**	**63,39**	**64,61**

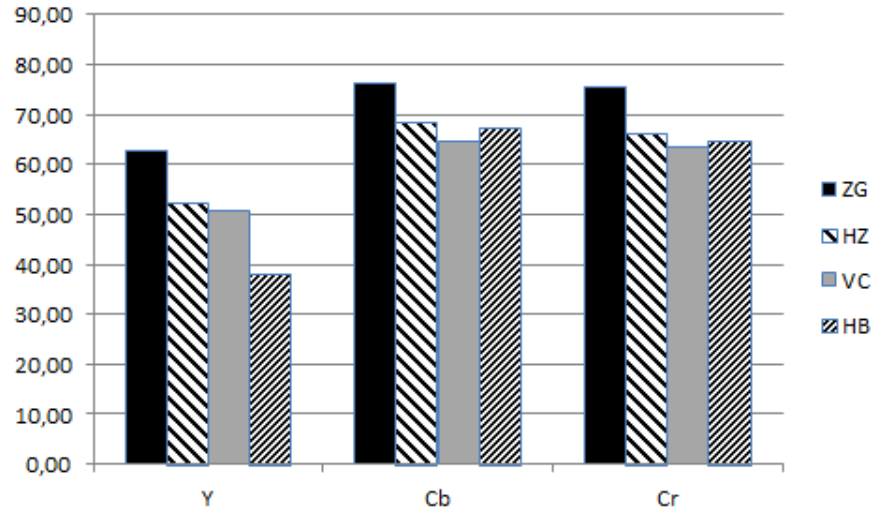


FIGURE IV.25 – Histogramme de la *présence* des différents scans de l'ensemble des images de test (table IV.17).

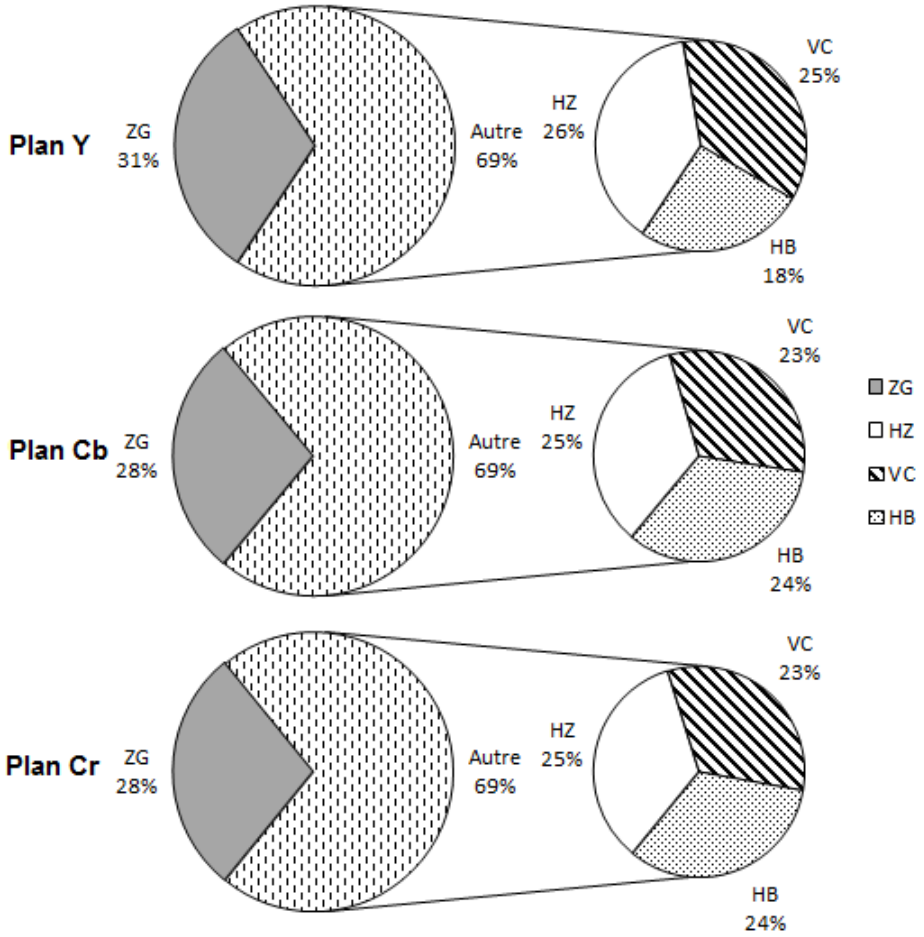


FIGURE IV.26 – Comparaison entre la *présence* du scan *Zigzag* et les autres scans des données de la table IV.16.

- Le vecteur *Header* : [0-64][0-3][1-7],
- Le vecteur *NN* : [1-255],
- Le vecteur *dLUT* : [1-63].

Expérience 1 :

La table IV.18 récapitule les performances obtenues pour chaque image dans l'espace YCbCr pour taille de bloc 8×8 et 16×16. On remarque aisément le gain acquis par le *codage séparé* des vecteurs résultants du codeur. Sur la figure IV.27 se voit clairement l'intérêt d'un tel *codage séparé*.

Expérience 2 :

Un essai sur la totalité des images pour différentes valeurs du *PSNR* ; dont les résultats sont reportés à la table IV.19 ; confirme d'une façon très claire que le *codage séparé* des vecteurs résultants du codeur apporte une amélioration incontestable en termes de

pbb et *CR*. Cette amélioration est en moyenne de **33,08%** pour le *CR* et de **23,30%** concernant le *bpp* pour le même *PSNR*. Cela est traduit sur la figure IV.28 par une courbe de *codage séparé* nettement au-dessus de celle du *codage non séparé*.

Conclusion

Les expériences de ce test nous permettent de dire clairement que le *codage séparé* des vecteurs *Header*, *NN* et *dLUT* apporte un avantage efficace indépendamment du *PSNR* et de la nature des images.

TABLE IV.18 – Performances de la compression de chaque image de test dans l'espace YCbCr avec différentes tailles du bloc et pour différentes valeurs de Q *avec* et *sans codage séparé* des vecteurs *Header*, *NN* et *dLUT*.

Taille du bloc		8×8					16×16				
Codage			**Non Séparé**		**Séparé**			**Non Séparé**		**Séparé**	
	Image	**PSNR**	**bpp**	**CR**	**bpp**	**CR**	**PSNR**	**bpp**	**CR**	**bpp**	**CR**
Q=7	Airplane	31,14	1,07	22,51	0,66	36,58	31,16	0,67	35,84	0,48	49,79
	Peppers	31,18	1,51	15,86	1,04	23,17	31,12	1,25	19,16	0,97	24,66
	Lena	32,78	1,47	16,33	1,01	23,78	32,83	1,20	20,03	0,90	26,68
	Girl	35,88	0,81	29,72	0,45	53,93	35,89	0,52	46,54	0,38	63,11
	Couple	32,90	1,37	17,53	0,93	25,76	32,80	1,05	22,81	0,82	29,45
	House	32,14	1,34	17,87	0,90	26,53	32,11	1,00	24,01	0,78	30,79
	Zelda	32,08	1,45	16,51	0,99	24,21	31,94	1,11	21,58	0,86	27,93
	Moyenne	**32,59**	**1,29**	**19.48**	**0,85**	**30,56**	**32,55**	**0,97**	**27,14**	**0,74**	**36,06**
Q=8	Airplane	31,17	1,16	20,72	0,74	32,23	31,12	0,70	34,23	0,51	46,62
	Peppers	31,20	1,61	14,94	1,13	21,30	31,19	1,14	20,98	0,88	27,31
	Lena	32,70	1,48	16,19	1,03	23,29	32,65	0,97	24,75	0,74	32,43
	Girl	35,90	0,89	27,01	0,51	46,65	35,87	0,51	46,85	0,38	63,58
	Couple	32,84	1,48	16,27	1,02	23,44	32,85	1,04	22,98	0,82	29,25
	House	32,16	1,47	16,29	1,02	23,64	32,08	0,98	24,45	0,77	31,26
	Zelda	32,03	1,58	15,21	1,11	21,69	32,01	1,07	22,46	0,82	29,15
	Moyenne	**32,57**	**1,38**	**18,09**	**0,94**	**27,46**	**32,54**	**0,92**	**28,10**	**0,70**	**37,08**
Q=9	Airplane	31,04	1,26	19,09	0,84	28,67	31,08	0,77	31,36	0,57	41,90
	Peppers	31,24	1,77	13,53	1,28	18,77	31,20	1,24	19,40	0,97	24,84
	Lena	32,76	1,63	14,71	1,17	20,57	32,65	1,04	23,00	0,81	29,63
	Girl	35,93	1,02	23,64	0,62	38,64	36,00	0,59	40,71	0,44	54,40
	Couple	32,88	1,67	14,39	1,19	20,18	32,98	1,21	19,85	0,96	25,04
	House	32,20	1,68	14,28	1,19	20,15	32,02	1,10	21,75	0,86	27,78
	Zelda	32,07	1,79	13,43	1,28	18,68	32,11	1,23	19,50	0,96	24,94
	Moyenne	**32,59**	**1,54**	**16,15**	**1,08**	**23,67**	**32,58**	**1,03**	**25,08**	**0,80**	**32,65**

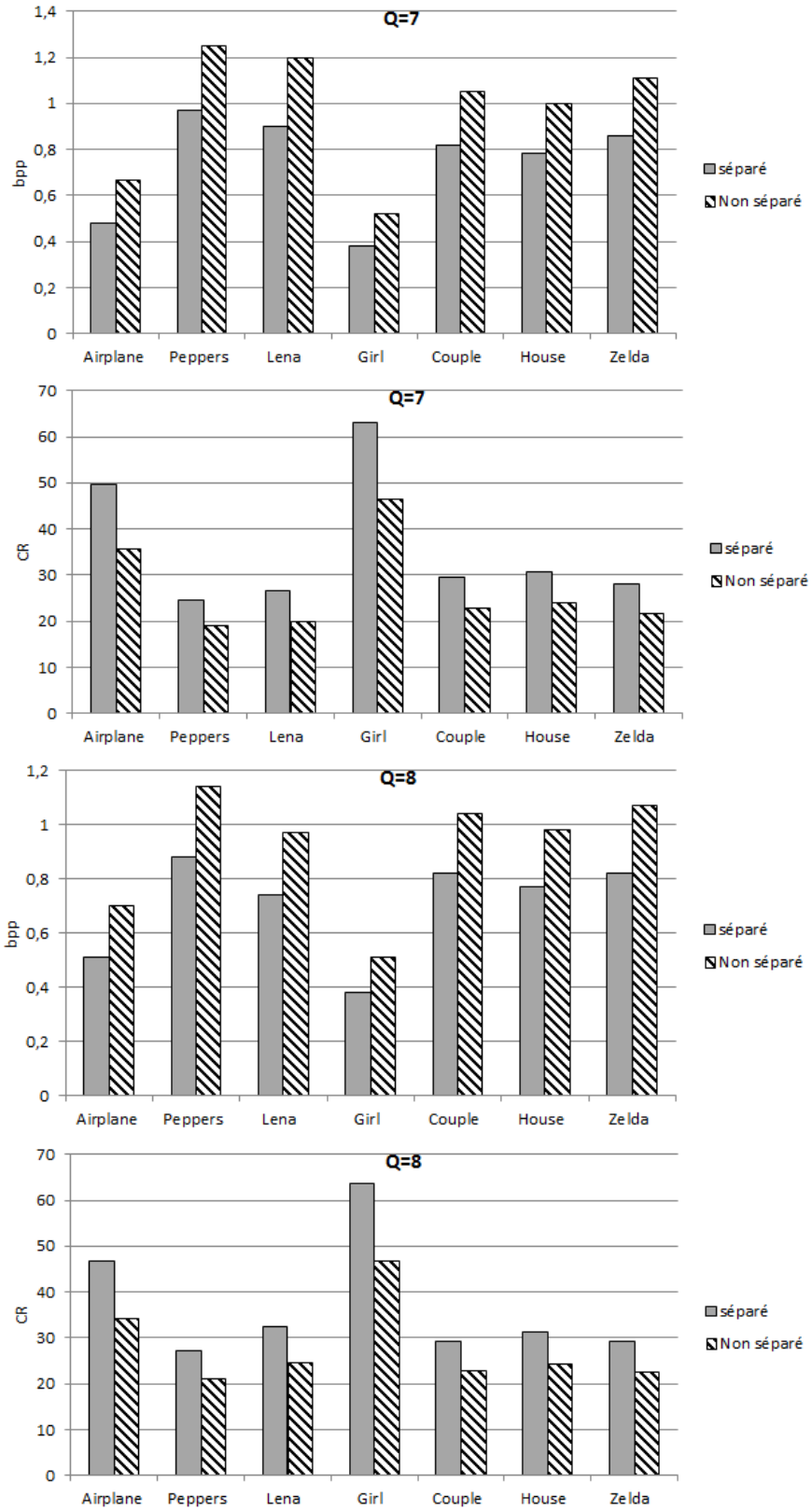


FIGURE IV.27 – Comparaison des performances de la compression de chaque image de test dans l'espace YCbCr pour la taille du bloc 16×16 *avec* et *sans codage séparé* des vecteurs *Header*, *NN* et *dLUT* pour Q=7 et Q=8 (table IV.18).

TABLE IV.19 – Performances de la compression de l'ensemble des images de test dans l'espace YCbCr pour différentes valeurs du *PSNR* avec la taille du bloc 16×16 et Q=8 *avec* et *sans codage séparé* des vecteurs *Header*, *NN* et *dLUT*.

Codage	**Non Séparé**		**Séparé**	
PSNR	**bpp**	**CR**	**bpp**	**CR**
30,34	0,58	43,14	0,43	58,91
31,08	0,67	37,64	0,51	50,73
31,80	0,78	32,71	0,59	43,68
32,46	0,89	28,86	0,68	38,16
33,02	1,00	25,59	0,77	33,63
33,71	1,20	21,52	0,93	28,09
34,27	1,40	18,47	1,09	24,00
34,93	1,69	15,52	1,31	20,17
Moyenne 32,70	**1,03**	**27,93**	**0,79**	**37,17**

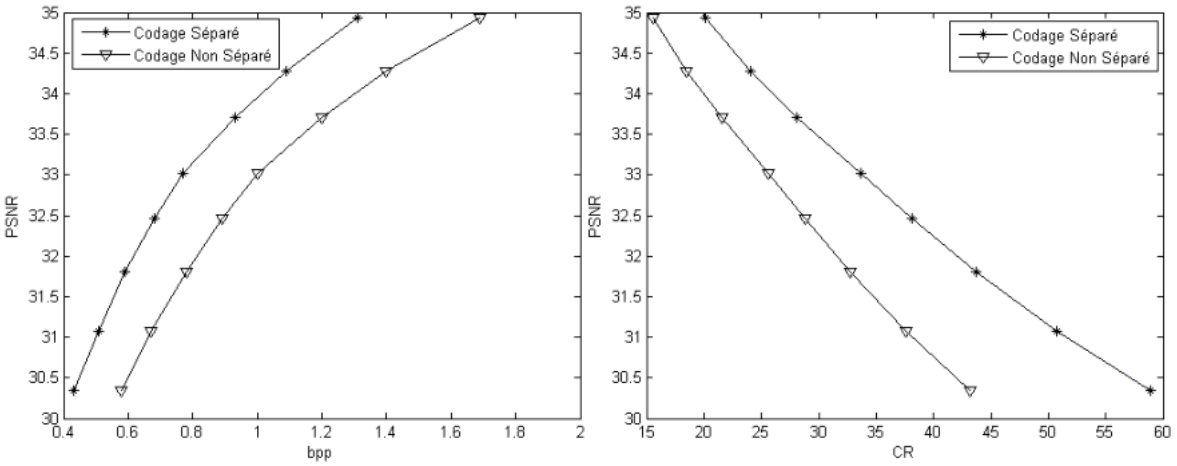


FIGURE IV.28 – Comparaison des performances de la compression de l'ensemble des images de test dans l'espace YCbCr pour Q=*8* et la taille du bloc 16×16 *avec* et *sans codage séparé* des vecteurs *Header*, *NN* et *dLUT* (table IV.19).

IV.4.5.1 Temps de calcul

Pour évaluer la performance de notre codeur en matière de *temps de calcul*, nous avons récapitulé aux tables IV.20 et IV.21 les *temps de codage* et de *décodage* pour les différentes images de test et pour différents *PSNR*. On remarque clairement que le temps de décodage est nettement inférieur à celui du codage qui contient une étape de recherche du *scan adaptatif*. Dans le cas d'un codage à *scan Zigzag* seul les *temps de codage* et de *décodage* serons très proches.

TABLE IV.20 – Temps de codage et de décodage en secondes de chaque image de test (sur une machine *i3 234M 2.30GHz RAM=4Go*).

Image	Taille	PSNR	Temps de codage	Temps de décodage
Airplane	262144	31,1160	3,7566	0,1722
Peppers	262144	31,1876	4,8564	0,1734
Lena	262144	32,6541	4,2694	0,1695
Girl	262144	35,4590	0,6682	0,0403
Couple	65536	32,8510	1,0706	0,0428
House	65536	32,0774	0,9859	0,0425
Zelda	65536	32,0118	1,0568	0,0423

TABLE IV.21 – Temps de codage et de décodage pour différentes valeurs du *PSNR* de l'image *Lena* (sur une machine *i3 234M 2.30GHz RAM=4Go*).

PSNR	Temps de codage	Temps de décodage
29,0581	3,8930	0,1875
29,4818	5,7698	0,2117
29,9688	2,7380	0,1582
30,5943	2,9688	0,1581
30,9618	3,1558	0,1611
31,6802	3,5210	0,1622
31,9801	3,7106	0,1654
32,3017	3,9489	0,1661

IV.4.6 Comparaison avec d'autres travaux existants

Nous souhaitons à présent faire une comparaison de notre approche vis-à-vis différents algorithmes existants. Afin de permettre une comparaison judicieuse nous avons reporté les différents résultats sur plusieurs tables avec des *PSNR* les plus proches de chaque l'algorithme visé.
Pour être juste dans la comparaison, il faut que les *PSNR* de notre méthode soient les plus proches des *PSNR* correspondants et cela pour chaque image de test. La table IV.22 regroupe les résultats de comparaison avec les travaux CDABS [DOU11] et CBTC-PF [DHA07] vérifiant cette contrainte pour chaque image.

On remarque la supériorité de notre méthode [MES16] par rapport aux autres méthodes. La moyenne sur l'ensemble des images affirme cette supériorité.

TABLE IV.22 – Comparaison de la méthode proposée avec différents algorithmes.

	Méthode proposée [MES16]		CDABS [DOU11]		CBTC-PF [BIP07]	
Image	**PSNR**	**bpp**	**PSNR**	**bpp**	**PSNR**	**bpp**
Airplane	30,96	0,50	30,38	0,59	30,36	1,04
Peppers	30,11	0,64	30,06	0,81	30,15	1,50
Lena	31,68	0,57	31,97	0,81	31,93	1,17
Girl	35,08	0,32	35,00	0,46	35,13	0,60
Couple	31,82	0,65	32,29	0,93	32,44	1,00
House	31,70	0,72	31,73	0,82	31,79	1,20
Zelda	31,28	0,72	31,33	0,87	31,31	1,12
Moyenne	**31,80**	**0,59**	**31,82**	**0,76**	**31,87**	**1,09**

Pour d'autres gammes du *PSNR* effectuées par les travaux de CDABS [DOU11] et JPEG [DHA07] (table IV.23), notre approche domine et montre aussi son avantage.

TABLE IV.23 – Comparaison de la méthode proposée avec différents algorithmes.

	Méthode proposée [MES16]		CDABS [DOU11]		CBTC-PF [BIP07]	
Image	**PSNR**	**bpp**	**PSNR**	**bpp**	**PSNR**	**bpp**
Airplane	31,16	0,48	31,40	0,72	31,46	0,90
Peppers	31,19	0,88	30,33	0,88	30,47	1,47
Lena	32,65	0,74	32,77	1,00	32,76	1,03
Girl	35,86	0,38	36,96	0,69	36,85	0,62
Couple	32,62	0,79	33,07	1,13	33,02	0,94
House	32,27	0,79	31,32	0,75	31,34	1,24
Zelda	32,01	0,82	32,05	1,09	32,06	1,00
Moyenne	**32,54**	**0,70**	**32,56**	**0,89**	**32,57**	**1,03**

Une comparaison avec un algorithme GA-DWT [BOU12] qui utilise la transformée en ondelettes discrètes reportée à la table IV.24. Il est à remarquer que les performances notre approche sont très collées aux performances de la méthode GA-DWT [BOU12] qui utilise la DWT en combinaison avec un algorithme génétique pour chercher une conversion colorimétrique autre que l'espace YCbCr adapté à chaque image. Cela est fort pénalisant en matière de temps de compression et rend la méthode dépendante de la nature de l'image à compresser. Ce qui met notre algorithme à un pas en avance par rapport à cette méthode.

Pour un même *bpp*, notre méthode permet d'avoir des images reconstruites de meilleures qualités que celles proposées par le standard JPEG [DHA07] (table IV.25), CBTC-PF [DHA07] (table IV.26) et CDABS [DOU11] (table IV.27). Cela affirme la dominance de la présente approche sur ces méthodes.

TABLE IV.24 – Comparaison de la méthode proposée avec la méthode GA-DWT.

	Méthode proposée [MES16]		GA-DWT [BOU12]	
Image	**PSNR**	**bpp**	**PSNR**	**bpp**
Airplane	31,16	0,48	31,16	0,49
Peppers	31,19	0,88	31,20	0,83
Lena	32,65	0,74	32,76	0,66
Girl	35,86	0,38	35,90	0,41
Couple	32,62	0,79	32,87	0,89
House	32,27	0,79	32,10	0,83
Zelda	32,01	0,82	31,98	0,76
Moyenne	**32,54**	**0,70**	**32,57**	**0,70**

TABLE IV.25 – Comparaison de la méthode proposée avec l'algorithme JPEG [DHA07].

	Méthode proposée [MES16]		JPEG [BIP07]	
Image	**PSNR**	**bpp**	**PSNR**	**bpp**
Airplane	34,01	0,90	31,46	0,90
Peppers	31,72	1,11	30,47	1,47
Lena	33,68	1,03	32,76	1,03
Girl	37,60	0,56	36,85	0,62
Couple	34,67	1,23	33,02	0,94
House	34,15	1,23	31,34	1,24
Zelda	33,08	1,14	32,06	1,00
Moyenne	**34,13**	**1,03**	**32,57**	**1,03**

TABLE IV.26 – Comparaison de la méthode proposée avec l'algorithme CBTC-PF.

	Méthode proposée [MES16]		CBTC-PF [BIP07]	
Image	**PSNR**	**bpp**	**PSNR**	**bpp**
Airplane	34,01	0,90	30,36	1,04
Peppers	32,18	1,32	30,15	1,50
Lena	33,68	1,03	31,93	1,17
Girl	37,60	0,56	35,13	0,60
Couple	34,67	1,23	32,44	1,00
House	34,15	1,23	31,79	1,20
Zelda	33,62	1,36	31,31	1,12
Moyenne	**34,27**	**1,09**	**31,87**	**1,09**

TABLE IV.27 – Comparaison de la méthode proposée avec l'algorithme CDABS.

	Méthode proposée [MES16]		CDABS [DOU11]	
Image	**PSNR**	**bpp**	**PSNR**	**bpp**
Airplane	33,30	0,76	31,40	0,72
Peppers	31,72	1,11	30,33	0,88
Lena	33,22	0,88	32,77	1,00
Girl	37,03	0,48	36,96	0,69
Couple	33,98	1,06	33,07	1,13
House	33,62	1,06	31,32	0,75
Zelda	32,42	0,91	32,05	1.09
Moyenne	**33,61**	**0,89**	**32,56**	**0,89**

IV.5 Conclusion

Ce chapitre a fait l'objet de l'étude de la technique élaborée pour la compression d'images couleurs ainsi que les tests de simulation numériques visant à confirmer les résultats théoriques. De ce fait, nous avons étudié en détails les différentes étapes de la technique proposée. Cette dernière est composée de plusieurs blocs :

- Une conversion d'espace de couleur RGB vers YCbCr s'avère nécessaire pour avoir des taux de compression élevés cela est justifié par les résultats du *Test 1* effectué sur l'ensemble d'images de test.
- La transformation choisie est la DCT vue sa simplicité et son efficacité, le couplage de notre codeur avec la transformation DCT a donné des résultats très proches de celle utilisant une transformation en ondelettes en association avec un algorithme génétique de [BOU12] mais avec une simplicité claire et *un temps de calcul* nettement plus faible.
- Les étapes de *seuillage* et *quantification* permettent d'arrondir et d'éliminer les coefficients de faibles valeurs pour être encoder efficacement par la suite.
- Un codage *sans pertes* des coefficients issus des étapes précédentes.

Le codeur proposé est basé sur l'utilisation d'une table de différence d'indices $dLUT$ pour enregistrer les indices (adresses) des coefficients *non nuls* de chaque bloc. L'usage de la différence d'indices permet de réduire efficacement la taille nécessaire pour encoder le bloc des coefficients. Les indices des coefficients sont liés à la méthode de lecture du bloc ou ce qu'on appelle habituellement le scan.

Le nombre de bits du codage de la table $dLUT$ est déterminé par le *scan adaptatif*. Le meilleur scan pour chaque bloc est ensuite codé dans l'*entête* du bloc pour être utilisé ultérieurement au niveau du décodeur. En plus du *Header*, la séquence de

binaire de chaque bloc contient les coefficients *non nuls NN* suivis par le vecteur $dLUT$.

Les valeurs des vecteurs de *Headers*, de NN et de $dLUT$ sont très différentes pour cela, on a vu utile de les codés séparément. Les résultats du test dédié au *codage séparé* (*Test 5*) illustre bien ce choix. Finalement, ces vecteurs sont compressés *sans pertes* avec un encodeur *arithmétique*.

Les figures et tables obtenues pour les différents tests et diverses expériences montrent bien l'intérêt et l'importance de chaque partie de la technique proposée. La dernière section par des tables comparatives illustre les résultats de comparaison avec d'autres méthodes existantes et montre que notre méthode donne de meilleurs résultats en la comparant avec celles-ci.

Conclusion Générale et Perspectives

Le volume des données échangées circulant sur Internet ou via mobile, déjà impressionnant, continuera de croitre dans les années à venir du fait du développement du multimédia et des applications interactives et des systèmes communications de plus en plus performants. Les images et les vidéos constituent la part de lion de ces données échangées.

La compression, née du développement des techniques de numérisation et du multimédia, consiste à coder l'ensemble de ces données sur un nombre réduit de bits. La nécessité de compresser les images apparaît donc aujourd'hui incontournable pour remplir les fonctionnalités d'archivage et de transmission rapide. En effet, il serait illusoire de croire que la compression devient obsolète à cause de la capacité des mémoires qui augmente sans cesse, car la quantité de données à conserver croît de la même manière, et les réseaux évoluent pour leur part beaucoup moins vite.

Le schéma de codage d'un signal est en général constitué d'une *mise en forme* du signal ; afin de l'adapter à la compression ; d'une étape de *quantification* et d'une étape de codage. L'étape de la *mise en forme* se justifie par le fait que certains signaux, tels que les images, sont pour la plupart des cas fortement corrélés. Afin de comprimer efficacement les images, il est impératif de prendre en compte ce phénomène. L'information redondante est éliminée alors que l'information la plus essentielle à l'obtention d'une image finale de bonne qualité est préservée.

Nous avons abordé dans cette thèse un point important en traitement de signaux, celui de la compression des images. Les limites des méthodes de compression existantes sont sans cesse repoussées et le challenge consiste à en imposer de nouvelles. Diverses méthodes de compression ont été élaborées par la communauté scientifique. En dépit du grand nombre de succès qu'il a pu rencontrer, le standard de compression d'images JPEG est loin de nous donner des performances et des résultats infranchissables.

L'objectif souhaité dans cette étude est d'élaborer de nouvelles techniques de compression d'images offrant des performances en compression meilleurs. Les signaux ciblés par cette thèse ; qui sont les images naturelles (*non synthétiques*) de *personnes*, *paysage*,... fixes couleurs à usage général ; ont été présentés dans le *premier chapitre*. En plus, la physiologie de la vision humaine des couleurs a été introduite. En essayant d'expliquer les mécanismes qui régissent la vision humaine des couleurs, plusieurs modèles de couleur ont été proposés. En réalité, il existe un éventail très large de systèmes de couleur. Ces systèmes sont développés par différents organismes et selon telles ou telles contraintes technologiques. Ces divers modèles conjugués aux contraintes technologiques, liées à certaines industries, ont conduit à la coexistence d'un grand nombre de systèmes de couleur dans la littérature.

Les différents travaux menés pour le choix d'un espace de couleur universel, qui soit le mieux approprié pour le traitement des images en couleurs, ne convergent pas vers un seul système. Il est difficile à priori de déterminer un espace de couleur qui soit le mieux adapté pour la compression des images couleurs. Le choix d'un système de couleur est, avant tout, fonction de l'application abordée et le type des images à traiter. Néanmoins, plusieurs études convergent vers le choix des systèmes de couleur décorrélés, dissociant l'information *chromatique* de l'information *achromatique* tel que le modèle YCbCr pour la compression des images en couleurs.

Dans le *deuxième chapitre*, nous avons abordé la problématique de la compression d'images fixes *avec* ou *sans pertes*. En effet, la compression peut s'effectuer à l'aide des techniques ne concédant aucune perte d'informations ou reposant sur l'estimation que certaine détails difficilement perceptibles par l'usager, peuvent délibérément être omis. Ainsi se différencient deux grandes catégories de techniques dites *sans perte* et *avec pertes*. La compression *avec pertes* assure la reconstitution intégrale de l'image originale suite au processus de décompression. Ce type de compression exploite uniquement les redondances dans l'image ce qui ne permettait pas une réduction significative du volume de l'image.

La compression *avec pertes* est définie de manière à mettre en œuvre une compression efficace, tout en contrôlant la dégradation de la qualité de l'image reconstruite. Ainsi, elle cherche à garantir un meilleur compromis débit-distorsion. Une telle technique de compression est nécessaire pour des applications requérant des taux de compression élevés.

Le fait que l'information contenue dans une image est distribuée sur toute la matrice image rend sa compression efficace difficile. Par contre dans le domaine fréquentiel,

l'information (qui est strictement équivalente) est généralement plus *concentrée*. La transformation du domaine spatial au domaine fréquentiel se fait par la décomposition de l'image dans une base adéquate de fonctions de telle sorte que les coefficients de la transformation soient indépendants et qu'un nombre minimum de ces coefficients contient une proportion importante de l'énergie de l'image. Les transformations les plus utilisées en compressions d'images sont la *transformée en cosinus discrète* (DCT) et la *transformée en ondelettes* (WT). Cet état de l'art présenté dans ce chapitre a permis de dégager l'orientation principale de notre étude.

Malgré la supériorité théorique et le grand succès de la transformée en ondelettes sur les diverses transformées existantes dans la littérature de la compression, la DCT continue à occuper une place respectée. Plusieurs travaux récents ont utilisés cette transformation au cœur de leurs codeur [ALA17][FAN15][VER15][MA15]. Notre contribution [MES16] utilisant la DCT a donné des performances identique à la méthode [BOU12] utilisant les ondelettes. Pour cette raison, le *troisième chapitre* a été consacré à cette transformation. Les composantes du standard JPEG ont été exposées dans cette même partie de thèse.

Le *dernier chapitre* de cette thèse à fait l'objet d'une étude détaillée de notre technique proposée. En premier lieu, nous avons introduit les critères d'évaluation *PSNR*, *CR* et *bpp* afin de juger la qualité de l'image reconstruite. Le schéma de l'approche proposé se décline en quatre étapes : une étape de *conversion d'espace de couleur* RGB vers YCbCr, suivie de celle de la *transformation*, puis l'étape de *seuillage* et de *quantification* et finalement l'étape du *codage*.

L'évidence de chaque étape a été sujet d'un test constitué de plusieurs expériences. La *première expérience* de chaque test concerne l'application de la méthode proposée sur chaque image de l'ensemble de test avec un *PSNR* choisie dans un intervalle d'utilisation selon la nature de chaque image. Nous visons par ce choix le comportement de notre algorithme vis à vis la nature de chaque image en plus une comparaison avec les autres algorithmes.

Le but de la *deuxième expérience* est de tester la robustesse de notre méthode pour une diversité d'images. Cela est réalisé en prenant la moyenne de toutes les images pour différentes valeurs du *PSNR*. Nous avons appliqué notre méthode sur une image de l'ensemble de test comme une *troisième expérience* pour voir l'effet visuel sur l'image compressée.

Le *premier test* concerne l'intérêt de la conversion colorimétrique RGB-YCbCr dans notre algorithme. Une amélioration en moyenne de **41,26%** en *CR* et **27,08%** en *bpp* (réduction) pour pratiquement le même *PSNR* a été remarquée. Ce qui justifie le choix de cette conversion colorimétrique dans notre algorithme.

Le découpage en blocs pour la transformation DCT est primordial. C'est pour cela que le *deuxième test* étudie l'impact de la taille du bloc de la transformation DCT. Trois valeurs de découpage 8×8, 16×16 et 32×32 ont été testées. Le découpage 16×16 a donné des résultats meilleurs de **35,03%**, **4,51%** pour le *CR* et de **25,53%** et **4,11%** pour le *bpp* par rapport aux découpages 8×8 et 32×32 respectivement. Les expériences de ce test ont montré la supériorité du découpage 16×16 pour chaque image. Ce qui signifie que ce découpage est le mieux adapté indépendamment de la nature de l'image. Cela nous a conduits à adopté cette taille de découpage.

Certainement l'étape de quantification est importante dans schéma de compression. Le *troisième test* vise l'impact du nombre de bit du quantificateur. Plusieurs valeurs de celui-ci ont été testées. Les résultats du choix de Q=8 et taille du bloc 16×16 sont les meilleurs et cela indépendamment de la qualité de l'image reconstruite. Pour pratiquement le même *PSNR*, on a remarqué en moyenne des résultats meilleurs de **2,86%** (Q=7), **13,60%** (Q=9) pour le *CR* et de **5,41%** (Q=7), **12,50%** (Q=9) pour le *bpp*. Cela a permet de dégager que la valeur Q=8 avec un découpage de 16×16 est la meilleur combinaison.

Le codeur proposé s'articule sur plusieurs idées simples mais très efficaces lorsqu'elles sont employées dans une même technique. La *première* d'elles est qu'il est bien connu que la *différence* entre les valeurs d'une suite engendre une suite d'une dynamique plus faible que la suite originale. Le codeur proposé traduit cette idée en utilisant une table nommée *dLUT* contenant la *différence des indices* des coefficients *non nuls* de chaque bloc au lieu de leurs indices.

Comme une *deuxième idée* qui permet de soutenir la *première* est : si les *indices* des coefficients *non nuls* de chaque bloc serait proches les uns des autres alors la *différence* de leurs *indices* serait plus petite ce qui renforce l'idée *première idée*. Cela a été réalisé par *scan* (lecture) *adaptatif* de l'ordre de coefficients *non nuls*. Un scan parmi quatre scans (*Zigzag*, *Horizontal*, *Vertical* et *Hilbert*) qui donne la *différence minimale* des indices est choisi pour la lecture des indices de ces coefficients.

La contribution de ces deux idées a fait l'objet du *quatrième test* appliqué sur les différentes images. Le *scan adaptatif* a apporté une amélioration en moyenne de **6,80%**

pour le *CR* et de **6,67%** pour le *bpp* pour le même *PSNR* par rapport au scan *Zigzag* seul. Les résultats de ce test montrent, en plus de l'amélioration acquise par le scan *adaptatif*, l'importance et la présence de chaque scan dans le codeur. Ainsi, si on doit choisir un seul scan, certes le scan *Zigzag* est de loin le plus efficace et le plus présent justifiant ainsi sa sélection dans la norme JPEG.

La *troisième idée* est que *grouper* les données de même nature (dynamique) ensemble, et de coder ces *groupes séparément* donne un résultat meilleur que si les ces données sont *codées ensembles*. Cela est traduit par le codage *séparé* des vecteurs issues des étapes précédentes ; vecteur de *Headers*, vecteurs *NN* et vecteurs *dLUT*. Ce codage est réalisé par une compression *sans pertes* avec un codeur *arithmétique*. Une amélioration en moyenne de **33,08%** pour le *CR* et de **23,30%** concernant le *bpp* pour le même *PSNR* a été acquise. Les résultats du test dédié à cette idée justifient bien son importance.

Afin de mieux confronter nos résultats, nous avons comparé l'algorithme proposé avec d'autres méthodes [DHA07][DOU11][BOU12]. Pour différentes gammes du *PSNR*, notre méthode a montré sa dominance, son avantage et sa supériorité sur les algorithmes CDABS [DOU11] et JPEG [DHA07]. Pour un même *bpp*, notre méthode a permet de reconstruire les images originales avec une qualité meilleure que celles proposées par le standard JPEG [DHA07] et les méthodes CBTC-PF [DHA07] et CDABS [DOU11]. La combinaison de la DWT et un algorithme génétique de GA-DWT [BOU12] n'a pas donner des résultats meilleurs que notre méthode avec un déséquilibre fatal à notre profit en terme de *temps de compression*. Les tables obtenues de la dernière partie montrent que notre approche malgré qu'elle est simple, elle est très efficace en la comparant avec les méthodes testées.

Perspectives

Finalement, nous sommes loin d'avoir épuisé toutes les pistes d'améliorations. Le travail présenté dans cette thèse peut être étendu dans diverses directions. Nous présentons ici une liste non exhaustive des améliorations, des applications et d'adaptations possibles dont on peut citer :

1. Application de la méthode dans le domaine de l'imagerie médicale en tenant compte de leurs caractéristiques, et les contraintes sur la qualité de l'image reconstruite afin de ne pas nuire l'étape de diagnostic,

2. Vue sa simplicité de notre méthode, elle peut facilement adapter et appliquer sur les séquences vidéo,

3. Adaptation et application de la méthode proposée sur les images satellitaires,

4. Il sera intéressant de tester d'autres types de scan afin de trouver les meilleurs scans selon le domaine d'application,

5. Le codeur proposé peut être utilisé dans la compression des images *sans pertes* (*lossless* ou *near lossless*). Dans ce type de compression, les coefficients *nuls* dans le bloc à coder sont minoritaires. Il sera très utile de coder les indices des coefficients nuls à la place des coefficients *non nuls* ce qui permet de réduire efficacement la taille du vecteur dLUT,

6. La conversion colorimétrique utilisée RGB-YCbCr n'est pas forcement la meilleur, il fort attirant d'utiliser un espace de couleur approprié pour chaque application.

Annexe A

Code *Matlab* pour le tracé de la figure III.3
Propriété du compactage d'énergie de la transformation DCT.
Algorithme :

1. Lire l'image,
2. Calcul de la DCT de l'image,
3. Trier Descendant des coefficients DCT en valeur absolue,
4. Retenir les coefficients DCT triés avec le pourcentage voulu,
5. Reconstruire l'image à partir des coefficients DCT retenus.

```
% Auteur : Messaoudi Abdelhamid
% Le 10/01/2017

clc
close all
I = imread('lena256.tif');
J = dct2(I);
[n,m] =size(I);
L = n*m;

% Tri descendant des coefficients
[Js,idx] = sort(abs(J(:)),'descend');
p = 5; % Choix du pourcentage
indices = idx(1:L*p/100);
JJ = zeros(size(J));
JJ(indices) = J(indices);    % Retenir 5% de coefficients
Irec = uint8(idct2(JJ));     % Reconstruire l'image
J = uint8(J);                % convertion pour affichage

figure;imshow(I)   %Afficher image originale
figure;imshow(Irec)%Afficher image reconstruite (5%)
figure;imshow(J)   %Afficher les coefficients DCT en tant que image
```

Annexe B

Code *Matlab* pour le tracé de la figure III.4
Propriété de la décorrelation de la transformation DCT

```
% Auteur : Messaoudi Abdelhamid
% Le 10/01/2017

clc
close all
I = imread('lena256.tif');
J = dct2(I);
[n, m] =size(I);
PointsI =[ ];
PointsJ =[ ];

for i=1:n
 for j =1:m
  x = I(i,j);
  X = J(i,j);
  k = j+1;
  if k n
     k > j;   % derniere colonne (pas de voisin a droite)
  end
  y = I(i,k); % Voisin Droit
  Y = J(i,k); % Voisin Droit
  %Construction du vecteur pour l'affichage du voisinage I
  PointsI=[PointsI;[x y]];

  %Construction du vecteur pour l'affichage du voisinage J
  PointsJ =[PointsJ;[X Y]];
 end
end
%Correlation entre points de l'image Lena
figure, plot(PointsI(:,1),PointsI(:,2),'.');

%Correlation entre coefffcients DCT
figure, plot(PointsJ(:,1),PointsJ(:,2),'.');
```

BIBLIOGRAPHIE

[$AHM74$] N. Ahmed, T. Natarajan et R. K. Rao, *Discrete cosine transform*, IEEE Transactions on Computers, Vol 100, No 1, pp 90–93, 1974.

[$ALA17$] L. Alam, P.K Dhar, A. F. M. Mirza, H. Rashidul, M.G.S. Bhuyan et G.M. Daiyan, *An Improved JPEG Image Compression Algorithm by Modifying Luminance Quantization Table*, IJCSNS International Journal of Computer Science and Network Security, Vol 17, No 1, January 2017.

[$ANW01$] A Anwander, *Segmentation d'images couleur par un opérateur gradient vectoriel multiéchelle et contour actif : Application à la quantification des phases minéralogiques du clinker de ciment*, Thèse de Doctorat, Université de Lyon 2001.

[$ARO09$] M. el Aroussi, *Information Fusion towards a Robust Face Recognition System*, Thèse de Doctorat, Université Mohammed V – Agdal, 2009.

[$BAT12$] B. Battin, *Compression multi-vues des flux auto-stéréoscopiques*, Thèse de Doctorat, Université de Reims Champagne-Ardenne (URCA), 2010.

[$BEA97$] P. Beaurepaire, *Compression d'images appliquée aux angiographies cardiaques : aspects algorithmes, évaluation de la qualité diagnostique*, Thèse de Doctorat, Université de Technologie de Compiègne, 1997.

[$BEL12$] M. Beladgham, *Construction d'une technique d'aide au diagnostic en Imagerie médicale. Application à la Compression d'images*, Thèse de Doctorat, Université de Tlemcen, 2012.

[$BEN03$] R. Benzid, F. Marir, A. Boussaad, M. Benyoucef et D. Arar, *Fixed percentage of wavelet coefficients to be zeroed for ECG compression*, Electronics Letters, Vol 39, No 11, pp 830-831, 2003.

[$BEN05$] R. Benzid, *Ondelettes et statistiques d'ordre supérieur appliquées aux signaux uni et bidimensionnels*, Thèse de Doctorat, Université de Batna, 2005.

[$BEN07$] M. Benabdellah, *Outils de compression et de crypto compression : Applications aux images fixes et vidéo*, Thèse de Doctorat, Université de Mohammed Agdal, 2007.

[$BEN08$] R. Benzid, A. Messaoudi et A. Boussaad, *Constrained ECG compression algorithm using the block-based discrete cosine transform*, Digital Signal Processing, Vol 18, No 1, pp 56-64, 2008.

[*BER*16] S. Bertorello, *La Lumière*, http ://serge.bertorello.free.fr/optique/lumiere /lumiere.html, [10/10/2016].

[*BOU*10] A. Boucetta, *Etude de l'effet des transformées de décorrélation en compression des images couleurs RGB*, Thèse de Magister, Université de Batna, 2010.

[*BOU*12] A. Boucetta et K.E. Melkemi, *DWT based-approach for color image compression using genetic algorithm*, Image and Signal Processing Springer Berlin Heidelberg, pp 476–484, 2012.

[*BOU*13] N. Boukhenoufa, *Elaboration d'algorithmes pour la compression de données*, Thèse de Doctorat, Université de Batna, 2013.

[*BOU*14] I. Bouklihacene, *Compression d'images médicales par ondelettes de seconde génération*, Thèse de Doctorat, Université de Tlemcen, 2014.

[*BRA*04] J.J. Brault, D. Dougherty, *Les formats de compression d'image*, Rapport de projet, Institut Universitaire de Technologie de Tours, 2004.

[*BRA*11] N. Brahimi, *Développement et implémentation des algorithmes de compression d'images basés sur des transformées entières*, Thèse de Magister, Université de Sétif, 2011.

[*CAL*04] P. Le Callet et D. Barba, *Modèle de perception couleur : application à l'évaluation de la qualité*, Traitement du signal, Vol 21, No 5, pp 461-477, 2004.

[*CAL*06] P. Le Callet, C. Viard-Gaudin et D. Barba, *A convolutional neural network approach for objective video quality assessment*, IEEE Transactions on Neural Networks, Vol 17, No 5, pp 1316-1327, 2006.

[*CHE*05] H.C. Chen, *A memory-efficient realization of cyclic convolution and its Application to discrete cosine transform*, IEEE Transactions on circuits and systems for vedio Technology, Vol 15, No 3, pp 445-453, 2005.

[*CHE*77] W.H. Chen, H.C. Smith et S.C. Fralick, *A Fast Computational Algorithm for the Discrete Cosine Transform*, IEEE Transactions on Communications, Vol 25, No 9, pp 1004-1009, 1977.

[*CHO*05] S. Chouchane et W. Puech, *Intégration d'un nouveau marqueur dans le codeur d'images EZW basée sur les ondelettes*, CORESA : COmpression et REprésentation des Signaux Audiovisuels, Rennes, France. 2005.

[*CHR*06] E. Christophe, *Compression des images hyperspectrales et son impact sur la qualité des données*, Thèse de Doctorat, Ecole Nationale Supérieure de l'Aéronautique et de l'Espace, Paris, 2006.

[*CRO*76] R.E. Crochiere,S . A Webber, et J.L. Flanagan, *Digital coding of speech in subbands coding*, Journal Bell Syst. Tech, Vol 55, No 8, pp 1069-1085, Oct. 1976.

[*CZI*99] A. Czihó, *Quantification vectorielle et compression d'image. Application à l'imagerie médicale*, Thèse de Doctorat, Université de Rennes1, 1999.

[*DAU*92] I. Daubechies, *Ten lectures on wavelets*, Society for industrial and applied mathematics, 1992.

[*DHA*07] B.C Dhara, et B. Chanda, *Color image compression based on block truncation coding using pattern fitting principle*, Pattern Recognition, Vol 40, No 9, pp 2408-2417, 2007.

[*DOU*11] F. Douak,R. Benzid et N. Benoudjit, *Color image compression algorithm based on the DCT transform combined to an adaptive block scanning*, AEU-International Journal of Electronics and Communications, Vol 65, No 1, pp 16–26, 2011.

[*EMR*12] Z. Emre, *Compression multimodale du signal et de l'image en utilisant un seul codeur*, Thèse de Doctorat, Université de PARIS EST, 2012.

[*FAN*15] Y. Fanfan, C. Wang, et X. Zhou, *JPEG-Like Color Image Compression Based on All Phase Biorthogonal Transform and Improved Quantization Table*, Journal of Communications, Vol 10, No 11, November 2015.

[*GON*02] R.C. Gonzalez et R.E. Woods, *Digital image processing*, 2nd Ed, prentice Hall, New jersey, 2002.

[*GON*77] R.C. Gonzales et P. Wintz, *Digital Image Processing*, Addison Wessley, 1977.

[*GOU*02] A. Gouze, *Schéma lifting quinconce pour la compression d'images*, Thèse de Doctorat, Université de Nice 2002.

[*HEA*94] G. Healey et R. Kondepudy, *Radiometric CCD camera calibration and noise estimation Pattern Analysis and Machine Intelligence*, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol 16, No 3, pp 267–276, 1994.

[*HEU*09] R. Heus, *Approches virtuelles dédiées à la technologie des puces à tissus "Tissue MicroArrays TMA" : Application à l'étude de la transformation tumorale du tissue colorectal*, Thèse de Doctorat, Université de Joseph Fourier Grenoble I, 2009.

[*HOU*87] H.Hou, *A fast Recursive Algorithm for Computing the Discrete Cosine Transform*, IEEE Transaction on Speech and Signal Processing, Vol 35, No, 10, pp 1455-1461, oct 1987.

[*HUF*52] D. A. Huffman, *A method for the construction of minimum-redundancy codes*, In Proceedings of the Institute of Radio Engineers, Vol 40, No 9, pp 1098-1101, 1952.

[*ISA*02] A. Isar, A. Cubiţchi et M. Naforniţa, *Algorithmes et techniques de compression*, Editura Orizonturi Universitare, Timisoara, 2002

[*KAD*99] C. Kaddour et B. Aissa, *Compression des Images Fixes par Fractales basée sur la Triangulation de Delaunay et la Quantification Vectorielle*, Mémoire de fin d'études, Université de Houari Boumediene, 1999. (http ://web.archive.org/web/20030818214438/ http ://www.kaddour.com/index.htm).[10/10/2016]

[*KHA*03] S. A. Khayam, *The discrete Cosine Transform (DCT) : theorie and application*, Information theory and codage, ECE. pp 802-602 séminaire1-CEE, March 2003.

[*KUN*93] M. Kunt, G. Grunland et M. Kocher, *Traitement de l'information : traitement numérique des images*, Presses ploytechniques et Universitaires romandes, 1993.

[*LEE*84] B.G. Lee, *A new algorithm to compute the discrete cosine transform*, IEEE Transactions on Acoustics, Speech, and Signal Processing, Vol ASSP-32, No 6, pp 1243-1245, December é1984.

[*LEM*78], A. Lempel et J.Ziv, *Compression of individual sequences via variable-rate coding*, IEEE transactions on Information Theory, Vol 24, No 5, pp 530-536, 1978.

[*LEO*89] : C. Loeffler, A. Lieenberg et G.S. Moschytz, *Practical fast 1-d DCT algorithms with 11multiplications*, Acoustics, Speech, and Signal Processing, ICASSP-89., 1989 International Conference, IEEE, pp 988-991, 1989.

[*LIN*04] D. Lingrand, *Introduction au traitement d'images*, Edition Vuibert, Paris 2004.

[*LIN*80] Y. Linde, A. Buzo, et R. Gray, *An algorithm for vector quantizer design*, Communications, IEEE Transactions on, Vol 28, No 1, pp 84-95, January 1980.

[$LUK06$] R Lukac et N, P Kostantinos, *Color Image Processing : Methods and Applications*, CRC Press, 2006.

[$LUX16$] LUXORION, *photo-numerique*, http ://www.astrosurf.com/luxorion/photo-numerique.htm, [25/09/2016].

[$MA15$] J. Ma, T. Zhang, et M. Dong, *A Novel ECG Data Compression Method Using Adaptive Fourier Decomposition With Security Guarantee in e-Health Applications*, IEEE Journal of Biomedical and Health Informatics, Vol 19, No 3, May 2015.

[$MAL89$] S. Mallat, *A theory for multi-resolution signal decomposition : The wavelet representation*, IEEE Trans. On Pami, Vol 11, No 7, 1989.

[$MAN74$] J.L Mannos et D.J Sakrison, *The effect of visual fidelity criterion on the encoding of images*, IEEE ransactions on Information Theory, Vol 20, No 4, pp 525-536, 1974.

[$MAR03$] M. Mark, S. Grgic, et M. Grgic, *Picture quality measures in image compression systems*, EUROCON 2003, Computer as a Tool. The IEEE Region 8, Vol 1, pp 233-237, 2003.

[$MAR10$] A. Martin, *Représentations parcimonieuses adaptées à la compression d'images*, Thèse de Doctorat, Université de Rennes1, 2010.

[$MES16$] A. Messaoudi et K. Srairi, *Colour image compression algorithm based on the DCT transform using difference lookup table*, Electronics Letters, Vol 52, No 20, pp 1685-1686, 2016.

[$MIS04$] M. Misiti, Y. Misiti, G. Oppenheim et J.M. Poggi, *Wavelet Toolbox For Use with MATLAB User's Guide Version 3' Wavelet Toolbox User's Guide*, 2004.

[$MOF98$] A. Moffat, R. M. Neal et I. H. Witten, *Arithmetic coding revisited. ACM Transactions on Information Systems*, Vol 16, No 3, pp 256–294, July 1998.

[$MOI07$] M. Moinard, *Quantification et codage après transformée en ondelettes orientées Application à la compression d'images fixes*, Mémoire de Master de recherche, Institut de Recherche en Communications et en Cybernétique de Nantes, 2007.

[$MOR09$] N. Moreau, *Outils pour la compression : Application à la compression des signaux audio*, Thèse de Doctorat, Université de Paris Tech, 2009.

[$OUA09$] A. Ouafi, *Compression d'images avec pertes par codages imbriqués, Proposition d'une optimisation de l'algorithme EZW*, Thèse de Doctorat, Université de Biskra, 2009.

[$OUE12$] Y. Ouerhani, *Contribution à la définition, à l'optimisation et à l'implantation d'IP de traitement du signal et des données en temps réel sur des cibles programmables*, Thèse de Doctorat, Université de Bretagne occidentale – Brest, 2012.

[$PIL15a$] J.F. Pillou, *La lumière*, 2015, http ://CommentCaMarche.net. [15/10/2016].

[$PIL15b$] J.F. Pillou, *Le capteur*, 2015, http ://www.commentcamarche.net/ contents/2116-le-capteur [15/10/2016].

[$PIL15c$] J.F Pillou, *Le codage HSL-TSL*, 2015, http ://www.commentcamarche.net/ contents/le-codage HSL-TSL [15/10/2016].

[$RAB02$] M. Rabbani et R. Joshi, *An overview of the JPEG2000 still image compression standard*, Signal Processing : Image Communication, Vol 17, pp 3–48, 2002.

[$RAF94$] A. Rafaa, *Représentation multi-résolutions et compression d'images : Ondelettes et codage scalaire et vectoriel*, Thèse de doctorat, Université de Metz, 1994.

[*RIS*76] J.J. Rissanen , *Generalized Kraft Inequality and Arithmetic Coding*, IBM J. Res. Dev, Vol 20, pp 198-203, 1976.

[*SAI*96] A. Said et W.A. Pearlman, *A new fast and efficient image codec based on set partitioning in hierarchical trees*, IEEE Trans. Circuits and Systems for Video Technology, Vol 6, pp 243–250, June 1996.

[*SAL*04] D. Salomon, *Data Compression The Complete Reference*, Third Edition, 2004.

[*SAN*14] *Œil- comment-ça-marche 2014*, http ://www.sante365.org/2014/05/06/oeil-comment-ca-marche/, [01/12/2016].

[*SAY*06] K. Sayood, *Introduction to data compression*, Elsevier, San Francisco, USA, $4^{ème}$ édition, é2006.

[*SHA*48] C.E. Shannon, *A mathematical theory of communication, Part I, Part II*, Bell System Technical Journal, Vol 27, pp 623-656, 1948.

[*SHA*93] J. Shapiro, *Embedded Image Coding using Zerotree of Wavelet Coefficients*, IEEE trans. Signal processing, Vol 41, pp 3445-3463, December, 1993.

[*SIM*05] V. Simard, *Transformée en ondelettes pour un système d'acquisition de signaux corticaux implantable*, Thèse de Doctorat, Université de Montréal, 2005.

[*SUE*86] N. Suehiro et M. Hatori, *Fast algorithms for the DFT and other sinusoidal transforms*, IEEE Transactions on Acoustics, Speech, and Signal Processing, Vol ASSP-34, No 3, pp 642-644, June 1986.

[*TAQ*10] J. Taquet, C. Labit, *Une introduction à la compression d'images médicales Volumiques*, Rapport de recherche, INRIA, 2010.

[*TAQ*11] J. Taquet, *Techniques avancées pour la compression d'images médicales*, Thèse de Doctorat, Université de Renne 1, 2011.

[*TCH*07] D. Tchiotsop, *Modélisations polynomiales des signaux ECG. applications a la compression*, Thèse de Doctorat, Université de Nancy, 2007.

[*TOT*07] T. Totozafiny, *Compression d'images couleur pour application à la télésurveillance routière par transmission vidéo à très bas débit*, Thèse de Doctorat, Université de Pau et Des Pays de L'Adour, 2007.

[*TRA*99] Trac D. Tran. *A fast multiplierless block transform for image and video compression*, IEEE International Conference on Image Processing (ICIP'99), Kobe Japon, Vol 3, pp 822-826, October 1999.

[*USC*16] *USC Image Data Base*, http ://sipi.usc.edu/database/, [10/01/2017].

[*VAL*06] C. Valade, *Compression d'images complexes avec pertes : application à l'imagerie Radar*, Thèse de Doctorat, Université Paris Tech, 2006.

[*VER*15] J.K. Verma, A. Kumar et A.K Jaiswal, *Enhancement of ECG signal by DFT using Fast Fourier Transform (FFT) Algorithm*, International Journal of Current Engineering and Technology, 2015.

[*VET*84] M. Vetterli et H. J. Nussbaumer, *Simple FFT and DCT algorithms with reduced number of operations*, Signal Processing (North Holland), Vol 6, No 4, pp 267-278, August 1984.

[*WAK*93] J. Waku Kouomou, *Ondelettes et Applications en Imagerie et Calcul de Surface*, Thèse de Doctorat, Université Joseph Fourier, Grenoble I, 1993.

[$WAN84$] Z. Wang, *Fast algorithms for the discrete W transform and for the discrete fourier transform*, IEEE Transactions On Acoustics, Speech, and Signal Processing, Vol ASSP-32, No 4, pp 803-816, August 1984.

[$WEL84$] T. Welch, *A technique for high-performance data compression*, IEEE Computer, Vol 6, No 17, pp 8-19, 1984.

[$WIK16a$] Wikipedia, *Photographie argentique*, https ://fr.wikipedia.org/wiki/Photographie_argentique, [01/10/20016].

[$WIK16b$] Wikipedia, *Capteur photographique*, https ://fr.wikipedia.org/wiki/Capteur_photographique, [01/10/20016.]

[$WIK16c$] Wikipedia, *Caméra numérique*, https ://fr.wikipedia.org/wiki/Camera_numerique, [01/10/2016]

[$WIT87$] I.H. Witten, R.M. Neal et J.G. Cleary, *Arithmetic coding for data compression. Communications of the ACM*, Vol 30, No 6, pp 520–540, June 1987.

[$ZIT13$] C. Zitzmann, *Détection statistique d'information cachée dans des images naturelles*, Thèse de Doctorat, Université de technologie de Troyes, 2013.

[$ZIV77$] J. Ziv et A. Lempel, *A universal algorithm for sequential data compression*, IEEE Transactions on information theory, Vol 23, No 3, pp 337–343, May 1977.