MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE FERHAT ABBAS-SETIF
UFAS (ALGERIE)

MEMOIRE

Présenté à la Faculté de Technologie

Département d'Electronique

Pour l'Obtention du Diplôme de

MAGISTER

Option : Communication

Par

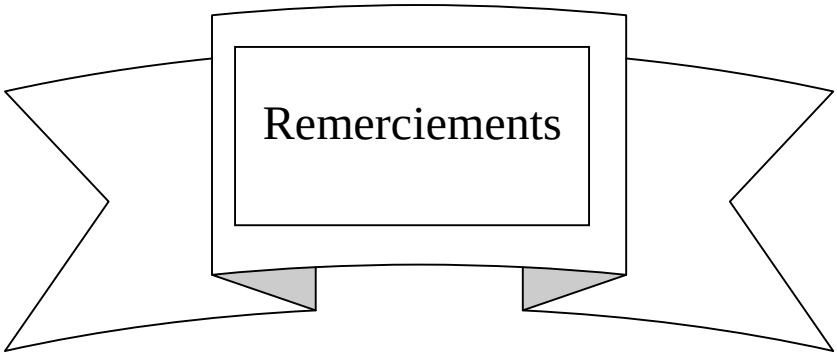
Mr : SAHIR MOURAD

Thème

Compression des images numériques par la technique des ondelettes

Soutenu le : 19 / 06 / 2011 devant la commission d'examen :

Mr : R. E. BEKKA	Prof à l'université de Sétif	Président
Mr : N. BOUZIT	Prof à l'université de Sétif	Examinateur
Mr : N. AMARDJIA	M.C à l'université de Sétif	Examinateur
Mr : A. FERHAT HAMIDA	M.C à l'université de Sétif	Examinateur
Mr : D. CHIKOUCHE	Prof à l'université de M'sila	Rapporteur



Je voudrais d'abord adresser mes vifs remerciements au Professeur Chikouche Djamel pour son encadrement, pour son aide et ses conseils très précieux tout au long de ce travail.
Je voudrais aussi exprimer mes remerciements aux membres du jury en acceptant de juger ce travail et d'y apporter leur caution :

- Professeur Rais El hadi Bekka, département d'électronique, Université de Sétif.
- Professeur Nacerdine Bouzit, département d'électronique, Université de Sétif.
- M.C. Nouredine Amardjia, département d'électronique, Université de Sétif.
- M.C. A / Elhak Ferhat Hamida, département d'électronique, Université de Sétif.

Que tous les enseignants qui ont contribué à ma formation trouvent en ce modeste travail, le témoignage de ma profonde gratitude.
Je tiens aussi à remercier tous qui de près ou de loin m'ont aidé et encouragé dans ce travail.

Résumé

L'objectif de ce travail est l'étude de la compression des images numériques fixes en niveaux de gris et couleur par la technique des ondelettes, plus précisément la norme de compression d'images JPEG 2000 (Joint Photographic Expert Group) basée sur les ondelettes. Cette nouvelle méthode, récemment proposée dans la littérature scientifique, a reçu une attention particulière d'un grand nombre de chercheurs ces dernières années. La norme de compression JPEG à base de la TCD (Transformée en Cosinus Discrète) a beaucoup de succès et reste encore une méthode de compression d'images très performante dans le domaine de l'Internet et des appareils photos numériques. Pour les faibles valeurs du taux de bits (quotient > 30), la qualité de l'image se dégrade rapidement. De plus, JPEG offre très peu de résistance aux erreurs à cause du codage à longueur variable de type Huffman. JPEG2000 avait comme objectifs principaux de répondre aux divers besoins, des nouvelles applications et d'apporter une nette amélioration par rapport à JPEG en termes de qualité et de résistance aux erreurs de transmission. Des tests comparatifs des performances de la méthode JPEG2000 en termes des taux de bits et qualité des images reconstruites avec la méthode de compression JPEG (Baseline) en mode avec perte ont été effectués dans ce travail sous Matlab7 pour différentes images de tests (images en niveaux de gris et couleur, image médicale). Les résultats obtenus montrent que la norme JPEG2000 est meilleure que la norme classique JPEG. D'autres tests ont été également réalisés sur les nouvelles fonctionnalités apportées par JPEG2000 telle que la Région d'intérêt(ROI), Codage de l'image avec ou sans pertes avec un seul algorithme, Décodage de l'image avec différents niveaux de résolution ou avec différents niveaux de qualité et aussi avec différents niveaux de résolution et qualité en même temps.

Mots-clés : Compression d'images fixes, JPEG2000, ondelettes, JPEG, DCT, qualité d'image, Région d'intérêt (ROI).

SOMMAIRE

Introduction générale ……... 1

Chapitre I : Généralités sur les images numériques

I-1. Introduction …….. 3
I-2. Définition de l'image…….
I-3. Image Numérique……..
I-4. Caractéristiques d'une image numérique ……. 4
I-4-1. Pixel …….....
I-4-2. Dimension……
I-4-3. Histogramme de l'image …….
I-4-4. Images aux Niveaux de Gris …… 5
I-4-5. Images en Couleurs……. 6
I-4-5-1. La représentation en couleurs réelles ……..
I-4-5-2. La représentation en couleurs indexées ……..
I-4-5-3. Autres modèles de représentation …….....
I-5. Images Bitmap et Images Vectorielles ……
I-6. Compression d'images ……..... 7
I-7. Principe général de la Compression des images…….. 8
I-8. Classification des méthodes de compression…… 9
I-8-1. Méthodes de compression sans perte d'informations…….. ….....
I-8-1-1. Méthode de codage RLE…….
I-8-1-2. Codage Huffman……..
I-8-1-2-1. Principe 10
I-8-1-2-2. Exemple de codage Huffman …….....
I-8-1-3. Codage arithmétique…… 11
I-8-1-3-1. Description de la méthode ……..
I-8-1-3-1-1. Le codage ……
I-8-1-3-1-2. Le décodage …… 12
I-8-1-3-1-3. Traitement d'un exemple……
I-8-2. Méthodes de compression avec perte d'informations …… 14
I-9. Evaluation de la compression …….. 15
I-10. Qualité de reconstruction des images ……. 15
I-11. Conclusion ……..... 17

Chapitre II : Etude du codeur d'images numériques JPEG

II-1. Introduction ……..... 18
II-2. Fonctionnement général du JPEG ……..
II-3. Transformation des couleurs …….. 19
II-4. Sous- échantillonnage de la chrominance (Cb et Cr) …… 20
II-5. Découpage en blocs …… 22
II-6. Transformation DCT……
II-6-1. Apport de la DCT dans la compression JPEG …… 23
II-7. La Quantification …… 24
II-7-1. Linéarisation des blocs …… 26

II-8. Le codage entropique …… 27
II-8-1. Codage intermédiaire ……
II-8-1-1. Codage intermédiaire des éléments DC ……
II-8-1-2. Codage intermédiaire des éléments AC …… 29
II-8-2. Codage HUFFMAN …….. 30
II -8-2-1. Codage d'HUFFMAN des coefficients Diff DC …… 30
II-8-2-2. Codage HUFFMAN des coefficients AC ……. 31
III-9. Phase de décompression …… 33
II-10. Conclusion …….. 34

Chapitre III : Etude du codeur d'images numériques JPEG2000

III-1. Introduction …… 35
III-2. Fonctionnement général du JPEG2000 …… 36
III-3. Prétraitement ou Traitements préliminaires …… 37
III-4. Transformée en couleur …… 38
III-5. Transformée en ondelettes …….. 39
III-5-1. Décomposition en deux Dimensions (2D) et niveaux de résolution …… 40
III-6. Quantification / Déquantification ……
III-6-1. Règle de Déquantification …… 43
III-7. Codage tier-1 …… 44
III-7-1. Codeur EBCOT ……
III-7-1-1. Codage Passe de signification …… 48
III-7-1-1-1. Codage et décodage du signe …… 49
III-7-1-2. Codage passe d'affinage …… 51
III-7-1-3. Codage passe de nettoyage ……
III-7-2. Le codeur arithmétique (MQ) …… 52
III-7-2-1. Principe de fonctionnement ……
III-7-2-1-1. Traitement d'échange conditionnel du sens de MPS …… 54
III-7-2-2. Principe de création et mise à jour des probabilités …….. 55
III-7-2-3. Description du codeur arithmétique JPEG2000…… 56
III-7-2-3-1. Les registres du codeur JPEG2000 …….. 57
III-7-2-3-2. Codage d'une décision (0 ou 1) ……
III-7-2-3-3. Codage de la décision comme un MPS ou LPS …… 58
III-7-2-3-4. Procédure de renormalisation des intervalles …… 59
III-7-2-4. Description du décodeur arithmétique JPEG2000 …… 60
III-8. Codage Tier 2 (Organisation du bitstream) ……..... 61
III-9. Régions d'intérêt …… 63
III-10. Contrôle du taux de compression…… 64
III-11. Résistance aux erreurs …… 65
IV-12. Conclusion …… 65

Chapitre IV : Résultats expérimentaux et interprétations

IV-1. Introduction …… 66
IV-2. Description des programmes et fonctions utilisés dans Matlab ……

IV-2-1. Bloc Acquisition des images …….. 67
IV-2-2. Bloc compression et décompression des images ……
IV-2-3. Bloc évaluation des résultats de tests et affichage des images……
IV-3. Tests d'applications …… 68
IV-3-1. Tests d'image au niveau de gris ……..
IV-3-2. Test avec différents niveaux de décomposition en ondelette et code-blocs…… 84
IV-3-3. Test sur une image médicale…… 85
IV-3-4. Test sur une image couleur…… 88
IV-3-5. Test Scalabilité en qualité sur une image au niveau de gris Lena…… 93
IV-3-6. Test Scalabilité en résolution sur une image au niveau de gris Lena…… 96
IV-3-7. Test Scalabilité en résolution et Qualité sur image au niveau de gris Lena….. . 98
IV-3-8. Test Scalabilité en qualité sur une image couleur poivron…… 100
IV-3-9. Test Scalabilité en résolution sur une image couleur poivron……..... 104
IV-3-10. Test Scalabilité en résolution et Qualité sur image couleur poivron……. 106
IV-3-11. Test Région d'intérêt (ROI) sur une image au niveau de gris Lena…… 107
IV-3-12. Test Région d'intérêt (ROI) sur une image couleur Lena……. 108
IV- 4. Conclusion…… 109
Conclusion générale …… 110
Annexes
Bibliographie

Introduction générale

L’image est un support d’information très performant, et comme on dit : une image vaut plus que mille mots.Vu l’importance de l’image, et la grande quantité d’information qu’elle peut contenir, le monde s’intéresse de plus en plus à l’image et tends vers l’universalisation de son utilisation. En effet, l’image a touché plusieurs domaines de notre vie : la médecine, la météo, la télécommunication, la cartographie, la géologie, etc.

Avec le développement de l’outil informatique, plusieurs techniques de traitement des images ont vu le jour [1]. Parmi les nombreux traitements qu’on peut effectuer sur les images, on trouve l’opération de compression des images. Cette opération devient plus que nécessaire vu le volume important d’information mis en œuvre lors de l’utilisation des images numériques. La contrepartie réside dans une quantité de données générées considérable qui peut rapidement saturer les systèmes conventionnels de transmission et de stockage [1]. A titre d’exemple, le CHU de Nancy-Brabois produit des examens de type Pet-Scan où les données sont parfois produites à partir d’une acquisition allant de la tête aux pieds du patient. Ainsi, un seul de ces examens peut à lui seul dépasser aisément le Giga octet. Un autre exemple est la taille d’examens plus classiques comme l’IRM ou le scanner qui varie actuellement entre plusieurs dizaines et centaines de Méga octets. Pour un PACS (Picture Archiving Communication System), d’un service classique de radiologie, cette masse de données se chiffre à plusieurs téra octets de données en une année. L’augmentation croissante et continue des capacités de stockage apporte une réponse partielle à ce problème mais demeure la plupart du temps insuffisante. La nécessité de compresser les images apparaît donc aujourd’hui incontournable. De plus, la compression présente un intérêt évident pour la transmission des images qui peut s’avérer délicate du fait des bandes passantes existantes limitées.

L’idée de base de la compression des images est de réduire le nombre moyen de bits par pixel nécessaire à sa représentation. Il est possible dans une certaine limite de réduire ce nombre sans perte d’information. Au-delà, il est nécessaire d’élaborer des algorithmes de compression irréversibles (avec pertes) induisant une distorsion pas ou peu visible dans les conditions normales d’observation des images .

L'objectif de ce travail de magister est l'étude de la compression des images numériques fixes au niveaux de gris et couleur par la technique des ondelettes , plus précisément la norme de compression d’images JPEG 2000 (Joint Photographic Expert Group) basée sur les ondelettes [2]. Cette nouvelle méthode, récemment proposée dans la littérature scientifique, a reçu une attention particulière d'un grand nombre de chercheurs ces dernières années. Nous

allons présenter une étude comparative des performances de cette méthode en termes de taux de bits et qualité de l'image reconstruite avec la méthode de compression conventionnelle JPEG 1992 à base de la TCD (Transformée en Cosinus Discrète) [3] qui a été adoptée comme norme internationale en 1992. Seul le profil de base de cette norme a connu un franc succès grâce en partie à l'arrivée de l'internet. C'est le mode de compression avec pertes basé sur la TCD. JPEG a beaucoup de succès et reste encore une méthode de compression d'images très performante dans le domaine de l'Internet et des appareils photos numériques. Pour les faibles valeurs du taux de bits (quotient > 30), la qualité de l'image se dégrade rapidement. Ceci est dû au fait que la TCD est appliquée sur des blocs 8x8. De plus, JPEG offre très peu de résistance aux erreurs à cause du codage à longueur variable de type Huffman [4], ce qui limite fortement le développement de JPEG dans les nouvelles applications. Suite à l'arrivée imminente de ces nouvelles applications, un groupe JPEG2000 a été formé en 1996 par, entre autres, les plus grands acteurs dans les domaines du mobile, de la photo et de l'Internet. JPEG2000 avait comme objectifs principaux de répondre aux divers besoins de ces nouvelles applications et d'apporter une nette amélioration par rapport à JPEG en termes de qualité et de résistance aux erreurs [5].

Afin de réaliser ces objectifs, le mémoire est structuré autour de quatre chapitres :

Le chapitre 1, est réservé à des généralités sur les différentes images et leurs définitions et les principales méthodes de codage entropique de données ainsi que la compression réversibles et irréversibles des images fixes et les principes critères d'évaluation de la compression et de la qualité des images reconstruites afin de se familiariser avec les notions et termes dans ce mémoire.

Le chapitre 2, traite en détails la chaîne de compression JPEG selon la norme, tout en décrivant les différentes étapes de compression et de la décompression dans le cas avec pertes.

Le chapitre 3, est consacré à l'étude de la norme de compression d'images JPEG2000 en détails en suivant la chaîne de compression et de la décompression de cette norme. Les fonctions de base décrites sont : la segmentation de l'image, la transformation de plans de couleurs, la transformée en coefficients d'ondelettes, la quantification, le codage des plans de bits avec le codeur arithmétique et l'arrangement des données dans le fichier.

Enfin, le dernier chapitre, est dédié à notre contribution. Des Programmes et fonctions sous Matlab 7 ont été développés afin de faciliter la compression et la décompression des images et l'évaluation des tests de qualité des images après décompression avec les deux codecs utilisés. Une discussion des perspectives de recherche sur la compression des images fixes par les ondelettes et la DCT clôture ce chapitre.

Chapitre I

Généralités sur les images numériques

I-1. Introduction

La compression des images est une nécessité plus vitale dans le domaine des multimédia, car les images numériques comme nous allons le découvrir, forment un énorme ensemble de données, même lorsqu'il s'agit d'une simple image fixe, d'où leur stockage et leur transmission qui deviennent très pénibles. Pour cela, nous observons la naissance de toute une branche de la science, celle qui traite les images fixe et mobile.

I-2. Définition de l'image

L'image est une représentation d'une personne ou d'un objet par la peinture, la sculpture, le dessin, la photographie, le film,…etc. C'est aussi un ensemble structuré d'informations qui, après affichage sur l'écran, ont une signification pour l'œil humain.

Elle peut être décrite sous la forme d'une fonction I(x,y) de brillance analogique continue, définie dans un domaine borné, tel que x et y sont les coordonnées spatiales d'un point de l'image et I est une fonction d'intensité lumineuse et de couleur. Sous cet aspect, l'image est inexploitable par la machine, ce qui nécessite sa numérisation.

I-3. Image Numérique

Contrairement aux images obtenues à l'aide d'un appareil photo (analogique), ou dessinées sur du papier, les images manipulées par un ordinateur sont numériques (représentées par une série de bits). L'image numérique est l'image dont la surface est divisée en éléments de tailles fixes appelés cellules ou pixels, ayant chacun comme caractéristique un niveau de gris ou de couleurs prélevé à l'emplacement correspondant dans l'image réelle, ou calculé à partir d'une description interne de la scène à représenter **.**

La numérisation d'une image est la conversion de celle-ci de son état analogique (distribution continue d'intensités lumineuses dans un plan xOy) en une image numérique représentée par une matrice bidimensionnelle de valeurs numériques X(n,m) où : n, m sont les coordonnées cartésiennes d'un point de l'image et X(n,m) le niveau de gris ou de couleur en ce point.

I-4. Caractéristiques d'une image numérique

L'image est un ensemble structuré d'informations caractérisé par les paramètres suivants:

I-4-1. Pixel

Le pixel est la contraction de l'expression anglaise " picture elements". Le pixel, étant le plus petit point de l'image, C'est une entité calculable qui peut recevoir une structure et une quantification.

Dans une image couleur (R.V.B.), un pixel peut être représenté sur trois octets : un octet pour chacune des couleurs : rouge (R), vert (V) et bleu (B).

I-4-2. Dimension

C'est la taille de l'image. Cette dernière se présente sous forme de matrice dont les éléments sont des valeurs numériques représentatives des intensités lumineuses (pixels). Le nombre de lignes de cette matrice multiplié par le nombre de colonnes nous donne le nombre total de pixels dans une image.

I-4-3. Histogramme de l'image

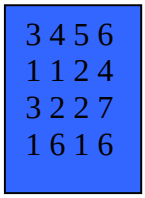
On veut souvent avoir une information sur la distribution des valeurs des pixels (niveaux) dans une image, pour cela on utilise souvent une table qui donne le nombre de pixels de chaque niveau dans l'image. Cette table, souvent représentée graphiquement, est appelée Histogramme de l'image, est notée h(v).

Exemple

Dans cet exemple, on veut représenter l'image de taille 4×4 pixels par son Histogramme de l'image et pour cela, on commence par calculer la Table Histogramme de l'image **(**voir tableau(I-1)) , ensuite on trace l'Histogramme de l'image comme montré dans la figure (I-1).

V : présente la valeur du pixel dans l'image.

h(v) : présente le nombre de fois répété de cette valeur du pixel dans l'image .



Image

v	1	2	3	4	5	6	7
h(v)	4	3	2	2	1	3	1

Tableau (I-1) : Table Histogramme.

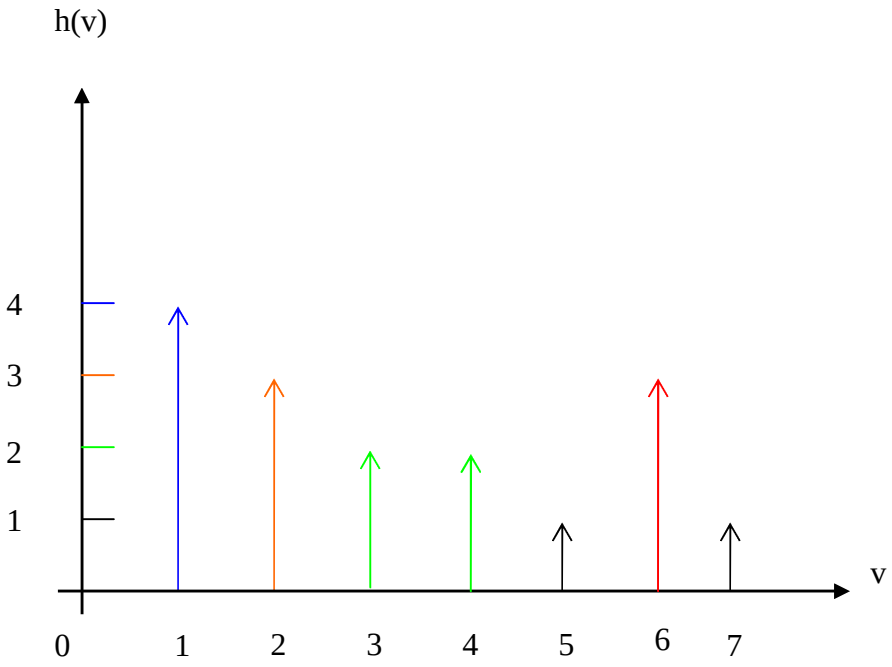


Figure (I-1) : Tracé de l'histogramme.

I-4-4. Images aux Niveaux de Gris

Le niveau de gris est la valeur de l'intensité lumineuse en un point. La couleur du pixel peut prendre des valeurs allant du noir au blanc en passant par un nombre fini de niveaux intermédiaires. Donc pour représenter les images à niveaux de gris, on peut attribuer à chaque pixel de l'image une valeur correspondant à la quantité de lumière renvoyée. Cette valeur peut être comprise entre 0 et 255. Par convention, la valeur zéro représente le noir (intensité lumineuse nulle) et la valeur 255 le blanc (intensité lumineuse maximale). Le nombre de niveaux de gris dépend du nombre de bits utilisés pour décrire la "couleur" de chaque pixel de l'image. Plus ce nombre est important, plus les niveaux possibles sont nombreux.

I-4-5. Images en Couleurs

Il existe plusieurs types de représentation des images couleurs, parmi celle-ci nous trouvons :

I-4-5-1. La représentation en couleurs réelles

Elle consiste à utiliser 24 bits pour chaque point de l'image. Huit bits sont employés pour décrire la composante rouge (R), huit bits pour la composante verte (V) et huit bits pour la composante bleu (B). Il est ainsi possible de représenter environ 16,7 millions de couleurs différentes simultanément.

I-4-5-2. La représentation en couleurs indexées

Pour réduire la place occupée par l'information de couleur, on utilise une palette de couleurs attachée à l'image. On parle alors de couleurs indexées : la valeur associée à un pixel ne véhicule plus la couleur effective du pixel, mais renvoie à l'entrée correspondant à cette valeur dans une table (ou palette) de couleurs appelée look-up table ou LUT en anglais, dans laquelle on dispose de la représentation complète de la couleur considérée.

I-4-5-3. Autres modèles de représentation

Le modèle RVB représentant toutes les couleurs par l'addition de trois composantes fondamentales, n'est pas le seul possible. Il en existe de nombreux autres. L'un d'eux est particulièrement important. Il consiste à séparer les informations de couleurs (chrominance) et les informations d'intensité lumineuse (luminance). Il s'agit du principe employé pour les enregistrements vidéo. La chrominance est représentée par deux valeurs (selon des modèles divers) et la luminance par une valeur.

I-5. Images Bitmap et Images Vectorielles

Les images appartiennent à deux grandes familles : bitmap (image-bit) et vectorielle. Alors qu'une image vectorielle est décrite à l'aide de courbes et d'équations mathématiques, une image bitmap est constituée de pixels et se réduit donc à une matrice de points. Si les images vectorielles peuvent être manipulées avec beaucoup de facilité, les modifications de taille, par exemple, apportées à une image bitmap ne sont pas sans incidence.

I-6. Compression d'images

Avec le développement de l'outil informatique, on effectue des échanges de volumes importants d'information. Or la gestion d'une telle masse pose des problèmes de stockage et de transfert. Pour cela, des études ont été menées afin de mettre en évidence des algorithmes de compression et de décompression de données. Leur but est de changer le format des informations de telle sorte qu'elles occupent moins de volume. Une fois compressées, les données ne sont plus accessibles en tant que données cohérentes; pour les récupérer, il suffit de les décompresser.

Vu les variétés des données (variétés des domaines d'utilisations des informations), il n'existe pas un seul compresseur de données, mais plusieurs types de compresseurs qui correspondent à autant d'algorithmes différents. L'efficacité de l'algorithme dépend du domaine d'application; par exemple, un algorithme qui donne de bons résultats pour la compression de fichiers textes, peut aboutir à de mauvais résultats pour la compression d'images.

Soit une image numérique X (N,N), donc constituée de N^2 éléments ; si on suppose que chaque échantillon est quantifié sur B bits, alors il nous faut B N^2 bits pour représenter cette image. Si on considère un cas très courant ou N=256, B=8, on doit utiliser 8×256^2 bits ou (65536 octets) pour représenter une petite image. Donc on remarque que le nombre de bits est beaucoup plus élevé (il va à des millions de bits) si on passe à une image de taille plus élevée.

Cette représentation de l'image, appelée la forme canonique d'une image ou image codée MIC (Modulation par Impulsion Codée) ou PCM (Pulse Coded Modulation), est très coûteuse, vu le très grand nombre de bits utilisés lors d'une transmission ou d'une mémorisation. L'idée donc est de réduire le nombre de bits nécessaires à la représentation d'une image en tirant parti de la dépendance des échantillons adjacents. Cette opération s'appelle "Réduction de redondance" ou "compression d'image". Parmi les principaux critères d'évaluation de toute méthode de compression en trouve :

1. La qualité de reconstitution de l'image.
2. Le taux de bits (Tb).
3. La rapidité du codeur et décodeur (codec).

I-7. Principe général de la Compression des images

La compression d'une image se fait en général selon le schéma synoptique suivant [6] :

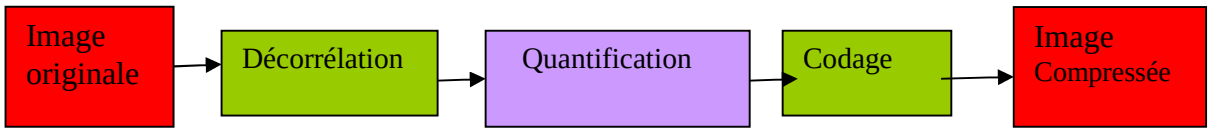


Figure (I-2) : Schéma synoptique de compression.

1) Décorrélation

La dépendance existante entre chacun des pixels et ses voisins (la luminosité varie très peu d'un pixel à un pixel voisin) traduit une corrélation très forte sur l'image. On essaie donc de tirer partie de cette corrélation, pour réduire le volume d'information en effectuant une opération de décorrélation des pixels. La décorrélation consiste à transformer les pixels initiaux en un ensemble de coefficients moins corrélés, c'est une opération réversible [6].

2) Quantification

La quantification des coefficients a pour but de réduire le nombre de bits nécessaires pour leurs représentations. Elle représente une étape clé de la compression. Elle approxime chaque valeur d'un signal par un multiple entier d'une quantité q, appelée quantum élémentaire ou pas de quantification. Elle peut être scalaire ou vectorielle.

3) Codage

Une fois les coefficients quantifiés, ils sont codés. Un codeur doit satisfaire à priori les conditions suivantes :

- Unicité : deux messages différents ne doivent pas être codés de la même façon.
- Déchiffrabilité : deux mots de codes successifs doivent être distingués sans ambiguïté.

Plusieurs types de codage seront détaillés ci-après.

I-8. Classification des méthodes de compression

Les différentes méthodes de compression d'images peuvent être classées en deux grandes classes d'après leurs propriétés [7]. Les méthodes de compression sans perte d'informations (réversibles) et les méthodes de compression avec perte d'informations (irréversibles).

I-8-1. Méthodes de compression sans perte d'informations

Ce sont des méthodes qui permettent de trouver exactement les échantillons de l'image originale. Parmi ces méthodes, on trouve : la méthode RLE, la méthode HUFFMAN...etc.

I-8-1-1. Méthode de codage RLE

La méthode de compression RLE (Run Length Encoding, parfois notée RLC pour Run Length Coding) est utilisée par de nombreux formats d'images. Elle est basée sur la répétition d'éléments consécutifs [8].

Le principe de base consiste à coder un premier élément donnant le nombre de répétitions d'une valeur puis le compléter par la valeur à répéter. Ainsi selon ce principe la chaîne "AAAAAHHHHHHHHHHHHHH" compressée donne "5A14H". Le gain de compression est ainsi de (19-5)/19 soit environ 73,7%. En contrepartie pour la chaîne "REELLEMENT", dans lequel la redondance des caractères est faible, le résultat de la compression donne "1R2E2L1E1M1E1N1T"; la compression s'avère ici très coûteuse, avec un gain négatif valant (10-16)/10 soit -60%!

En réalité la compression RLE est régie par des règles particulières permettant de compresser lorsque cela est nécessaire et de laisser la chaîne telle quelle lorsque la compression induit un gaspillage.

I-8-1-2. Codage Huffman

Le codage Huffman [9], crée des codes à longueurs variables sur un nombre entier de bits. L'algorithme considère chaque message à coder comme étant une feuille d'un arbre qui reste à construire. L'idée est d'attribuer aux messages de plus faibles probabilités, les mots codés les plus longs et les mots codés les plus courts pour les messages de fortes probabilités.

I-8-1-2-1. Le principe est le suivant

1. Calculer la probabilité associée à chaque symbole dans la chaîne à coder.
2. Classer les probabilités d'apparition des symboles par ordre de probabilités décroissantes.
3. Sommer les deux plus faibles probabilités correspondantes aux deux symboles de la séquence (la plus petite probabilité prenne un 0 et celle plus grande prenne un 1) ; ce qui nous donne une nouvelle probabilité.
4. Répéter les opérations 2 et 3, jusqu'à ce qu'il ne reste que deux probabilités (la racine).
5. Lire le code binaire de chaque symbole de haut vers le bas.

I-8-1-2-2. Exemple de codage Huffman

Soit une source composée d'un alphabet de 8 symboles avec les probabilités données sur le tableau (I-2).

Symbole	A	B	C	D	E	F	G	H
Probabilité	0.01	0.02	0.05	0.09	0.18	0.19	0.21	0.25

Tableau (I-2) : Symboles avec leurs probabilités.

L'arbre de Huffman est alors donné par la figure (I-3).

Le Codage des symboles est alors donné par le tableau (I-3).

Symboles	A	B	C	D	E	F	G	H
Code	110000	110001	11001	1101	111	00	01	10

Tableau (I-3) : Code des symboles.

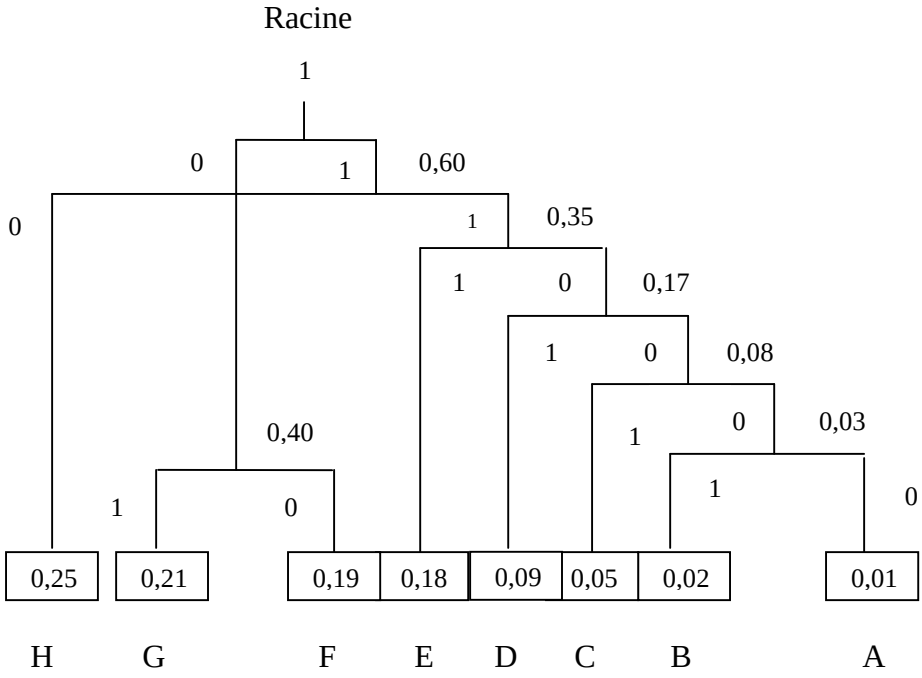


Figure (I-3) : Arbre de Huffman après construction.

I-8-1-3. Codage arithmétique

Le codeur arithmétique traite le fichier (messages) dans son ensemble [10], en lui associant un unique nombre décimal rationnel. Ce nombre est compris entre 0 et 1.

I-8-1-3-1. Description de la méthode

I-8-1-3-1-1. Le codage

Un message est représenté par un nombre réel compris entre 0 et 1. La génération de ce réel se base sur les probabilités d'apparition de chaque symbole. A chaque symbole, on lui affecte un intervalle de représentation, cette affectation n'a aucune influence sur la compression et la décompression. Le codage ce fait selon ces principes de base :

① Calculer la probabilité associée à chaque symbole dans la chaîne à coder.

② Associer à chaque symbole un sous intervalle proportionnel à sa probabilité, dans l'intervalle [0,1] (l'ordre de rangement des intervalles sera mémorisé car il est nécessaire au décodeur).

③ Initialiser la LimiteBasse de l'intervalle de travail à la valeur 0 et la LimiteHaute à la valeur 1.

④ Tant qu'il reste un symbole dans la chaîne à coder :

- Valeur =AncienneLimiteHaute-AncienneLimiteBasse.
- NouvelleLimiteHaute=AncienneLimiteBasse+Valeur*Valeur_Haute(c).
- NouvelleLimiteBasse=AncienneLimiteBasse+Valeur*Valeur_basse(c).

(c) : Indique l'intervalle courant du symbole en cours.

⑤ La LimiteBasse inférieure code la chaîne de manière unique.

I-8-1-3-1-2. Le décodage

Le décodage se fait par le processus inverse, ensuivant ces principes de base :

Valeur = AncienneLimiteHaute-AncienneLimiteBasse.

Nombre = (AncienNombre – AncienneLimiteBasse) / Valeur.

I-8-1-3-1-3. Traitement d'un exemple

Dans cet exemple nous allons compressé et décompressé un message dont le nom est : TECHNICIEN

La première étape consiste à calculer les probabilités d'apparition des différents caractères.

Pour ce cas on obtient le tableau (I-4) :

Caractère	**T**	**E**	**C**	**H**	**N**	**I**
Probabilité	1/10	2/10	2/10	1/10	2/10	2/10

Tableau (I-4) : Probabilités associée à chaque caractère à coder.

En deuxième étape, on affecte un intervalle à chaque caractère en se basant sur sa probabilité, on obtient ainsi les intervalles suivants tableau (I-5) :

Caractère	Probabilité	Intervalle
T	1/10	[0.0 - 0.1 [
E	2/10	[0.1 - 0.3[
C	2/10	[0.3 - 0.5 [
H	1/10	[0.5 - 0.6 [
N	2/10	[0.6- 0.8 [
I	2/10	[0.8 - 1.0 [

Tableau (I-5) : Association d'un intervalle à chaque caractère au fonction de sa probabilité.

Le premier caractère dans le message à compresser est le plus important, dans cet exemple c'est le caractère T, il doit être dans l'intervalle [0.0 - 0.1 [.le caractère suivant est E, il est dans l'intervalle [0.1 - 0.3 [,ainsi de suite jusqu'au dernier caractère N qui est dans l'intervalle [0.6 - 0.8 [.

Le calcul de mot de code des caractères est effectué suivant ces équations :

Valeur =AncienneLimiteHaute-AncienneLimiteBasse.

NouvelleLimiteHaute=AncienneLimiteBasse+Valeur*Valeur_Haute(c).

NouvelleLimiteBasse=AncienneLimiteBasse+Valeur*Valeur_basse(c).

Calculant le mot de code :

1) Initialisation

AncienneLimiteBasse = 0.

AncienneLimiteHaute = 1.

2) Déterminer les longueurs de sous intervalle liés au caractère T :

Valeur =AncienneLimiteHaute-AncienneLimiteBasse = 1-0 =1.

NouvelleLimiteHaute=AncienneLimiteBasse+Valeur*Valeur_Haute(c) = 0 + (1) (0.1) = 0.1.

NouvelleLimiteBasse=AncienneLimiteBasse+Valeur*Valeur_basse(c) = 0 + (1) (0) = 0.

3) Passer au codage sur le prochain caractère E :

Valeur =AncienneLimiteHaute-AncienneLimiteBasse = 0,1-0 = 0.1.

NouvelleLimiteHaute=AncienneLimiteBasse+Valeur*Valeur_Haute(c) = 0 + (0.1) (0.3) = 0.03.

NouvelleLimiteBasse=AncienneLimiteBasse+Valeur*Valeur_basse(c) = 0 + (0.1) (0.1) =0.01.

Ainsi de suite comme le résume le tableau (I-6) suivant :

Nouveau caractère	LimiteBasse	LimiteHaute
Aucun	0.0	1.0
T	0	0.1
E	0.01	0 .03
C	0.016	0.020
H	0.0180	0.0184
N	0.01824	0.01832
I	0.018304	0.018320
C	0.0183088	0.0183840
I	0.01836896	0.01838400
E	0.018370464	0.018373472
N	0.0183722688	0.0183728704

Tableau (I-6) : Tableau de codage du mot TECHNICIEN.

Le mot de code qui représente l'information compressée du message TECHNICIEN est La dernière valeur de la LimiteBasse de la dernière étape, dans ce cas c'est le 0.0183722688. Pour le décodage de l'information, il suffit de mettre en place le processus inverse.

I-8-2. Méthodes de compression avec perte d'informations

Ce sont des méthodes qui apportent une distorsion aux images reconstruites [11], [12]. Parmi ces méthodes, on trouve celles qui utilisent les transformées. Dans ces méthodes, l'image de dimension NxM est subdivisée en sous images ou blocs de taille réduite (la quantité de calcul demandée pour effectuer la transformation sur l'image entière est très élevée). Chaque bloc subit une transformation mathématique orthogonale inversible linéaire du domaine spatial vers le domaine fréquentiel, indépendamment des autres blocs (transformée en un ensemble de coefficients plus ou moins indépendants). Les coefficients obtenus sont alors quantifiés et codés en vue de leur transmission ou de leur stockage. Pour retrouver l'intensité des pixels initiaux, on applique sur ces coefficients la transformation inverse. Parmi les transformations linéaires existantes nous trouvons :

① Transformation de Karhunen-Loeve (TKL).

② Transformation de Fourrier discrète (TFD).

③ Transformation de Hadamard (TH).

④ Transformation en cosinus discrète (TCD).

⑤ Transformation en ondelettes (TO).

I-9. Evaluation de la compression

L'évaluation de la compression peut être effectuée par plusieurs paramètres [13], [14]. Parmi eux, on peut citer :

$$\textbf{1) Quotient} = \frac{\textbf{Taille des données originales}}{\textbf{Taille des données compressées}} \qquad \text{(I-1)}$$

Plus une compression sera forte, plus le quotient de compression sera lui aussi élevé.

2) Gain de compression = (1- 1/ Quotient) × 100 %, exprimé en pourcentage.

Plus le gain de compression est élevé, plus la taille du fichier compressé résultant est faible.

$$\textbf{3) Taux de bits (Tb) en bits/pixel (bpp)} = \frac{\textbf{Nombre original de bits/pixel}}{\textbf{Quotient}} \qquad \text{(I-2)}$$

Remarque :

Le taux de bits est aussi appelé débit de compression.

I-10. Qualité de reconstruction des images

Pour les méthodes de compression sans perte d'informations, les images reconstruites sont exactement les images originales. Le problème de la qualité se pose pour les méthodes de compression avec perte d'informations. En effet, l'image reconstruite n'est pas exactement l'image originale, ainsi il nous faut une mesure de la qualité de l'image reconstruite. Parmi ces fonctions, on trouve [15], [16]:

1) EQM (Erreur quadratique moyenne) : Elle permet de mesurer la dégradation entre une image $X_{origine}$ et l'image $X_{reconstruite}$ après compression de coordonnées (n,m) et de dimension (N,M).

$$EQM = \frac{1}{M \times N} \times \sum_{n=1}^{N} \sum_{m=1}^{M} \left(X_{origine}(n,m) - X_{reconstruite}(n,m) \right)^2 \qquad \text{(I-3)}$$

Avec :

$X_{origine}(n,m)$ et $X_{reconstruite}(n,m)$: représentent les valeurs des pixels d'image d'origine et reconstruite .

2) SNR (Signal to Noise Ratio) : Il permet de mesurer le signal original sur bruit d'une image.

$$SNR = 10 \times \log_{10} \left(\frac{\frac{1}{M \times N} \times \sum_{n=1}^{N} \sum_{m=1}^{M} \left(X_{origine}(n,m) \right)^2}{EQM} \right) dB \qquad \text{(I- 4)}$$

3) PSNR (Peak Signal to Noise Ratio) : C'est le rapport signal crête sur L'erreur d'une image.

$$PSNR = 10 \times \log_{10} \left(\frac{(X_{max})^2}{EQM} \right) dB \qquad \text{(I-5)}$$

Avec :

X_{max} : désigne la valeur du pixel maximale possible dans l'image qui prend une valeur 255 pour une image à niveau de gris codé sur 8 bits. Une valeur de PSNR infini correspond à une image non dégradée, et cette valeur décroît en fonction de la dégradation.

I-11. Conclusion

La représentation des images fixes est un des éléments essentiels des applications multimédias, comme la plupart des systèmes de communication. La manipulation des images pose cependant des problèmes beaucoup plus complexes que celle du texte. En effet, l'image est un objet à deux dimensions, censé représenter les détails de la scène photographie ou prise en vu, ce qui a deux conséquences majeurs :

- Le volume des données à traiter est beaucoup plus important.
- La structure de ces données est nettement plus complexe.

Il en résulte que la manipulation, le stockage et la représentation de ces données se heurtent à certaines limitations. Grâce au traitement d'images et plus précisément compression d'images, ces contraintes sont levées ou contournée.

Dans ce chapitre, nous avons présenté les notions de l'image numérique et ses différentes caractéristiques (pixel, dimension,…etc.) ainsi que les objectifs de la compression et les différentes solutions apportés, comme la compression sans perte d'informations (réversible) et avec perte d'informations (irréversible). Ensuite nous avons rappelé les critères d'évaluation de la compression et de la qualité des images reconstruites.

Chapitre II

Etude du codeur d’images JPEG

II-1. Introduction

Durant les dernières années précédentes, le groupe **joint photographic experts group**, plus connu sous l'acronyme **JPEG** [17], a fait de grands efforts dans le but d'établir un standard international pour la compression des images numériques. Le but de **JPEG** était de concevoir une norme internationale qui satisfait à tous les besoins des différentes applications (médical, militaire, astronomique, ou journalistique) de la compression des images fixes (à niveau de gris et couleur), entre autres le stockage des images ainsi que l'échange (transmission) entre divers organismes, qui doit être plus facile et plus économique. Parmi plusieurs propositions, le groupe a fait une rigoureuse sélection et c'était le choix de la DCT 8×8 qui a donné de bons résultats. Dans ce chapitre, nous décrirons l'algorithme de base de la norme JPEG (Baseline).

II-2. Fonctionnement général du JPEG

Le codage JPEG, s'effectue en plusieurs étapes comme le montre la figure (II-1) :

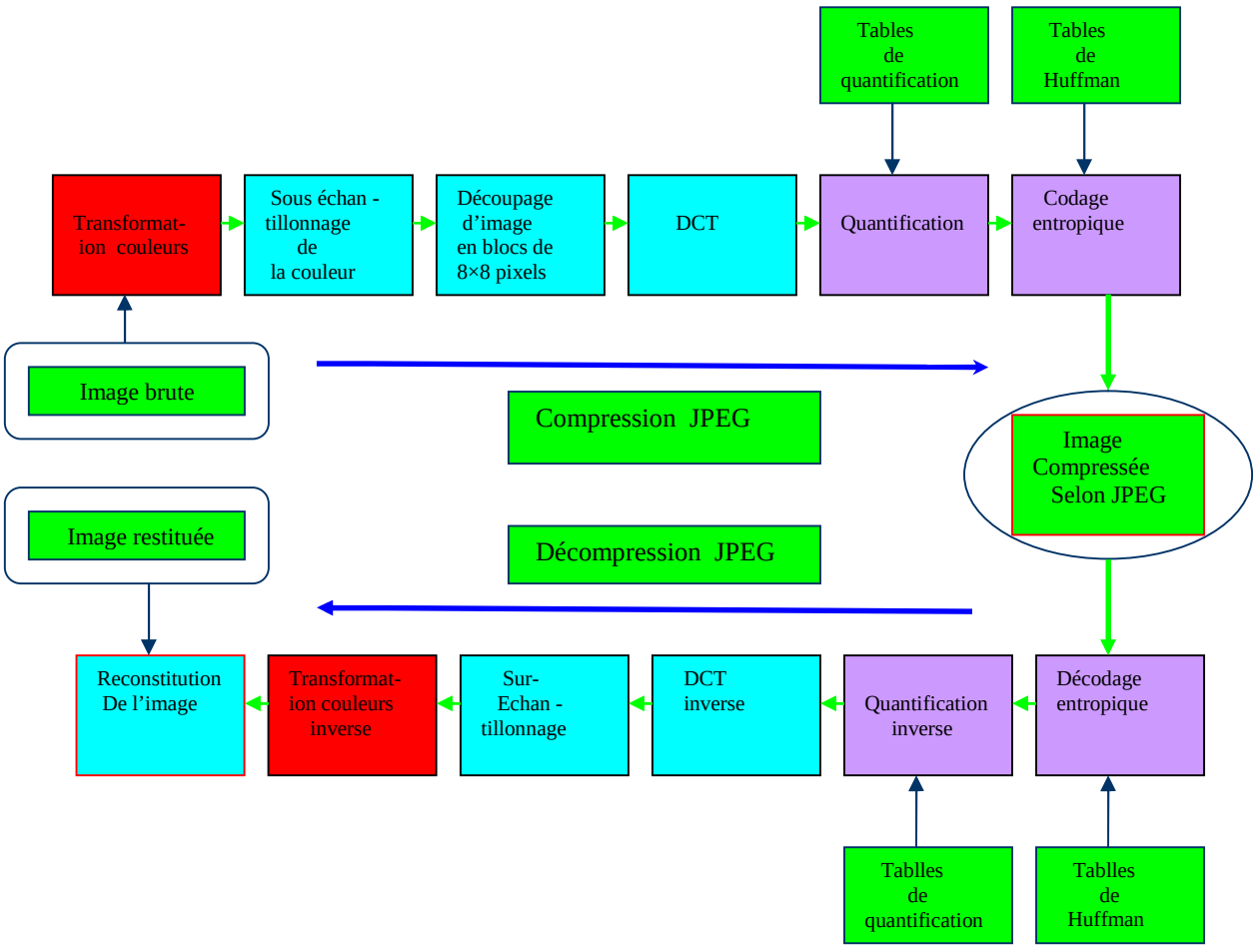


Figure (II-1): Principe général de la compression/décompression JPEG.

Le format JPEG, commence par traité séparément chaque composante de couleur. Il est possible d'utiliser directement le modèle RVB, mais JPEG est plus efficace sur le modèle YCbCr(après conversion éventuelle en luminance / chrominance). Comme l'œil humain est moins sensible aux changements de teinte (chrominance) que changements de luminosité (luminance). L'algorithme JPEG est basé sur cette différence de perception. D'ailleurs, il est même possible de sous – échantillonner ces composantes de chrominance , afin d'aboutir à un modèle prédéfini.Ensuite le format JPEG, commence par découper chaque composante YCbCr en blocs carrés de 64 (8 × 8) pixels(L'algorithme de compression est calculée pour chaque bloc séparé, ce qui explique pourquoi ces blocs ou groupes de blocs deviennent visibles lorsque la compression est trop appliquée) . L'étape suivante dans le processus de compression consiste à appliquer une transformée en cosinus discrète (DCT) à chaque bloc 8x8 pixels, Cette méthode permet de décrire chaque bloc en une carte de fréquences et en amplitudes plutôt qu'en pixels et couleurs, C'est l'étape qui coûte le plus de temps et de ressources dans la compression et la décompression , mais la plus importante car elle nous a permis de séparer les basses fréquences et les hautes fréquences présentes dans l'image. L'étape DCT lui-même est sans perte. La quantification est l'étape dans laquelle on perd réellement des informations (et donc de la qualité visuelle), mais c'est celle qui fait gagner beaucoup de place. Le but est ici est de garder les basses fréquences, c'est-à-dire celles auxquelles l'œil humain est très sensible et d'atténuer et de vouloir annuler les hautes fréquences, c'est à dire celles auxquelles l'œil humain est très peu sensible. JPEG utilise des tables de quantification, pour la luminance et pour la chrominance. Une fois que ces tables ont été construites, les constantes des tables sont utilisées pour quantifier les coefficients DCT. Chaque coefficient DCT est divisé par sa constante correspondante dans la table de quantification et arrondie au plus proche entier. Le résultat de la quantification des coefficients DCT sont parcouru en zig zag avant d'appliquer le codeur entropique qui est ici, Le codeur de Huffman qui compresse sans perte. Pour pouvoir reconstruire l'image originale à partir des données compressées, il faut appliquer l'étape de décompression donc de décodage, cette étape s'obtient en appliquant les procèdes inverses de l'étape du codage. Dans la suite nous allons voir les différentes étapes du codage et décodage JPEG en détails.

II-3.Transformation des couleurs

Une image couleur en mode RGB /RVB est constituée de trois plans (composantes) : rouge, vert et bleu. L'inconvénient de cette représentation réside dans le fait que chaque plan de couleur mélange à la fois la chrominance et la luminance. En effet, l'oeil humain est très sensible

à la luminance, soit à la variation de luminosité, et quasiment insensible à la chrominance, soit à la variation d'une même couleur.Il existe donc un premier moyen d'optimiser la compression de l'image. Transformer une image en mode RGB/RVB en une image en mode (YCbCr) ou (YUV) . Le mode YCbCr décompose une image en un plan définissant la luminance (Y), et deux plans définissant la chrominance notées (Cb et Cr).

Pour calculer les valeurs des composantes YCbCr d'une image à partir des composantes RGB/RVB , on utilise les formules suivantes [18]:

$$\begin{pmatrix} Y \\ C_b \\ C_r \end{pmatrix} = \begin{pmatrix} 0.2220 & 0.7067 & 0.0713 \\ -0.1195 & -0.3810 & 0.5 \\ 0.5 & -0.4542 & -0.0458 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix} \quad \text{(II-1)}$$

La conversion inverse se fait ainsi :

$$\begin{pmatrix} R \\ G \\ B \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1.5560 \\ 1 & -0.1866 & -0.48886 \\ 1 & 1.8580 & 0.0001 \end{pmatrix} \begin{pmatrix} Y \\ C_b \\ C_r \end{pmatrix} \quad \text{(II-2)}$$

Cette transformation des composantes RGB en composantes YCbCR est bijective. Ainsi il n'y a pas de pertes de données pendant cette étape.

II-4. Sous- échantillonnage de la chrominance (Cb et Cr)

La façon la plus simple d'exploiter la faible sensibilité de l'œil à la chrominance est simplement de sous-échantillonner les signaux de chrominance [8].

JPEG définit quatre (04) types de formats de sous – échantillonnage (voir figure (II-2)) :

1-Format 4 :4 :4 : sans sous-échantillonnage.

2-Format 4 :2 :2 : Enlever un échantillon sur deux en horizontal et ne rien faire en vertical.

3-Format 4 :2 :0 : Enlever un échantillon sur deux en horizontal et en vertical.

4-Format 4 :1 :1 : Enlever trois (03) échantillons sur quatre (04) en horizontal et ne rien faire en vertical. Ces sous-échantillonnages ne sont pas utilisés pour la luminance (Y).

La figure (II-3): montre un exemple de sous-échantillonnage de type 4 :2 :0

Lors de la décompression de l'image JPEG, il suffit de sur-échantillonner les plans Cb et Cr retrouvés après le décodage pour retrouver l'image d'origine.

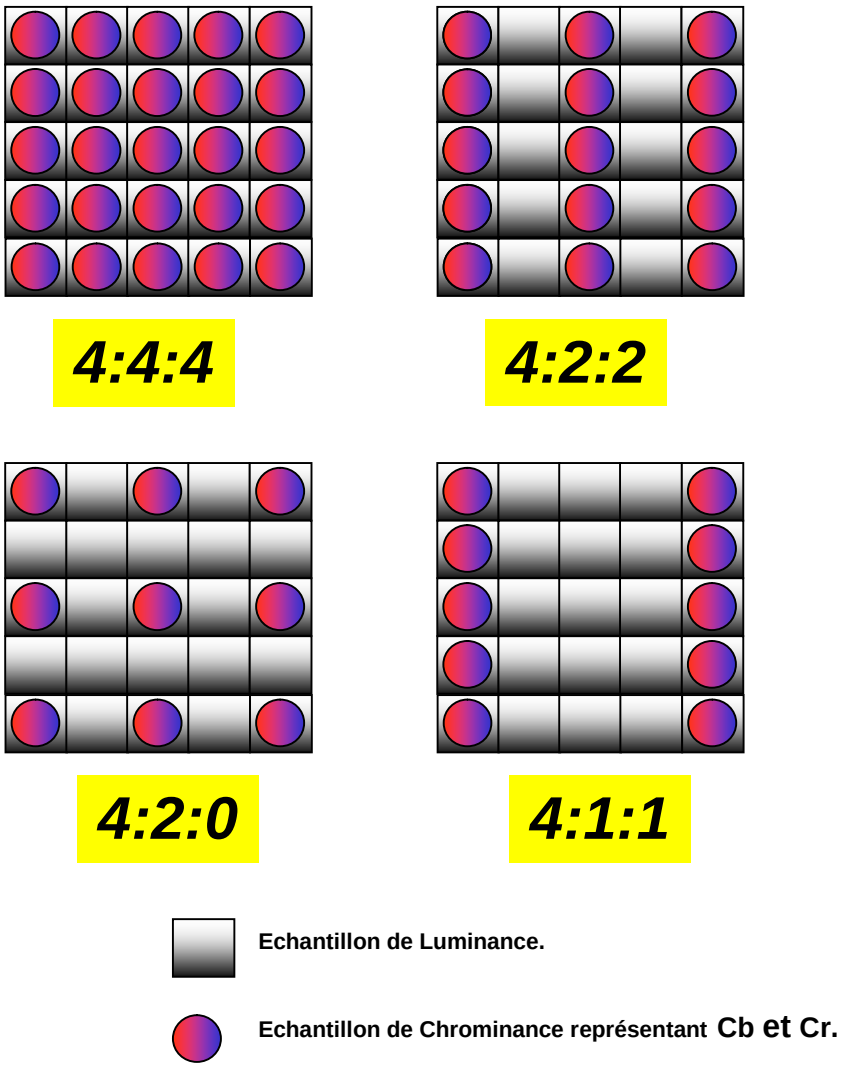


Figure (II-2): types de formats de sous – échantillonnage dans JPEG.

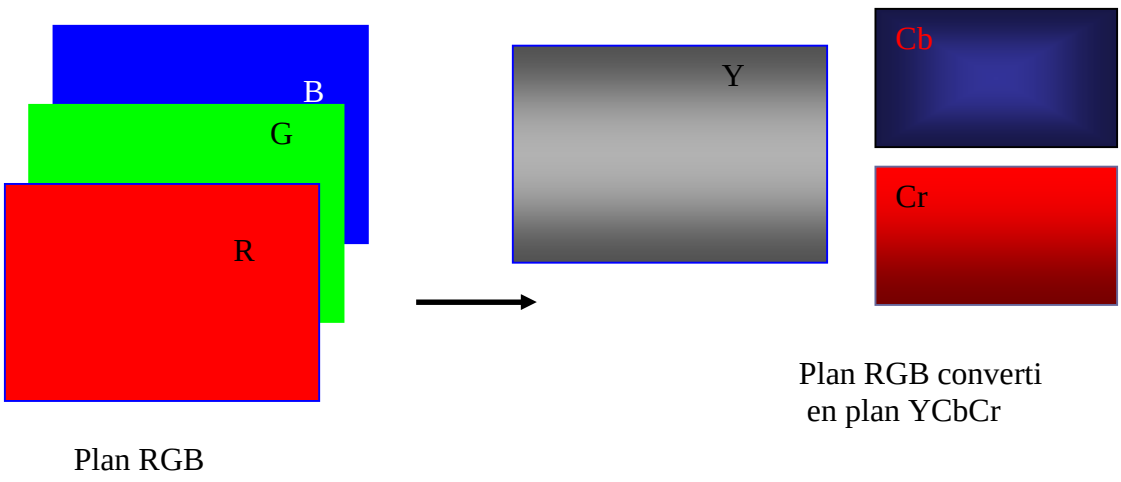


Figure (II-3): Exemple de sous-échantillonnage de type 4:2:0.

II-5. Découpage en blocs

L'image initiale est découpée en blocs de 8 × 8 pixels [19]. Ce découpage permet de réduire le temps de calcul et facilite les étapes de traitement qui suivent. La taille des blocs est toujours la même, telle que définie dans la norme.

II-6. Transformation DCT

La transformée DCT (Discrete Cosine Transform, en français transformée en cosinus discrète), est une transformation numérique qui est appliquée à chaque bloc de 8 × 8 pixels de chaque composante (YCbCr) ou (YUV) . La DCT converti des blocs d'échantillons en blocs de coefficients de fréquences, elle est de même nature que la transformée de Fourier.

La transformée DCT s'exprime mathématiquement par [20], [21] :

$$F(u,v) = \left(\frac{1}{4}\right).C(u).C(v)\sum_{i=0}^{7}\sum_{j=0}^{7} f(i,j).\cos\left((2.i+1).u.\frac{\pi}{16}\right).\cos\left((2.j+1).v.\frac{\pi}{16}\right) \quad \text{(II-3)}$$

Et la transformée DCT inverse s'exprime par :

$$f(i,j) = \left(\frac{1}{4}\right)\sum_{u=0}^{7}\sum_{v=0}^{7} C(u).C(v).F(u,v).\cos\left((2.i+1).u.\frac{\pi}{16}\right).\cos\left((2.j+1).v.\frac{\pi}{16}\right) \quad \text{(II-4)}$$

Avec :

$$\begin{cases} C(u) = C(v) = \frac{1}{\sqrt{2}} & \text{pour u = v = 0.} \\ C(u) = C(v) = 1 & \text{autres } u \neq 0 \text{ , } v \neq 0. \end{cases}$$

F(u,v) : représente la valeur de la DCT au point de coordonnées (u, v) dans le bloc résultat de 8 × 8 pixels.
f(i,j) : représente la valeur du pixel de coordonnées (i, j) dans le bloc de l'image originale de 8× 8 pixels.

Pour illustrer la transformée DCT nous avons pris un exemple d'un bloc de (8 × 8) pixels appelé S :

$$S = \begin{bmatrix} 139 & 144 & 149 & 153 & 155 & 155 & 155 & 155 \\ 144 & 151 & 153 & 156 & 159 & 156 & 156 & 156 \\ 150 & 155 & 160 & 163 & 158 & 156 & 156 & 156 \\ 159 & 161 & 162 & 160 & 160 & 159 & 159 & 159 \\ 159 & 160 & 161 & 162 & 162 & 155 & 155 & 155 \\ 161 & 161 & 161 & 161 & 160 & 157 & 157 & 157 \\ 162 & 162 & 161 & 163 & 162 & 157 & 157 & 157 \\ 162 & 162 & 161 & 161 & 163 & 158 & 158 & 158 \end{bmatrix}$$

En effectuant la transformée DCT à la matrice S on obtient la matrice des fréquences suivante :

$$F = \begin{bmatrix} 1260 & -1 & -12 & -5 & 2 & -2 & -3 & 1 \\ -23 & -17 & -6 & -3 & -3 & 0 & 0 & -1 \\ -11 & -9 & -2 & 2 & 0 & -1 & -1 & 0 \\ -7 & -2 & 0 & 1 & 1 & 0 & 0 & 0 \\ -1 & -1 & 1 & 2 & 0 & -1 & 1 & 1 \\ 2 & 0 & 2 & 0 & -1 & 1 & 1 & -1 \\ -1 & 0 & 0 & 1 & 0 & 2 & 1 & -1 \\ -3 & 2 & -4 & -2 & 2 & 1 & -1 & 0 \end{bmatrix}$$

II-6-1. Apport de la DCT dans la compression JPEG

Le coefficient de l'élément F(0,0), appelé coefficient continu(DC) (figure (II-4)), possède la valeur la plus élevée qui est proportionnelle à l'intensité moyenne du bloc de 64 pixels. Les autres coefficients, coefficients alternatifs (AC), correspondent aux variations d'un pixel à l'autres, ils représentent donc la contribution des différentes fréquences non nulles dans le spectre ainsi obtenu. Typiquement, les valeurs des pixels varient lentement de point au point adjacent à travers une image, en particulier pour des images qui ne présentent pas de discontinuités brutales. Par conséquent, les coefficients des basses fréquences, correspondant à des plages spatiales uniformes, ont des valeurs plus grandes que les coefficients des hautes fréquences. L'énergie du signal se trouve donc concentrée sur les fréquences spatiales inférieures, alors que la majorité des autres fréquences sont nulles ou presque.

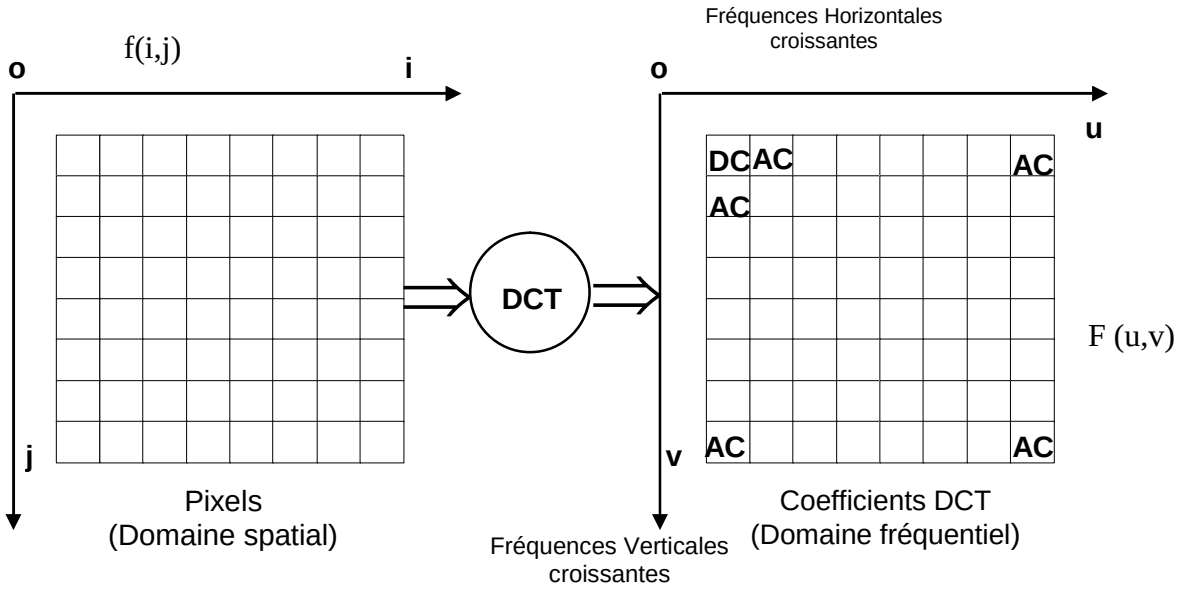


Figure (II-4) : Représentation de la DCT d'un bloc de 8 × 8 pixels.

II-7. La Quantification

Dans le bloc de 8×8 issu de la DCT, on a noté que la majorité de l'énergie du signal original est concentrée dans le coefficient DC (en haut à gauche du bloc de 8×8 coefficients) et dans quelques autres coefficients autour de celui-ci. En effet, les autres coefficients AC, en particulier ceux se situant en bas à gauche et à droite du bloc de 8×8, représentant les hautes fréquences et n'apportent donc que de la précision sur le signal original dans le bloc. Cette analyse permet de conclure à une quantification faible (petit pas de quantification) pour les coefficients se situant en haut à gauche du bloc et à une quantification de plus en plus forte (pas de quantification de plus en plus grand) à mesure que l'on se rapproche du coin inférieur droit et gauche du bloc. Cette quantification est donc réalisée par une matrice de quantification Q de la taille du bloc de 8×8 pixels. Elle s'applique sur un bloc de la DCT de telle sorte que [20], [22] :

$$F^*(u,v) = \text{entier le plus proche de} \frac{F(u,v)}{Q(u,v)} \qquad \text{(II-5)}$$

Avec :

F* (u,v) est la matrice quantifiée , F(u,v) est la matrice des coefficients à quantifier et Q(u,v) est la table de quantification.

Et pour la quantification inverse :

$$F^{\square}(u,v) = F^{*}(u,v) \times Q(u,v) \qquad \text{(II-6)}$$

Avec $F^{\square}(u,v)$ représente la matrice dequantifiée .

Dans les traitements d'images actuels, la table de quantification est définie par [22] :

$$Q(i, j) = 1 + (1 + i + j)\,Fq \qquad \text{(II-7)}$$

Où

Fq : Représente le facteur de qualité de la matrice de quantification. Plus Fq est grand, plus Q rendra les coefficients de la matrice à compresser se rapprochant du coin supérieur gauche à des valeurs nulles, et donc plus le signal bidimensionnel original sera dégradé. plus Fq est grand, plus la compression sera forte mais moins bonne sera la qualité de l'image reconstituée : Fq détermine le compromis à faire entre le gain en compression et la qualité de l'image reconstituée par rapport à l'image d'origine.

Exemple

Soit Fq = 5, alors la matrice de quantification Q sera :

$$\mathbf{Q} = \begin{bmatrix} 6 & 11 & 16 & 21 & 26 & 31 & 36 & 41 \\ 11 & 16 & 21 & 26 & 31 & 36 & 41 & 46 \\ 16 & 21 & 26 & 31 & 36 & 41 & 46 & 51 \\ 21 & 26 & 31 & 36 & 41 & 46 & 51 & 56 \\ 26 & 31 & 36 & 41 & 46 & 51 & 56 & 61 \\ 31 & 36 & 41 & 46 & 51 & 56 & 61 & 66 \\ 36 & 41 & 46 & 51 & 56 & 61 & 66 & 71 \\ 41 & 46 & 51 & 56 & 61 & 66 & 71 & 76 \end{bmatrix}$$

Lors de la décompression de l'image, il suffit de multiplier chaque bloc de 8×8 coefficients quantifiée par la matrice de quantification pour obtenir une estimation du bloc original. Revenant à notre exemple précédent. Pour quantifier notre matrice F résultante de la DCT nous avons pris la matrice de quantification suivante :

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

Ce qui donne comme matrice des fréquences quantifiées et arrondies en entier le plus proche :

$$F^* = \begin{bmatrix} 79 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ -2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

II-7-1. Linéarisation des blocs

Après la quantification et avant le codage entropique, la lecture des blocs ce fait d'une manière en Zig-Zag (balayage depuis les basses fréquences jusqu' au hautes fréquences) afin d'obtenir un maximum de zéros à la fin de la séquence (figure (II-5)) [20],[23].

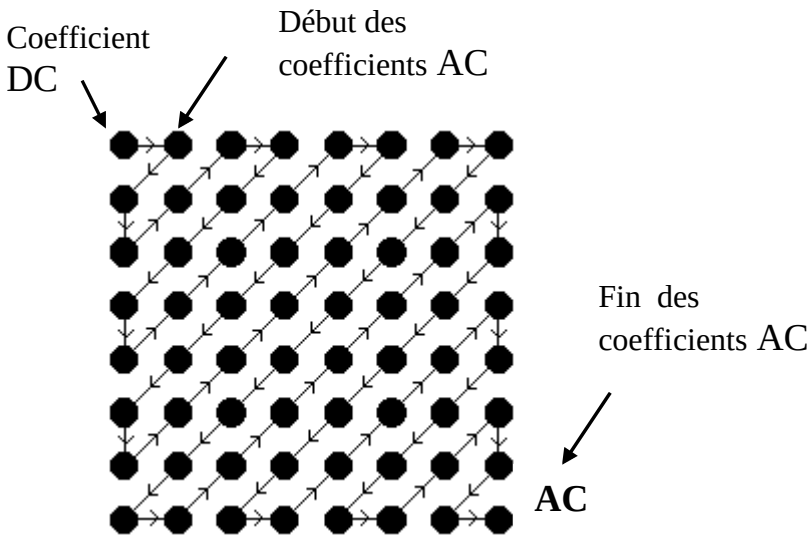


Figure (II-5) : Lecture en Séquence Zig-Zag.

Revenant toujours à notre exemple précédent. Le résultat de linéarisation de la matrice des fréquences quantifiées est :

79,0,-2,-1,-1,-1, 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, 0 ,0, 0 , 0 , 0, 0 , 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, EOB.

EOB : End of bloc (fin de bloc).

II-8. Le codage entropique [24] [25] [26]

Ce codage combine deux étapes. La première c'est le codage intermédiaire des coefficients DC et AC et la deuxième c'est le codage d'Huffman des coefficients différentiels Diff DC et des coefficients AC :

II-8-1. Codage intermédiaire

II-8-1-1. Codage intermédiaire des éléments DC [25]

Comme la variation des coefficients DC est très petite d'un bloc à un autre bloc adjacent, on codera non pas les coefficients DC mais la différence de deux coefficients entre deux blocs adjacents noté Diff DC (i) = DC (i) - DC (i – l). Les coefficients DC sont codés selon une prédiction unidimensionnelle entre le coefficient du bloc courant (i) et le coefficient du bloc précédent (i-l) (figure (II-6)), seul le coefficient du premier bloc d'image sera transmis sous sa forme absolue.

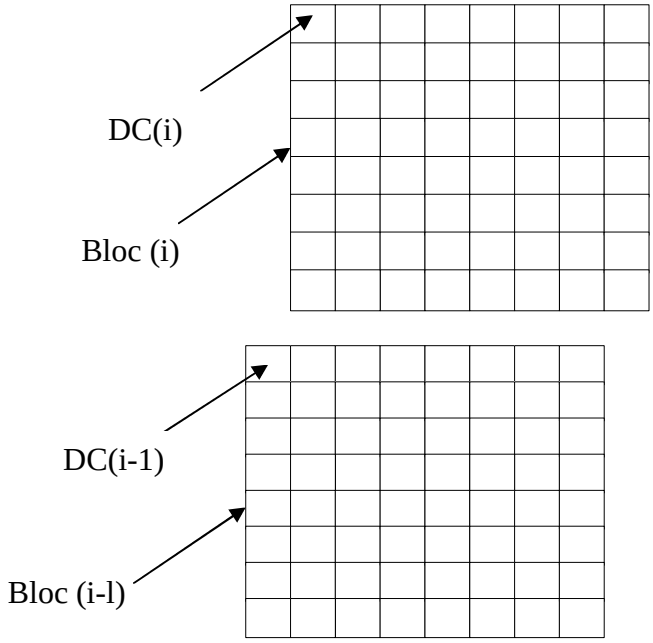


Figure (II-6) : Codage différentiel des coefficients DC.

Diff DC (i) = DC (i) - DC (i – l)

Cette différence est codée comme suivant :

On assigne au coefficient différentiel Diff DC (i) deux symboles de longueur variable.

SIZE / AMPLITUDE

SIZE : définit la catégorie d'amplitude ou appartiens le coefficient Diff DC (i) (voir tableau (II-1)) et qui est codée de 4 bits qu'on notera (SSSS) en binaire.

AMPLITUDE: représente la valeur de l'amplitude du coefficient différentiel Diff DC (i) en binaire.

Catégorie	SSSS	Amplitude
0	0000	0
1	0001	-1 à 0,0 à 1
2	0010	-3 à -2,2 à 3
3	0011	-7 à -4,4 à 7
4	0100	-15 à -8,8 à 15
5	0101	-31 à -16,16 à 31
6	0110	-63 à -32,32 à 63
7	0111	-127 à -64,64 à 127
8	1000	-255 à -128,128 à 255
9	1001	-511 à -256,256 à 1023
10	1010	-1023 à -512,512 à 1023
11	1011	-2047 à -1024,1024 à 2047

Tableau (II-1) : Catégories d'amplitudes des différences DC.

Remarque

Pour les nombres négatifs on va écrire le complément à 1 de la valeur absolue en la Diff DC (i) de l'amplitude.

Exemple 1

Pour deux blocs successifs on a :

D(i) = 500.

DC(i-l) = 355.

Diff DC(i) = 500 – 355 = 145.

D'après le tableau (II-1), 145 appartient à la catégorie 8 qui a une valeur en binaire 1000 (SSSS= 1000) et son codage complet sera 100010010001 :

Composé comme suit

SSSS	Amplitude
1000	10010001

Exemple 2

Pour deux autres blocs :

D(i) = 156

DC (i – l) = 1050

DIFF DC (i) = 156 – 1050 = - 894

D'après le tableau (II-1), - 894 appartient à la catégorie 10, son codage sera 10100010000001,

SSSS	Amplitude (en complément à 1)
1010	0010000001

II-8-1-2. Codage intermédiaire des éléments AC [26]

Pour les coefficients AC et comme la plus grande majorité d'entre eux sont tronqués à zéro on va utiliser un codage qui est basé sur la longueur de la séquence des zéro consécutifs le codage RLE (Run 1enght Encoding). Le codage de chaque coefficient AC, sera constitué de trois parties :

1) Longueur de la séquence des zéro (Run Length)

C'est le nombre de zéro consécutifs qui précède un coefficient AC non nul (Après un arrangement zigzag de ces coefficients). Ce nombre va être codé sur 4 bits (RRRR), il est compris entre 0 et 15.

2) size :

C'est la gamme dynamique de l'amplitude du coefficient AC non nul qui suit la séquence des zéro, définit la catégorie d'amplitude ou appartiens le coefficient AC (tableau (II-2)) et qui est codée en 4 bits qu'on notera (SSSS) en binaire.

Catégorie	SSSS	Amplitude
1	0001	-1 à 0,0 à 1
2	0010	-3 à -2,2 à 3
3	0011	-7 à -4,4 à 7
4	0100	-15 à -8,8 à 15
5	0101	-31 à -16,16 à 31
6	0110	-63 à -32,32 à 63
7	0111	-127 à -64,64 à 127
8	1000	-255 à -128,128 à 255
9	1001	-511 à -256,256 à 511
10	1010	-1023 à -512,512 à 1023

Tableau (II-2) : Catégories d'amplitudes des coefficients AC.

3) Amplitude:

Représente la valeur de l'amplitude du coefficient AC non nul en binaire.

Remarque 1

Lorsqu'une séquence de zéro est rencontrée à partir d'un point donné jusqu'à la fin du bloc, on écrira le code 00000000 (RRRR = 0000, SSSS = 0000).

Remarque 2

Pour les coefficients AC négatifs, on représentera leur amplitude par le complément à 1 de la valeur absolue du coefficient.

Exemple 1

Soit la séquence suivante issue d'un arrangement zigzag :

……..6,0,0,0,0,0,0,0,15,.

Son codage sera : 011101001111, composé comme suit :

RRRR	SSSS	Amplitude
0111	0100	1111
Nombre de zéros 7	appartiens à la Catégorie 4	15

Exemple 2

Soit la séquence suivante :

…….23,0,0,0,-19…….

Son codage sera : 0011010101100, composé comme suit :

RRRR	SSSS	Amplitude (en complètement à 1)
0011	0101	01100

II-8-2. Codage HUFFMAN

Le codage d'HUFFMAN nécessite plusieurs tables de codage. Celle-ci sont soit fournies au décodeur en les transmettant avec les données comprimées ou construites par lui-même.

II-8-2-1. Codage d'HUFFMAN des coefficients Diff DC [25] [26]

A chaque coefficient Diff DC, le codeur va lui attribuer un code de Huffman selon sa catégorie (indiquée par les 4 bits SSSS) dans la table de codage d' Huffman, la deuxième information (l'amplitude) n'est pas modifiée. Les coefficients Diff DC seront codés de la manière suivante :

Code HUFF (SSSS) + Amplitude

L'attribution du code HUFFMAN est donnée par le tableau (II-3).

Catégorie	SSSS	Code HUFFMAN	Longueur du code
0	0000	010	3
1	0001	011	3
2	0010	100	3
3	0011	00	2
4	0100	101	3
5	0101	1110	4
6	0110	11110	5
7	0111	111110	6
8	1000	1111110	7
9	1001	11111110	8
10	1010	111111110	9
11	1011	1111111110	10

Tableau (II-3) : Codes de HUFFMAN des coefficients Diff DC.

Exemple

Soit la séquence binaire suivante issue d'un codage intermédiaire d'un coefficient Diff DC : 100010010001, composé comme suit :

SSSS	Amplitude
1000	10010001

Son codage HUFFMAN d'après la table (II-3) pour la catégorie 8 (SSSS =1000) sera le suivant : 111111010010001, composé de la manière suivante :

Code HUFFMAN de SSSS	Amplitude
1111110	10010001

II-8-2-2. Codage HUFFMAN des coefficients AC

Pour les éléments AC du codage intermédiaire, seules les deux premières informations (codées sur 8 bits RRRRSSSS) vont être codées, la troisième information (amplitude du coefficient AC) ne sera pas modifiée. Les coefficients AC seront codés de la manière suivante :

Code HUFF (RRRRSSSS) + Amplitude.

Les codes de HUFFMAN des coefficients AC sont donnés dans le tableau (II-4) [26] :

Runlength /Catégorie(size)	Code de Huffman	(Longueur de code)
0 /0	1010 (=EOB)	4
0 /1	00	2
0 /2	01	2
0 /3	100	3
0 /4	1011	4
0 /5	11010	5
0 /6	111000	6
0 /7	1111000	7
0 /8	111110110	10
0 /9	1111111110000010	16
0 /A	1111111110000011	16
1/1	1100	4
1 /2	111001	6
1 /3	1111001	7
1 /4	111110110	9
1 /5	11111110110	11
1 /6	1111111110000100	16
1 /7	1111111110000101	16
1 /8	1111111110000110	16
1 /9	1111111110000111	16
1 /A	1111111110001000	16
2 /1	11011	5
2 /2	11111000	8
.	.	.
.	.	.
.	.	.
3 /5	1111111110010001	16
3 /6	1111111110010010	16
.	.	.
.	.	.
.	.	.
F /7	1111111111111011	16
F /8	1111111111111100	16
F /9	1111111111111101	16
F /A	1111111111111110	16

Tableau (II-4) : Codes de HUFFMAN des coefficients AC.

Exemple :

Soit la séquence binaire issue d'un codage intermédiaire d'un coefficient AC : 0011010101100, composé comme suit :

RRRR	SSSS	Amplitude
0011	0101	01100

Son codage HUFFMAN d'après le tableau (II-4) pour (Runlength /Catégorie (size)=3/5) sera : 111111111001000101100 , composé de la manière suivante :

Code HUFFMAN de (RRRRSSSS)	Amplitude
1111111110010001	01100

III-9. Phase de décompression

Pour pouvoir reconstruire l'image originale à partir des données compressées, il faut appliquer les étapes de la décompression donc de décodage qui s'effectuent dans l'ordre inverse de la compression suivant les méthodes définies précédemment.

Voici dans notre exemple le résultat de la matrice de décompression :

$$\hat{\mathbf{S}} = \begin{bmatrix} 144 & 146 & 149 & 152 & 154 & 156 & 156 & 156 \\ 148 & 150 & 152 & 154 & 156 & 156 & 156 & 156 \\ 155 & 156 & 157 & 158 & 158 & 157 & 156 & 155 \\ 160 & 161 & 161 & 162 & 161 & 159 & 157 & 155 \\ 163 & 163 & 164 & 164 & 162 & 160 & 158 & 156 \\ 163 & 163 & 164 & 164 & 162 & 160 & 158 & 157 \\ 160 & 161 & 162 & 162 & 162 & 161 & 159 & 158 \\ 158 & 159 & 161 & 161 & 162 & 161 & 159 & 158 \end{bmatrix}$$

Ainsi que la matrice d'erreur **E** :

$$\mathbf{E} = \mathbf{S} - \hat{\mathbf{S}} = \begin{bmatrix} -5 & -2 & 0 & 1 & 1 & -1 & -1 & -1 \\ -4 & 1 & 1 & 2 & 3 & 0 & 0 & 0 \\ -5 & -1 & 3 & 5 & 0 & -1 & 0 & 1 \\ -1 & 0 & 1 & -2 & -1 & 0 & 2 & 4 \\ -1 & 0 & 1 & -2 & -1 & 0 & 2 & 4 \\ -2 & -2 & -3 & -3 & -2 & -3 & -1 & 0 \\ 2 & 1 & -1 & 1 & 0 & -4 & -2 & -1 \\ 4 & 3 & 0 & 0 & 1 & -3 & -1 & 0 \end{bmatrix}$$

Remarque

Les valeurs sont, on peut le constater, très proches des valeurs originales : la différence maximale est de 5 sur quelques pixels, la moyenne des différences étant égale 1,7. On a donc finalement peu dégradée l'image originale alors que la quantification était importante.

II-10. Conclusion

Dans ce chapitre, nous avons fait une étude détaillée de la compression des images fixes selon la norme JPEG (Baseline), Qui s'effectue en plusieurs étapes qu'on peut résumer ici en deux étapes essentiels et cela après transformation couleur et sous-échantillonnage de l'image bien sur:

1) La première étape (DCT + Quantification) : C'est une compression qui va engendrer des pertes. En effectuant une transformation DCT sur une image, on va modifier sa représentation. C'est en tirant partie des propriétés de cette représentation (qui va concentrer l'information utile dans les basses fréquences) et en tolérant certaines erreurs dans nos calculs, que la première compression va se faire.

2) La deuxième étape Codage de Huffman : Qui s'effectue sans pertes. Un codage qui se base sur des caractéristiques statistiques; il s'agit du codage de Huffman. Dans cette partie, les informations codées seront reconstituées lors du décodage telles qu'elles sont, c'est-à-dire sans aucune perte.

Chapitre III

Etude du codeur d'images numériques JPEG 2000

III-1. Introduction

Le seul point commun que l'on pourrait trouver entre le standard de compression JPEG, et la compression JPEG2000 est le groupe de personnes à l'origine de ces algorithmes. Pour le reste, l'approche est radicalement différente. Le Joint Photographic Experts Group a voulu créer le successeur du standard JPEG, en prenant en compte ses limitations. Le besoin d'images compressées de bonne qualité est en constante augmentation et on a vu que la compression JPEG à son époque avait fourni des taux records qui ont permis par exemple le développement d'internet tel que nous le connaissons actuellement. Cependant, la méthode de la DCT par bloc est désormais un facteur limitant. A partir de la compression ``moyenne", on voit apparaître des artéfacts de compression, comme les blocs 8x8 pixels [27].

Ainsi le système JPEG2000 va reprendre certains éléments du JPEG comme le prétraitement de l'image en passant du RGB au YUV, mais ensuite, la compression va se faire sur l'image entière, et non plus sur des blocs réguliers. Cela permettra d'avoir une approche plus globale de la compression par l'usage d'ondelettes. On remplace ainsi la DCT (Discrete Cosinus Transform) par la DWT (Discrete Wavelet Transform). Cependant le principal intérêt du JPEG2000 réside dans sa grande flexibilité et ses très nombreuses fonctionnalités [28]. En effet la norme JPEG2000 ne définit que l'algorithme de décodage et le format des données compressées : cela pour permettre à n'importe quel système de lire et afficher une image compressée en JPEG2000.

Actuellement, JPEG2000 fournit une compression supérieure à JPEG pour les faibles valeurs des taux de bits, JPEG2000 est très largement meilleur. Il permet également une meilleure adaptation à des systèmes dont le débit d'information est faible : il peut envoyer en priorité certains bits correspondant à des régions importantes de l'image (ROI : Region Of Interest), ce qui fait que l'image sera intelligible plus rapidement. De plus, avec un seul standard de compression, on peut compresser avec ou sans perte. Cela n'était pas permis par JPEG. Nous vous présentons dans ce chapitre les différents blocs constituant le codeur JPEG 2000, les différentes subdivisions de l'image utilisées au cours du codage et les caractéristiques de ce nouveau système de compression d'images.

III-2. Fonctionnement général du JPEG2000

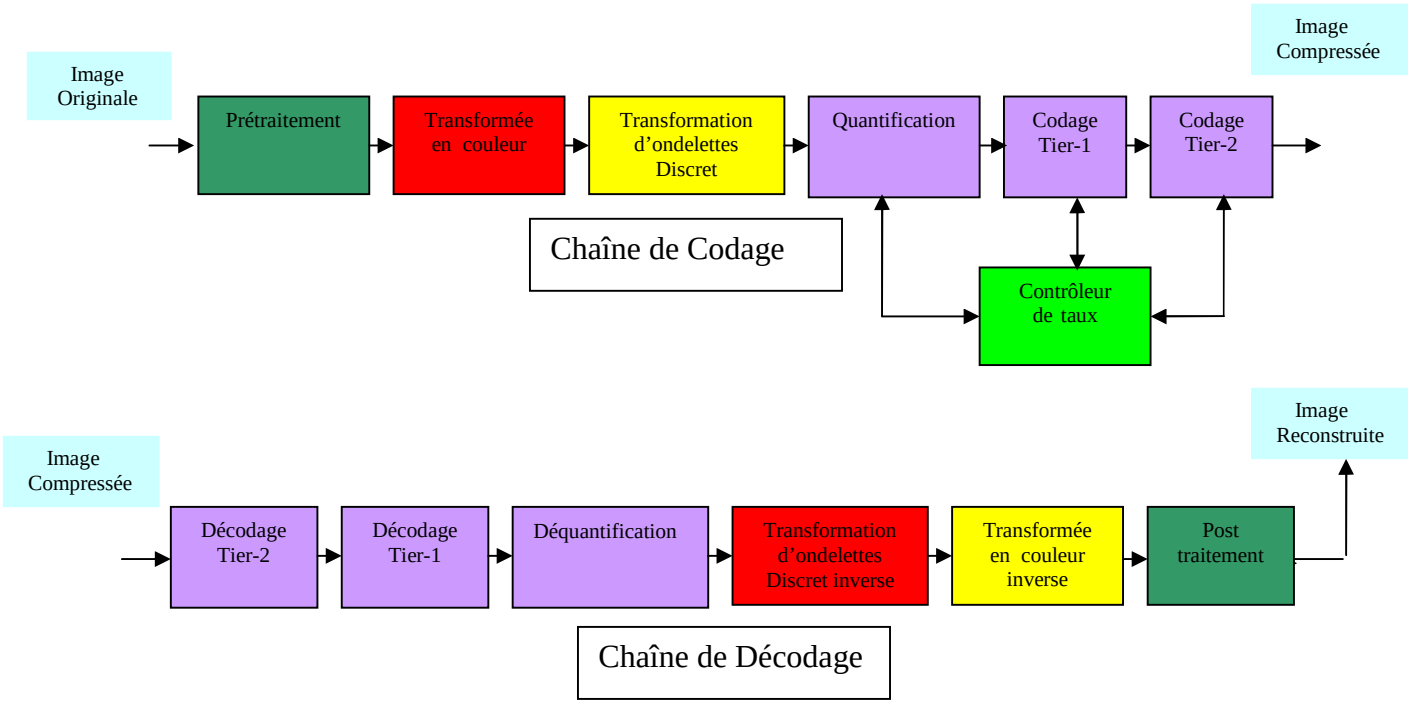


Figure (III-1) : Blocs constituants la chaîne de codage et décodage de JPEG2000 [15].

La chaîne de codage de JPEG-2000, commence par un prétraitement de l'image originale (figure (III-1)). L'image d'origine est segmentée en petites images appelées Tiles. Une Tile inclut tous les plans de couleur (RGB). Chaque plan d'une Tile est codé et décodé de façon indépendante. Cette méthode permet de faire du traitement parallèle matériel ou logiciel. Chaque plan de Tile est décorrélé par une transformée en ondelettes. Les coefficients d'ondelettes sont regroupés par niveau de décomposition. Chaque niveau de décomposition contient quatre sous-bandes. Cette représentation correspond au diagramme de Mallat [29]. Deux types de filtre sont proposés dans la première partie de la norme : les filtres (5,3) pour la transformation en ondelettes dite réversible et les filtres (9,7) pour la transformation dite irréversible qui fournit les meilleurs résultats en termes de décorrélation de l'image pour les faibles valeurs de taux de bits [30].

Une sous-bande regroupe les coefficients correspondant à une bande de fréquences spatiales horizontales et verticales d'un plan d'image. Pour l'instant, il n'y a pas encore de compression mais seulement une concentration de l'information sur un nombre réduit de coefficients.

Après quantification, l'information est encore plus concentrée sur un nombre réduit de coefficients. Les sous-bandes de « hautes fréquences » peuvent être quantifiées plus

fortement sans pertes visibles pour l'œil [31]. Chaque sous-bande d'un plan est divisée en blocs de coefficients. Ce sont des tableaux de coefficients rectangulaires qui pourront être extraits de façon indépendante afin de subir un codage entropique.

Le codage d'un bloc consiste à parcourir les coefficients plans de bits par plan de bits en allant du plan de bits de poids le plus élevé (MSB) vers le moins élevé (LSB). Le codage commence seulement lorsqu'un plan de bits devient signifiant. La position du plan de bits signifiant est alors stockée dans un en-tête du fichier et le codage des plans de bits commence. Chaque plan de bits est codé en trois passes successives excepté le premier plan signifiant. Chacune de ces trois passes récupère des informations contextuelles à propos des données du plan de bits, ces contextes qui serviront au codeur entropique de type arithmétique. Un codeur arithmétique utilise cette information contextuelle et son état interne pour coder les informations résultant du balayage de chaque plan de bits. Des paquets de données compressés sont générés. Un processus de terminaisons est mis en œuvre après chaque bloc de coefficients codés ou après chaque passe de plan de bits. Ce dernier mode de terminaison permettra de mieux résister aux erreurs de transmission. Dans la suite nous allons voir les différentes étapes du codage et décodage JPEG2000 en détails.

III-3. Prétraitement ou Traitements préliminaires

Deux opérations sont réalisées au cours de cette phase [32]**.** La première est le partitionnement de l'image en rectangles de même taille (sauf exception faite pour les zones situées aux extrémités de l'image) que l'on appelle Tiles (figure (III-2)) et qui ne se recouvrent pas. Les Tiles seront ensuite chacune codée séparément, avec leurs propres paramètres. Ce partitionnement est particulièrement utile dans les applications qui possèdent des ressources mémoire limitées.

La seconde opération : DC level shifting convertit les valeurs des composantes non signées de l'image en valeurs signées. Elle consiste à ramener ces coefficients en logique non signée de [0 255] en logique signée à [-128 127], c'est-à-dire centrés autour de zéro en soustrayant 128. Ce décalage (DC level shifting) est opéré afin de simplifier certains traitements ultérieurs.

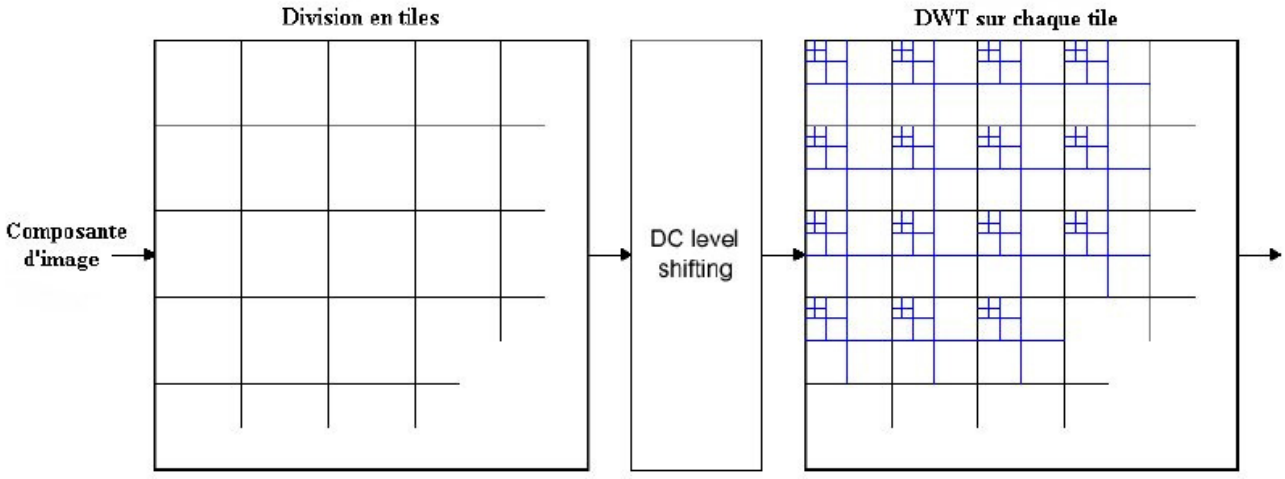


Figure (III- 2) : Division en tiles, décalage en DC et DWT sur chaque composante.

III-4. Transformée en couleur

Cette transformation est utilisée dans le cas des images appartenant à l'espace de couleur RVB, elle permet alors d'obtenir une représentation de l'image dans l'espace de couleur (YCrCb) ou (YUV) et un sous – échantillonnage de ces composantes de chrominance [32], [33]. Une telle transformée opère sur toutes les composantes et sert pour réduire la corrélation entre eux afin de pouvoir obtenir un codage plus performant.

Le standard supporte deux transformations possibles :

1) La première est une transformée irréversible, la ICT (Irreversible Color Transform) qui est utilisée avec une transformation en ondelettes irréversibles (filtres de Daubechies 9/7) dans le cas de la compression avec perte.

2) La deuxième est une transformée réversible, la RCT (Reversible color transform) qui est utilisée avec une transformation en ondelettes réversible (filtre de Gall 5/3) dans le cas de la compression sans perte.

Le tableau (III-1) donne un exemple des transformées directe et inverse de l'espace de couleur RVB dans l'espace de couleur (YUV) dans le cas de la transformée réversible et irréversible.

Mode	ICT	RCT
Codage	$\begin{pmatrix} Y \\ U \\ V \end{pmatrix} = \begin{pmatrix} 0.299 & 0.587 & 0.114 \\ -0.16875 & -0.33126 & 0.5 \\ 0.5 & -0.41869 & -0.08131 \end{pmatrix} \cdot \begin{pmatrix} R \\ G \\ B \end{pmatrix}$	$\begin{pmatrix} Y_r \\ U_r \\ V_r \end{pmatrix} = \begin{pmatrix} \left\lfloor \frac{R+2G+B}{4} \right\rfloor \\ R-G \\ B-G \end{pmatrix}$
Décodage	$\begin{pmatrix} R \\ G \\ B \end{pmatrix} = \begin{pmatrix} 1.0 & 0 & 1.402 \\ 1.0 & -0.34413 & -0.71414 \\ 1.0 & 1.772 & 0 \end{pmatrix} \cdot \begin{pmatrix} Y \\ U \\ V \end{pmatrix}$	$\begin{pmatrix} G \\ R \\ B \end{pmatrix} = \begin{pmatrix} Y_r - \left\lfloor \frac{U_r+V_r}{4} \right\rfloor \\ U_r+G \\ V_r+G \end{pmatrix}$

Tableau (III-1) : Transformation directe et inverse des plans de couleur pour le mode ICT (irréversible) et pour le mode RCT (réversible) en (YUV) .

III-5. Transformée en ondelettes

L'idée de base de la DWT (Discrete Wavelet Transform) est de séparer les basses et hautes fréquences d'une image [34]. Intuitivement, basses et hautes fréquences peuvent être comprises de la manière suivante. Les basses fréquences correspondent à une version grossière de l'image originale dans laquelle les valeurs des pixels ont été moyennées et ou aucune variation brusque n'est observée d'un pixel à l'autre. Les hautes fréquences quant à elles contiennent toute l'information sur les détails de l'image. On comprend aisément que plus d'information est contenue dans la version basses fréquences que dans celle ne fournissant que les détails de l'image. On dit que l'énergie de l'image est concentrée dans les basses fréquences.

L'objectif de la DWT est donc de concentrer l'information de l'image en une zone très localisée de manière à pouvoir ensuite compresser fortement les zones ne contenant que peu d'information.

En pratique, la transformée DWT consiste en l'application successive de paires de filtres passe-bas (H_0) et passe-haut (H_1), suivis d'un sous-échantillonnage de facteur deux (suppression d'un échantillon sur deux), comme l'illustre la figure (III-3). La transformée inverse consiste à appliquer une autre paire de filtres passe-bas (G_0) et passe- haut (G_1), précédés par un sur-échantillonnage de facteur 2 (insertion d'un zéro (0) entre chaque échantillons), ensuite une addition (figure (III-3).

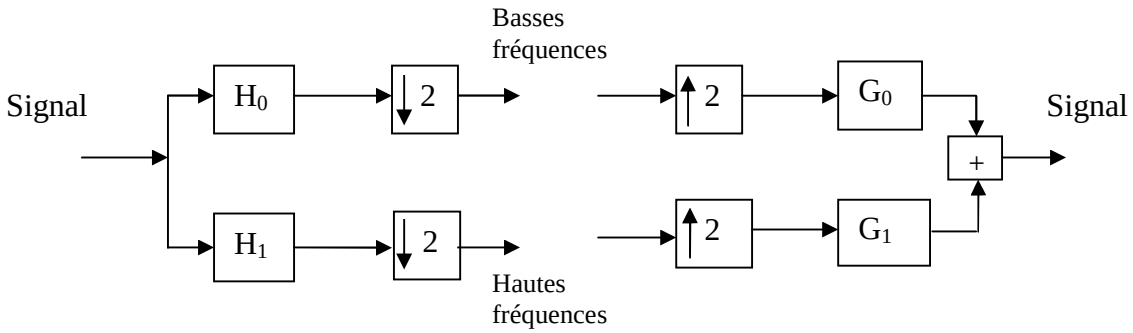


Figure (III-3) : Filtrage par des filtres passe-bas et passe-haut.

III-5-1. Décomposition en deux Dimensions (2D) et niveaux de résolution

Le principe de la décomposition 2D par filtrage passe-bas (H_0) et passe-haut (H_1) sur une image est représenté en figure (III-4). L'image est d'abord filtrée dans la direction horizontale (Ligne), et sous-échantillonnée avec un facteur de 2, ensuite dans le sens vertical (Colonne), et sous-échantillonnée avec un facteur de 2 aussi. Nous obtenons ainsi quatre sous-bandes (Sous-images) dans le domaine fréquentiel. Ces sous-bandes sont des matrices dont les dimensions sont réduites d'un facteur deux: LL (résulte du filtre passe-bas (H_0) dans les directions horizontale et verticale) correspond aux basses fréquences de l'image originale, ses coefficients sont donc les plus significatifs, LH (résulte du filtre passe - bas (H_0) horizontal et passe- haut (H_1) vertical) correspond aux détails verticaux de l'image originale, HL (résulte du filtre passe- haut (H_1) horizontal et passe- bas (H_0) vertical) correspond aux détails horizontaux de l'image originale et le HH (résulte du filtre passe-haut(H_1) horizontal et vertical) correspond aux détails diagonaux de l'image originale. Nous obtenons ainsi le premier niveau de résolution de la décomposition en ondelettes. Pour obtenir une décomposition à plusieurs niveaux de résolution en ondelettes, il suffit de décomposer de la même façon la sous-bande LL, nous obtenons alors un deuxième niveau de résolution de la décomposition, et ainsi de suite comme le montre le diagramme de Mallat en figure (III-5). La figure (III-6) donne un exemple visuel de l'image Barbara décomposée en deux niveaux de résolution de la décomposition en ondelettes.

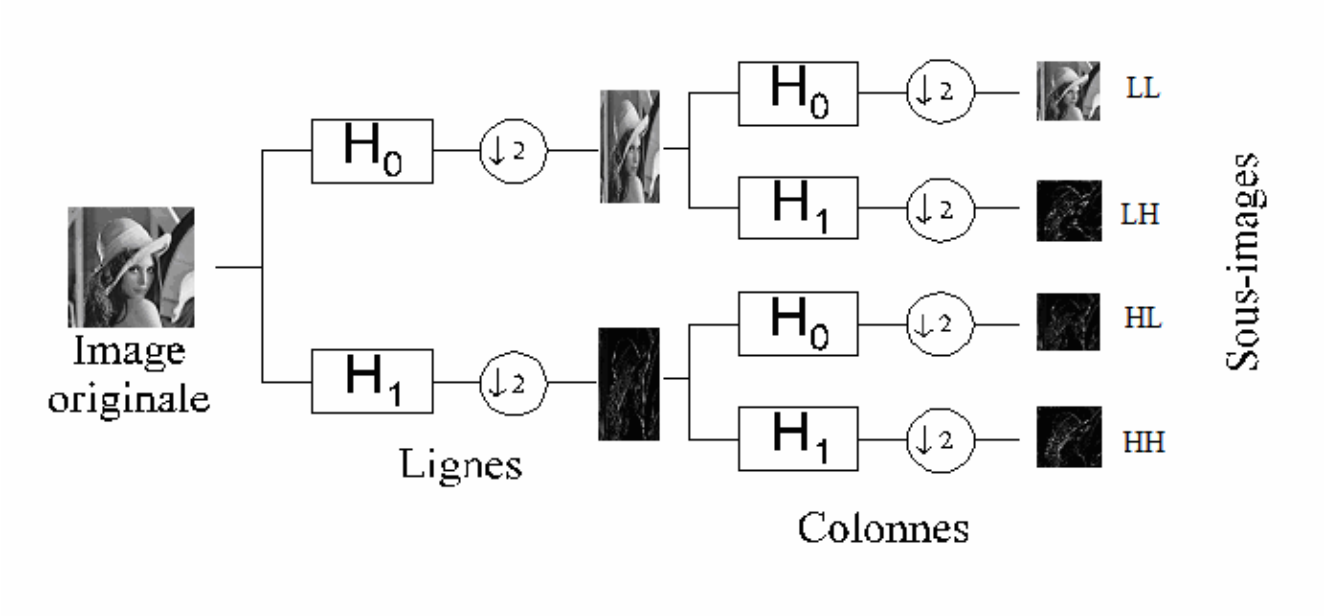


Figure (III-4) : Principe de décomposition 2D par filtrage passe - bas (H_0) et passe - haut (H_1) dans le sens horizontal (Ligne) et vertical (Colonne).

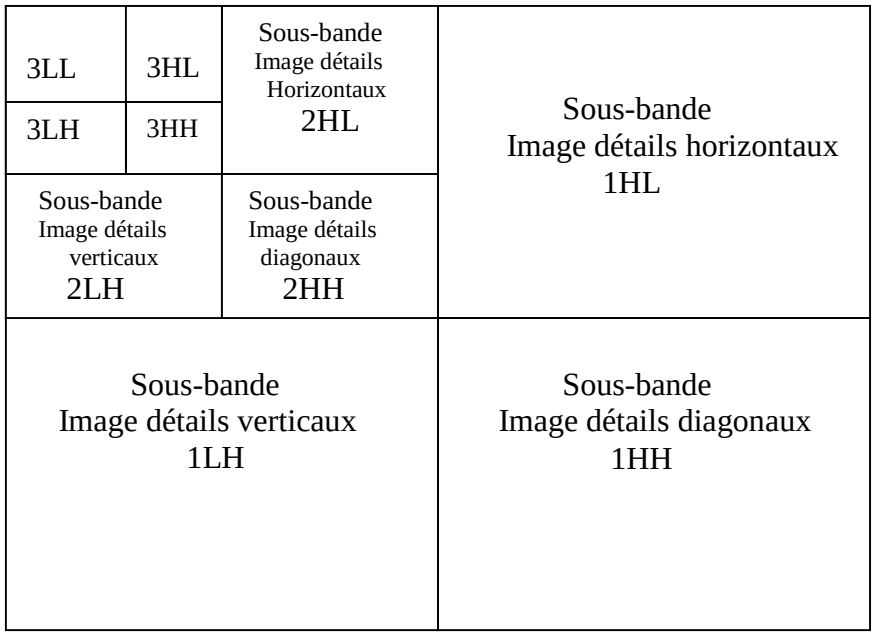


Figure (III-5) : Diagramme de Mallat représentant les coefficients d'ondelettes de la transformée, classés par sous-bandes de filtrage et niveau de décomposition.

Barbara image originale **Niveau 1** **Niveau 2**

Figure (III-6) : Exemple d'une décomposition en ondelettes à 2 niveaux de résolution.

Quatre sous-bandes d'un niveau de décomposition, par exemple 3LL, 3LH, 3HL et 3HH permettent de reconstruire une sous-bande de type LL du niveau de résolution R-1 soit 2LL (figure (III-5)). Ainsi, par recomposition successive, nous obtiendrons finalement l'image d'origine.

Le standard JPEG2000 utilise deux types de filtres d'ondelettes : la paire (9,7) de **Daubechies** (voir tableau (III-3)) utilisée pour la compression avec perte, et la paire (5,3) de **Le Gall** (voir tableau (III-2)) utilisée pour la compression sans perte. Le couple (9,7) veut dire qu'on a 9 coefficients du filtre passe-bas et 7 coefficients du filtre passe-haut dans le cas de la décomposition et le contraire dans le cas de la reconstruction, ainsi que Le couple (5,3) veut dire qu'on a 5 coefficients du filtre passe-bas et 3 coefficients du filtre passe-haut dans le cas de la décomposition et le contraire dans le cas de la reconstruction.

Coefficients du filtre de décomposition 5/3			**Coefficients du filtre de reconstruction 5/3**		
i	**Filtre passe-bas**	**Filtre passe-haut**	**i**	**Filtre passe-bas**	**Filtre passe-haut**
0	6/8	1	0	1	6/8
±1	2/8	-1/2	±1	1/2	-2/8
±2	-1/8		±2		-1/8

Tableau (III-2) : Coefficients des filtres correspondant à la transformation en ondelettes de type réversible de Le Galle (5,3) [35], [36].

Coefficients du filtre de décomposition 9/7			Coefficients du filtre de reconstruction 9/7		
i	**Filtre passe-bas**	**Filtre passe-haut**	**i**	**Filtre passe-bas**	**Filtre passe-haut**
0	0,6029490182364	1,1150870524570	0	1,1150870524570	0,6029490182364
±1	0,2668641184429	-0,5912717631142	±1	0,5912717631142	- 0,2668641184429
±2	- 0,0782232665290	-0,0575435262285	±2	-0,0575435262285	-0,0782232665290
±3	-0,0168641184287	0,0912717631142	±3	-0,0912717631142	0,0168641184287
±4	0,0267487574108		±4		0,0267487574108

Tableau (III-3) : Coefficients des filtres correspondant à la transformation en ondelettes de type irréversible (Daubechies 9/7) [37] .

III-6. Quantification / Déquantification

La transformation en ondelettes est suivie d'une quantification scalaire permettant de réduire la dynamique des coefficients , Cette opération est destructive, sauf si le pas dequantification est égal à 1 (Ce qui signifie qu'aucune quantification n'est appliquée). Mathématiquement, la fonction de quantification est donnée par l'équation suivante [32] :

$$q_b(u,v) = \text{signe}\left(a_b(u,v)\right) \times \frac{(|a_b(u,v)|)}{\Delta_b} \qquad \text{(III-1)}$$

Avec

q_b (u,v) : représentent les indices de quantification obtenus et a_b(u,v) sont les coefficients d'ondelettes à quantifier de la sous-bande b et Δ_b est le pas de quantification en fonction de la dynamique et du type de la sous-bande, du nombre de niveaux de décomposition .

III-6-1. Règle de Déquantification

La valeur de déquantification se calcule comme suit [32]:

$$\widetilde{q_b}(u,v) = \left[q_b(u,v) + r \times \text{signe}\left(q_b(u,v)\right)\right] \times \Delta_b \qquad \text{(III-2)}$$

Où

$\widetilde{q_b}(u,v)$: est la valeur reconstruite , $q_b(u,v)$ est l'index de quantification, Δ_b est le pas de quantification , le signe ($q_b(u,v)$) dénote le signe de $q_b(u,v)$, et r détermine le biais lié à la reconstruction et la valeur de r n'est pas normalisée par le standard JPEG2000.

- r = 0.5 résultat dans la reconstruction de midpoint.
  Une valeur populaire pour r est égal 0.375.

III-7. Codage tier-1

Cette partie est divisée en deux étapes de codage :

1) Codeur EBCOT : Il s'agit de l'entité qui parcourt les bits d'un **code-bloc** et qui envoie au codeur arithmétique binaire adaptatif MQ une succession de couples (bit à encoder (décision) et contexte du bit (CX) associé).

2) Codeur arithmétique binaire adaptatif MQ : Appeler aussi codeur MQ, il est chargé de coder ces couples (bit à encoder (décision) et contexte du bit (CX)).

III-7-1. Codeur EBCOT

Après la quantification scalaire, les coefficients des différentes sous-bandes sont codés avec un algorithme **EBCOT(Embedded Block Coding with Optimized Truncation)** [38], [39]. Cet algorithme a été créé en majorité par David Taubman, le principe de base de l'EBCOT est le suivant. Après la quantification scalaire, les coefficients issus des différentes sous-bandes de la transformée en ondelettes sont rangés en blocs , appelés code-blocs (figure (III-7)), de forme rectangulaire, de taille paramétrable, leurs hauteurs et largeurs correspondant à une puissance de deux, et le produit largeur hauteur ne devant pas dépasser 4096 (64 x 64) avec une hauteur et largeur minimale de 4. Chaque code-bloc est codé indépendamment, sans aucune référence aux autres code-blocs de la même sous-bande ou d'une autre. Ce codage indépendant offre des avantages importants, comme un accès spatial aléatoire au contenu de l'image, calcul parallèle durant le codage ou décodage [40].

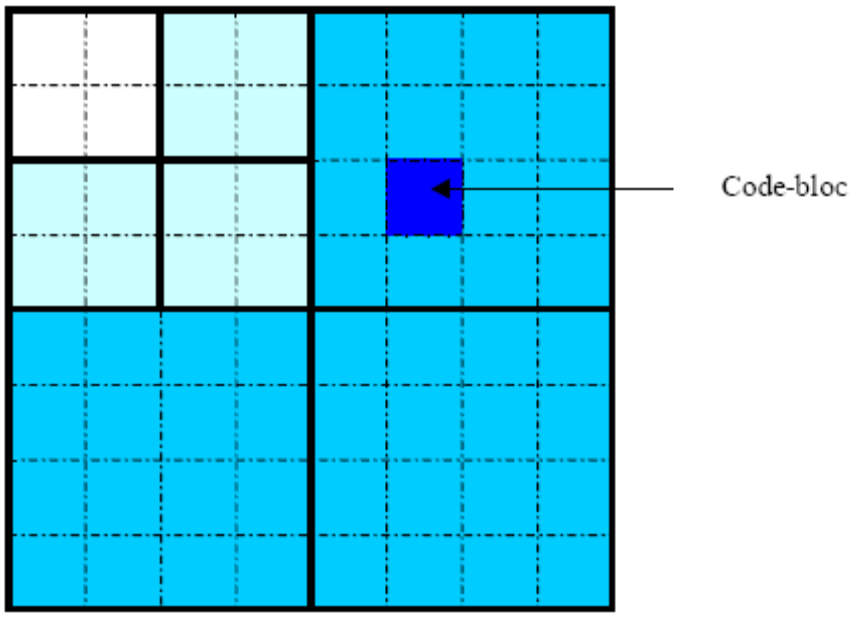


Figure (III-7) : Partitionnement des sous bandes en code-blocs.

Les code-blocs sont en fait des tableaux de coefficients qui peuvent être représentés par un tableau tridimensionnel binaire constitué de plusieurs plans de bits. La figure (III-8) illustre comment les coefficients dans chaque code-bloc sont rangés en plans de bits.

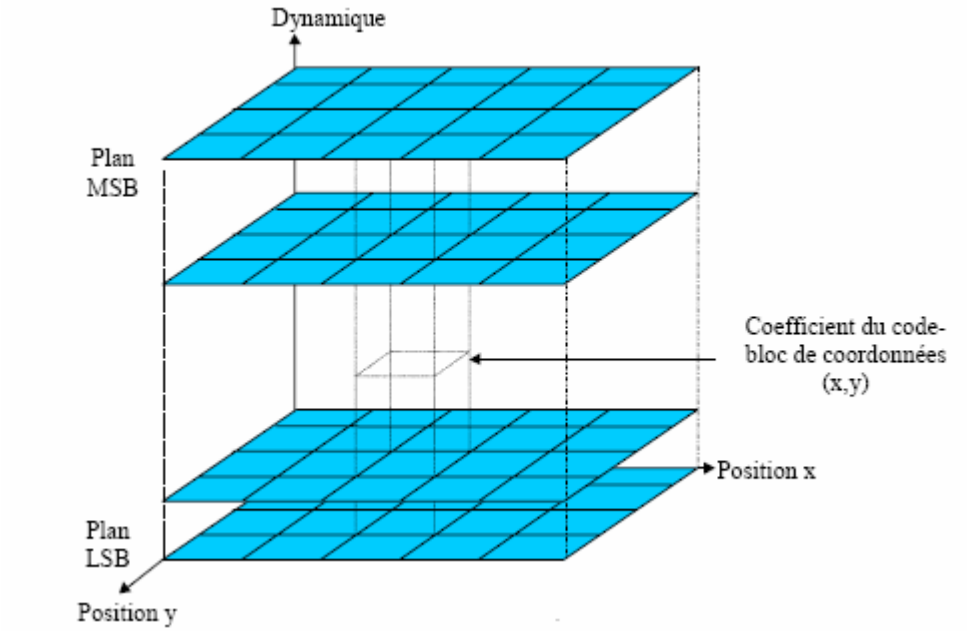


Figure (III-8) : Représentation d'un code –bloc en plan de bits.

Pour mieux illustrer la représentation d'un code-bloc en plans de bits, nous donnons un exemple pratique comme le montre la figure (III-9) , Le code-bloc utilisé est un bloc de

3 x 3 coefficients et sa représentation en plans de bits est faite sur trois plans de bits.

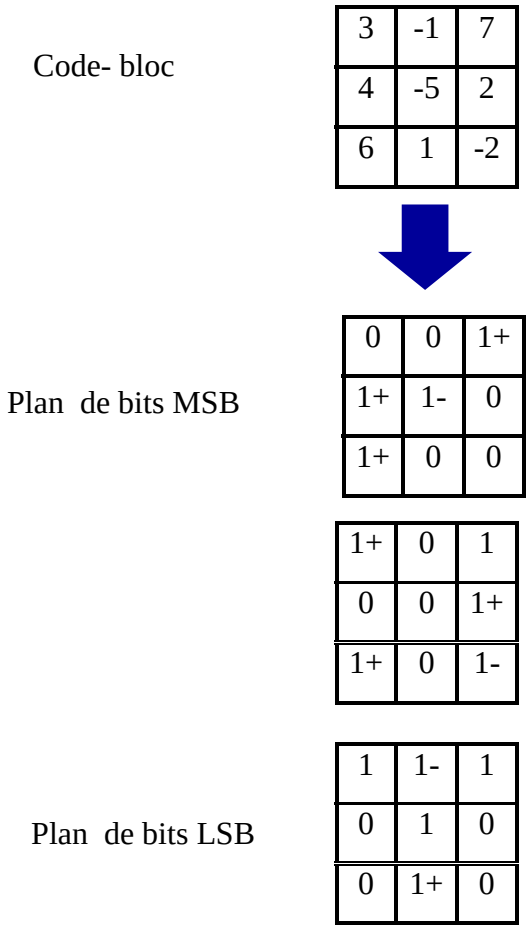


Figure (III-9) : Représentation d'un code-bloc de 3x3 coefficients en trois plans de bits.

Pour le codage des bits des plans de bits. Chaque plan de bits est balayé par bande de quatre bits de haut de gauche vers la droite et **stripe** par **stripe** de haut en bas comme le montre la figure (III-10) :

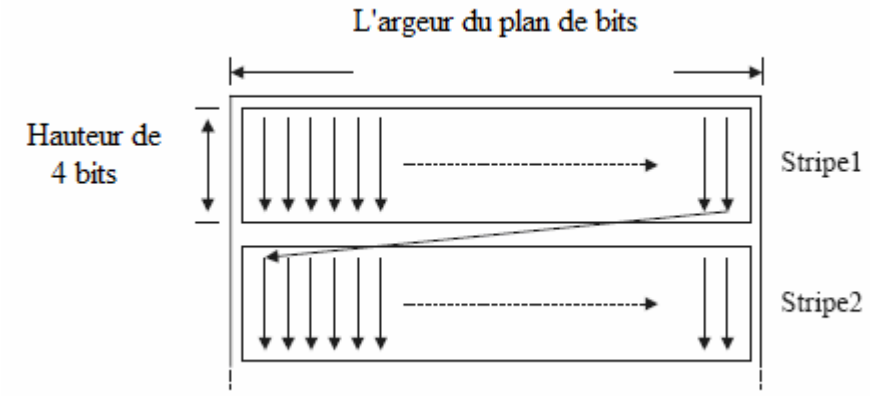


Figure (III-10) : Ordre de balayage d'un plan de bits.

Notons qu'un coefficient est dit significatif si le bit de poids le plus fort de ce coefficient (MSB du coefficient) a déjà été trouvé est codé et égal à 1 si non il est considéré non significatif. A chaque coefficient d'un code-bloc est associé une variable binaire d'état appelée 'signifiance _state' [41]. Cette variable est initialisée à '0' au début de chaque codage de code-bloc. Cette variable devient vraie '1' (coefficient significatif) quand le premier bit à 1 est trouvé (MSB du coefficient). Les plans de bits sont parcourus un à un en partant du plan de bits MSB vers le plan de bits LSB. Le nombre de plans de bits à partir du plan MSB qui sont nuls est signalé dans l'en-tête du paquet. Tous ces plans de bits à partir du plan de bits non nuls seront codés par trois passes successives qui sont : **Passe de signification (significance pass), Passe d'affinage (refinement pass), Passe de nettoyage (cleanup pass)** mise à part le premier plan de bits rencontré signifiant (Un bit est dit signifiant si il est égal à '1' et insignifiant si il est égal à zéro '0') qui sera codé seulement avec la **cleanup pass**. Chaque bit d'un plan de bits est codé une seule fois par l'une des trois passes. Pour chaque passe de codage, des contextes vont être créés pour les bits codés. Pour chaque bit de coefficient dans un plan de bits , on définit également son contexte (CX) associé qui se réfère à l'état de 'signifiance _state' de ses huit coefficients voisins plus proches (D, V, H : pour respectivement coefficients en Diagonal et en Vertical et en Horizontal) [41] (Voir la figure (III-11)). Tous les coefficients voisins de l'extérieur d'un code-bloc sont considérés non signifiants.

D	V	D
H	**X**	H
D	V	D

Figure (III-11) : Etats de voisins employés pour former le contexte de bit de coefficient X.

Le nombre total des contextes créés par les trois passes de codage est 19 [0 à 18] (voir tableau (III-4)).

Différents type de contextes	Valeur du contexte (CX)
Contexte significance pass	0 1 2 3 4 5 6 7 8
Contexte signe	9 10 11 12 13
Contexte refinement pass	14 15 16
Contexte run-length	17
Contexte UNIFORM	18

Tableau (III-4) : Différents contextes utilisés en fonction de type de codage.

Les bits issus de ces différentes passes, ainsi que leurs contextes associés, sont ensuite envoyés à un codeur arithmétique MQ.

III-7-1-1. Codage Passe de signification

Le codage des bits dans la Passe de signification inclut uniquement les bits de coefficients dits insignifiants et ayant des contextes différents de zéro, c.a.d ayant au moins un voisin signifiant tous les autres coefficients sont ignorés. Les huit coefficients voisins entourant un bit de coefficient sont utilisés pour créer le contexte de bit à coder. Les contextes sont déterminés à partir des sommations de 'signifiance _state' de ces coefficients. Le tableau (III-5) résume les résultats de ces contextes. Nous remarquons que le contexte prend en compte le type de sous-bande. En effet, la probabilité d'avoir par exemple un bit voisin en horizontal sera différente pour une sous-bande LH et HL. Neuf (09) contextes (de 0 à 8) sont alors générés dans cette passe de codage comme le montre le tableau (III-5).

Sous - bandes LL et LH (passe- haut vertical)			Sous - bandes HL (passe- haut horizontal)			Sous - bandes HH (passe- haut diagonal)		contextes
ΣH	ΣV	ΣD	Σ H	ΣV	ΣD	Σ (H+V)	ΣD	
2	x	x	x	2	x	x	≥ 3	8
1	≥ 1	x	≥ 1	1	x	≥ 1	2	7
1	0	≥ 1	0	1	≥ 1	0	2	6
1	0	0	0	1	0	≥ 2	1	5
0	2	x	2	0	x	1	1	4
0	1	x	1	0	x	0	1	3
0	0	≥ 2	0	0	≥ 2	≥ 2	0	2
0	0	1	0	0	1	1	0	1
0	0	0	0	0	0	0	0	0

X : signifie quel que soit sa valeur n'est pas prise en considération.

Tableau (III-5) : Contextes utilisé dans le codage passe de signification.

Si la valeur du bit MSB du coefficient est trouvée égal à '1', alors la 'signifiance _state' est mise à '1' et le bit suivant immédiat à être codé est le signe du coefficient. Si non, la 'signifiance _state' demeure'0'.

III-7-1-1-1. Codage et décodage du signe

Le contexte pour le codage du signe est déterminé en deux étapes :

Dans la première étape, la contribution des voisins en vertical et horizontal sont calculées (voir tableau (III-6)). La contribution diagonale n'est pas utilisée.

V_0 (ou H_0)	V_1 (ou H_1)	V (ou H) contribution
signifiant, positive	Signifiant , positive	1
signifiant, négative	signifiant, positive	0
insignifiant	signifiant, positive	1
signifiant, positive	signifiant, négative	0
signifiant, négative	signifiant, négative	-1
insignifiant	signifiant, négative	-1
signifiant, positive	insignifiant	1
signifiant, négative	insignifiant	-1
insignifiant	insignifiant	0

Tableau (III-6) : détermination de la contribution verticale et horizontale pour le codage du bit de signe.

Chaque voisin peut être dans un des trois états: signifiant positive, signifiant négative ou insignifiant.

La deuxième étape, permet de réduire les neuf permutations de contribution verticale et horizontale en 5 contextes (voir tableau (III-7)).

Contribution horizontale (H)	Contribution verticale (V)	contextes	XORBIT
1	1	13	0
1	0	12	0
1	-1	11	0
0	1	10	0
0	0	9	0
0	-1	10	1
-1	1	11	1
-1	0	12	1
-1	-1	13	1

Tableau (III-7) : Table de contextes de codage de signe.

Ces contextes sont envoyés au codeur arithmétique suivant l'opération suivante :

$$CX = Signebit \oplus XORBIT$$

Avec :

Signebit est le signe de bit à coder et $\oplus$ représente l'opération (ou exclusif binaire) et XORBIT sont utilisation est la même dans l'encodeur et le décodeur.

La valeur du Signebit décodé par le décodeur arithmétique MQ est donnée par l'opération inverse :

$$Signebit = CX \oplus XORBIT$$

III-7-1-2. Codage passe d'affinage

Cette passe de codage est la plus simple. Seuls les bits de coefficients déjà signifiants dans le plan de bits précédent sont codés. Le signe du bit est donc déjà connu. Il y aura seulement trois contextes possibles. Le contexte utilisé est déterminé par la sommation des ' signifiance _ state ' des huit coefficients voisins horizontaux, verticaux et diagonaux. Tableau (III-8) présente ces contextes.

ΣH + ΣV + ΣD	Première codage pour ce coefficient par passe d'affinage	contextes
X	Faux	16
≥1	vraie	15
0	vraie	14

X : signifie quel que soit sa valeur n'est pas prise en considération.

Tableau (III-8) : Table des contextes utilisé pour codage passe d'affinage.

III-7-1-3. Codage passe de nettoyage [42]

La cleanup pass code tous les bits d'un plan qui n'ont pas été codés par les deux autres passes. C'est à dire les bits de coefficients insignifiants qui ont un contexte égal à zéro (pas de voisins signifiants).Ils ont en général une probabilité très faible et restent souvent insignifiants. C'est pour cela qu'un mode spécial, le run-length, est utilisé pour rassembler les coefficients qui ont la plus grande probabilité de rester non-significatifs. Ce mode est utilisé lorsque tous les quatre coefficients d'une colonne ne connaissent pas de coefficients signifiants. De plus, ne connaissent pas de coefficients voisins à cette colonne qui soient signifiants. Ces deux conditions réunies, le codeur passe en mode run-length. Dans ce mode, un symbole binaire est encodé par le codueur MQ avec un seul contexte pour spécifier si ces quatre coefficients restent non- significatifs. Si la valeur encodée est 0 avec le contexte 17, cela implique qu'ils le demeurent. Si un 1 est encodé avec le contexte dit UNIFORM (18), cela signifie qu'au moins un des quatre est significatif dans ce plan de bits. Dans ce cas, deux symboles supplémentaires sont encodés pour spécifier l'emplacement du premier coefficient non-nul dans la colonne (00 pour le premier, 01 pour le second, 10 pour le troisième et 11 pour le quatrième). Cela permet de signaler au décodeur la position exacte de sortie du mode runlegth. Après avoir spécifié la position du premier coefficient non nul, les coefficients restant dans la

colonne sont encodés de la même manière que dans la significance pass et avec les même contextes.

III-7-2. Le codeur arithmétique (MQ)

Le codeur utilisé dans JPEG 2000 est un **codeur arithmétique binaire adaptatif** dénommé ''MQ''. Le fait qu'il soit **adaptatif** signifie que les probabilités des 2 types de symboles à encoder évoluent au cours du processus de codage. Prend comme entrées les Valeurs binaires et les contextes associes issus de l'étape précédente de modélisation binaire des coefficients, et cela dans l'ordre des passes de codage (figure (III- 12)).

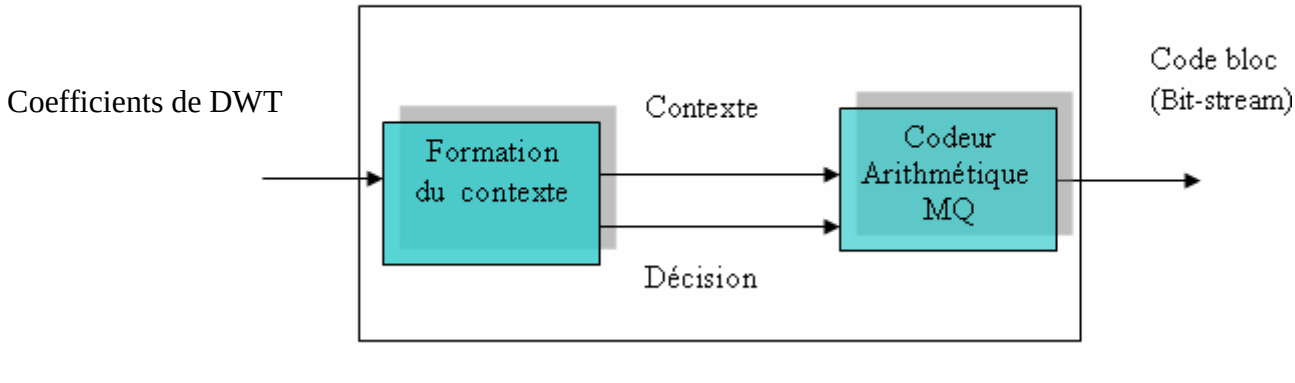


Figure (III- 12) : Entrées et la sortie du codeur arithmétique.

III-7-2-1. Principe de fonctionnement

Le principe du codeur arithmétique est la subdivision d'intervalles correspondant à la Probabilité du symbole LPS (Less Probable Symbol), utilisé dans le code d'Elias [33]. La figure (III- 13) représentée par l'intervalle A que l'on divise alors en deux sous intervalles correspondant à l'espèce minoritaire (LPS) et majoritaire (MPS). D'un point de vue représentation, on donne toujours comme intervalle inférieur pour le symbole LPS et l'intervalle supérieur pour le symbole MPS. L'intervalle associé au symbole LPS doit être petit que celui associé au symbole MPS. Qe est la Probabilité relatif à LPS. Les symboles dans un lancement de code sont classifiés en LPS (Less Probable Symbol) et MPS (More Probable Symbol) qui représentent respectivement les probabilités d'occurrence de l'espèce minoritaire et majoritaire.

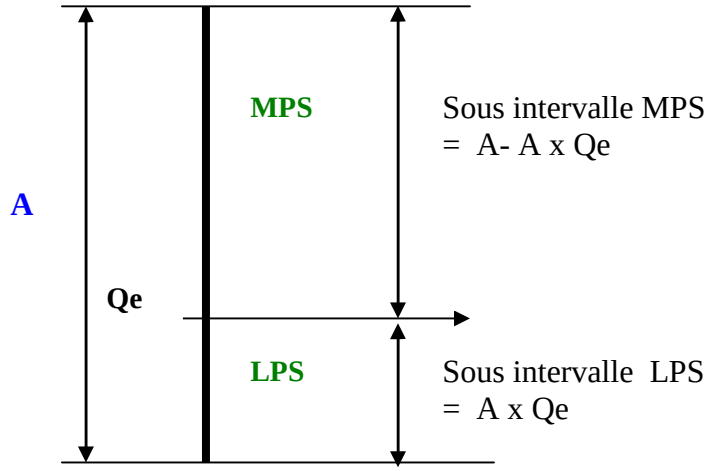


Figure (III- 13) : Conventions utilisées dans le codeur arithmétique.

Le codeur arithmétique utilisé dans JPEG2000 utilise deux registres (A et C) pour coder ces symboles binaires. Le registre A est un registre d'intervalle et le registre C est un registre de code. L'état des registres (A et C) changent et cela au fonction des valeurs des décisions binaires à coder :

1) Dans le cas ou la valeur de la décision à coder est un MPS :

$$A = A - A \times Qe \qquad \text{(III-3)}$$

$$C = C + A \times Qe \qquad \text{(III-4)}$$

Qe : est la probabilité de l'espèce minoritaire (LPS).

2) Dans le cas ou la valeur de la décision à coder est un LPS :

$$A = A \times Qe \qquad \text{(III-5)}$$

$$C = C \qquad \text{(III-6)}$$

Qe : est la probabilité de l'espèce minoritaire (LPS).

Garder A dans l'intervalle $0.75 \leq A < 1.5$ permet une approximation arithmétique simple pour être utilisé dans la subdivision d'intervalles. Comme la moyenne de la largeur de l'intervalle A est approximativement proche de 1, on a l'approximation (Qe x A) = Qe [42], Cette approximation est appliquée dans les équations ((III-3) et (III- 4) et (III-5) et (III-6))

ce qui va simplifier l'implémentation du codeur et du décodeur. L'algorithme devient alors :

1) Dans le cas ou la valeur de la décision est codé est un MPS :

$$A = A - Qe \quad \text{(III-7)}$$

$$C = C + A \quad \text{(III-8)}$$

2) Dans le cas ou la valeur de la décision est codé est un LPS :

$$A = Qe \quad \text{(III-9)}$$

$$C = C \quad \text{(III-10)}$$

La longueur de l'intervalle A est toujours maintenu dans la gamme $0.75 \leq A < 1.5$. Une renormalisation est utilisé à chaque fois que sa valeur devient inférieure à 0.75 en le doublant. Le registre de code C est également doublé à chaque fois que A est doublé.

III-7-2-1-1. Traitement d'échange conditionnel du sens de MPS

Si le codeur continue de recevoir des symboles MPS, la largeur de l'intervalle A va se rétrécire avec (A = A - Qe) ce qui va l'amené parfois à une situation où l'intervalle du LPS devient plus large que celui du MPS. Si par exemple la valeur de Qe est de 0.5 et que A est au minimum permis de 0.75, les équations ((III-7) et (III-9)) donnent 1/3 de l'intervalle de probabilité au MPS et 2/3 au LPS. Dans ce cas et pour éviter une inversion de taille, les intervalles du MPS et du LPS sont échangés et le codage du symbole le moins probable utilisera alors l'intervalle supérieur .

Cette condition d'échange MPS / LPS se produit quand une renormalisation est nécessaire. Chaque fois qu'une renormalisation se produit, un processus d'estimation de probabilité est appliqué pour déterminer une nouvelle estimation de probabilité pour le contexte en cours de codage [43].

III-7-2-2. Principe de création et mise à jour des probabilités

Nous avons vu précédemment que l'EBCOT envoyait au codeur MQ un contexte pour chaque bit codé.Il existe 19 contextes différents. Nous avons supposé jusqu'ici que la probabilité Qe du symbole LPS était connue. Voyons comment le codeur MQ gère cette valeur Qe. Pour chaque couple (symbole (D), contexte (CX)) issus du codeur EBCOT, le codeur MQ avant de coder ce couple associé à chaque contexte (CX) un index (I(CX)) et une valeur MPS(CX) (voir figure (III- 14)) .

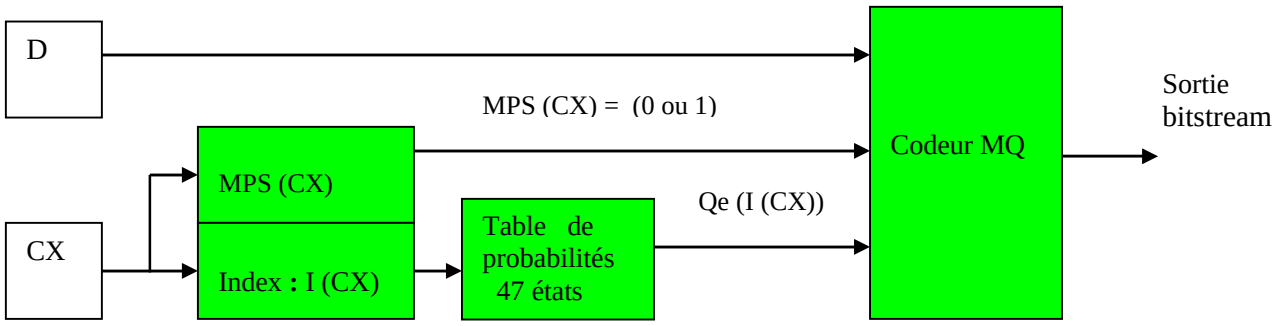


Figure (III- 14) : Création de probabilités liées à un contexte.

L'index (I (CX)): sert à pointer sur une table d'estimation de probabilités Qe(I(CX)) pour estimer la probabilité du contexte (CX) qui sera utilisée pour l'encodage d'un bit de contexte (CX).Dans JPEG2000, cette table comprend 47 états différents ce qui permet au codeur de pourvoir adapter de façon plus rapide la probabilité d'apparition du symbole pour chaque contexte (voir tableau (III-9)) [42].

Index (I(CX))	Qe (I (CX))	NMPS (I(CX))	NLPS (I(CX))	SWITCH (I(CX))
	Décimal			
0	0,503937	1	1	1
1	0,304715	2	6	0
2	0,140650	3	9	0
3	0,063012	4	12	0
4	0,030053	5	29	0
5	0,012474	38	33	0
6	0,503937	7	6	1
45	0,000023	45	43	0
46	0,503937	46	46	0

Tableau (III-9) : Table d'estimation de probabilité du LPS.

L'estimateur (tableau (III-9)) est une table de probabilités Qe(I(CX)) qui correspondant aux

symboles LPS et de prochains états de probabilités associés pour chaque type de renormalisation (c - à - d., nouvelles positions de table) **.** Le changement de l'état de probabilité se produit seulement quand le codeur arithmétique renormalise le registre d'intervalle.Ceci est toujours fait après codage des LPS, et toutes les fois que le registre d'intervalle est moins que 0.75 (dans la notation décimale) après codage du MPS. Après une renormalisation de LPS, NLPS (I(CX)) donne le nouvel index pour l'évaluation de probabilité de LPS. Après une renormalisation de MPS, NMPS (I(CX)) donne le nouvel index pour l'évaluation de probabilité de LPS. Si SWITCH (I (CX)) est égal à 1, alors le sens du symbole MPS (CX) est inversé (MPS (CX)= 1- MPS (CX)).

Lors du démarrage du codeur MQ, les informations de chaque contexte doivent être initialisées. La norme JPEG 2000 définit donc pour chaque contexte un état de départ dans la table d'estimation de probabilité (pour la majorité des contextes, il s'agit de l'état 0) ainsi qu'une valeur pour le MPS (CX) (il s'agit toujours de la valeur "0") (voir tableau (III-10) **)**.

Contextes	**index initial de la Table d'état**	**MPS(CX) inital**
UNIFORM(18)	46	0
Run-length (17)	3	0
Contexte zero (0)	4	0
Tout les autres contextes	0	0

Tableau (III-10) : Table des valeurs initiales des contextes au démarrage de codage.

Le MPS (CX) : prend comme valeur (0 ou 1) , sert à déterminer la valeur du symbole le plus probable ou le moins probable pour chaque symbole .Si par exemple on a la valeur du MPS (CX) = 0 et la valeur du symbole à coder (D = 0) alors le codeur arithmétique décide de coder le symbole (D = 0) comme un MPS (symbole le plus probable). Si la valeur du MPS(CX) diffère de la valeur du symbole (D) le codeur MQ décide de coder le symbole comme un LPS (symbole le moins probable).

III-7-2-3. Description du codeur arithmétique JPEG2000 [42]

Le codeur arithmétique (figure (III-15)). INITENC initialise les registres A= 0,75 et C= 0 et la variable du compteur de bit (CT = 12). Après une étape d'initialisation, le codeur codera tous les couples (D, CX), données et contextes qui lui sont présentés. FLUSH permet de vider les registres A et C, et d'ajouter un marqueur de terminaison.

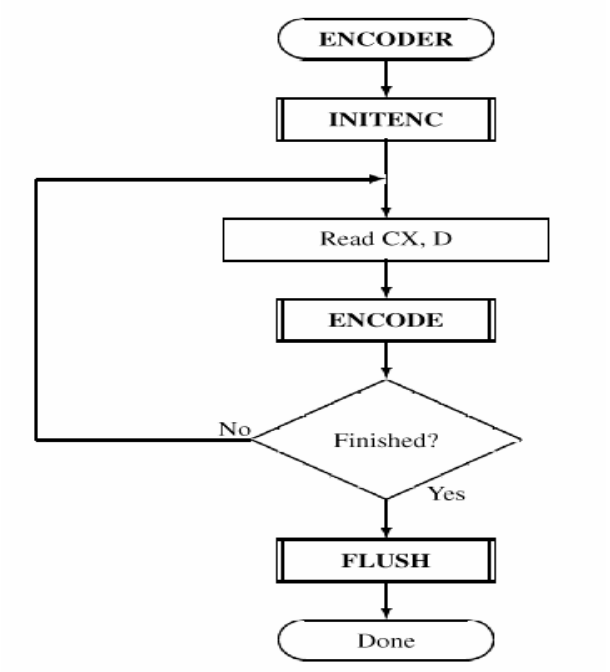


Figure (III- 15) : Schéma général du codeur MQ.

III-7-2-3-1. Les registres du codeur JPEG2000

La structure du registre A et C comme elle a été définie dans la norme JPEG 2000 est donné dans le tableau (III-11).

	MSB			LSB
Registre C	0000cbbb	bbbbbsss	xxxxxxxx	xxxxxxxx
Registre A	00000000	00000000	aaaaaaaa	aaaaaaaa

Tableau (III-11) : Structures des registres A et C à l'encodage.

Ces registres C et A occupent respectivement 32 et 16 bits et sont organisés de la manière indiquée dans le tableau (III-11). Les 16 bits bas de chacun des registres servent aux opérations arithmétiques. Le bit **c** dans le registre C sert à stocker le carry issu de ces opérations. Les bits **b** sont "remplis" à mesure que des décalages vers la gauche interviennent dans le registre C. les bits **s** bits d'espacement permettant de retarder la retenue 'carry'. Les bits **x** et **a** : bits de fraction dans le registre de code C et d'intervalle A. dés que les huit emplacements "**b**" sont occupés par des valeurs de Ci, on extrait l'ensemble de 9 bits "cbbb bbbb b" pour le placer à la suite des bits déjà codés.

III-7-2-3-2. Codage d'une décision (0 ou 1)

Le procédé de Codage (ENCODE) détermine si la décision D est un 0 ou pas. Puis un CODE0 ou un procédé CODE1 s'appelle convenablement (figure (III-16)).

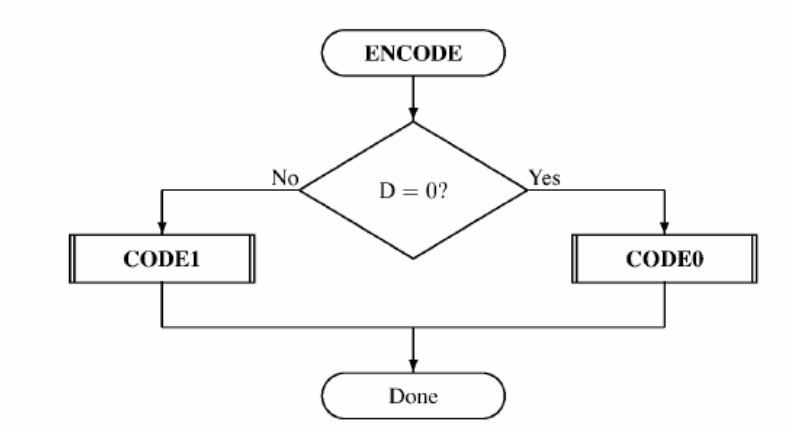


Figure (III-16) : La procédure d'Encodage.

III-7-2-3-3. Codage de la décision comme un MPS ou LPS

Le codeur code des symboles (MPS et LPS) plus ou moins probable et non des valeurs (D = 0 ou 1). Pour chaque contexte, la variable MPS (CX) précisera la valeur du symbole le plus probable (1 ou 0). CODE1 et CODE0 sont illustrés dans la figure (III-17) et la figure (III-18). Dans ces figures, CX est le contexte. Pour chaque contexte, l'index de l'évaluation de probabilité qui est d'être employée dans les opérations de codage et la valeur de MPS (CX) sont stockés dans le codeur.

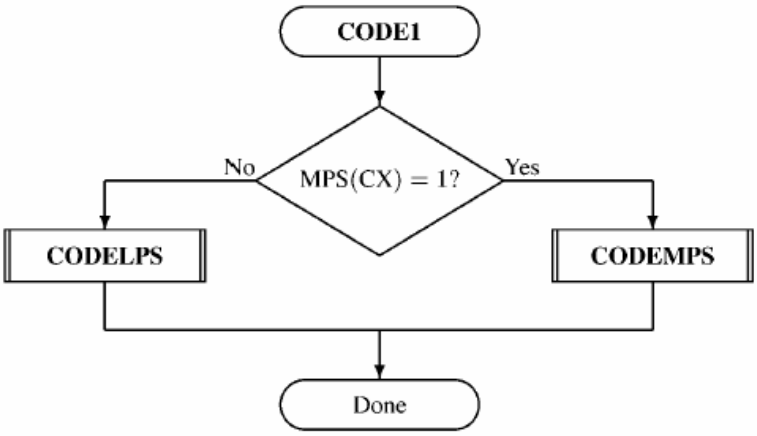


Figure (III-17) : procédure CODE1.

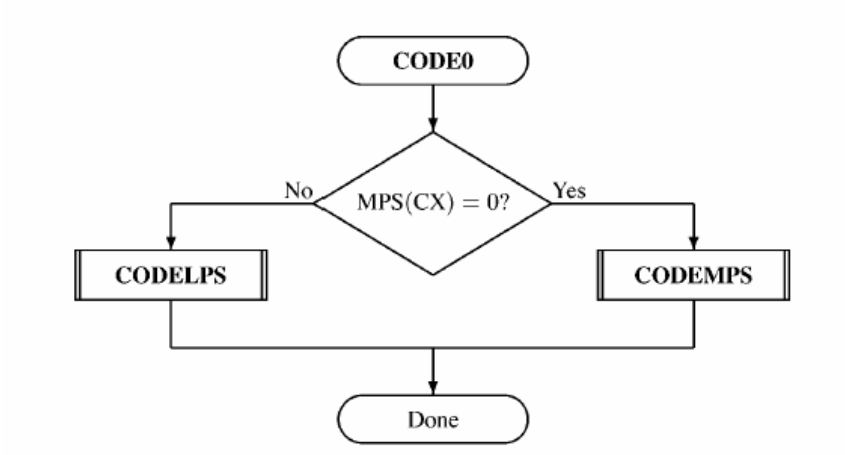


Figure (III-18) : procédure CODE0.

III-7-2-3-4. Procédure de renormalisation des intervalles

Il y aura décalage des registres A et C d'un bit à gauche chaque fois que la valeur de l'intervalle A sera inférieure à 0,75 afin de garder une bonne précision de l'intervalle A. La fonction RENORME (figure (III-19)) effectue ces décalages et décrémente le compteur de bits CT. Cette fonction est appelée par CODELPS et par CODEMPS [42].

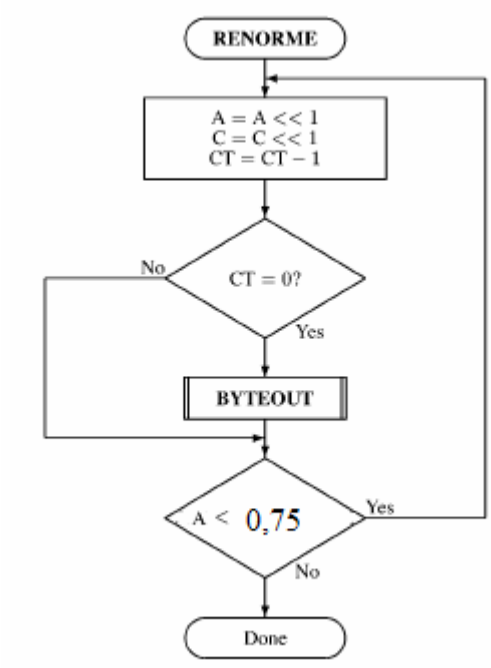


Figure (III-19) : La procédure de renormalisation des Intervalles.

Si CT=0, la fonction BYTEOUT transfère les bits 'cbb…bb' du registre de code C dans le fichier binaire (codestream) sous forme d'un octet (figure (III-20)) à l'aide du pointeur BP.
.

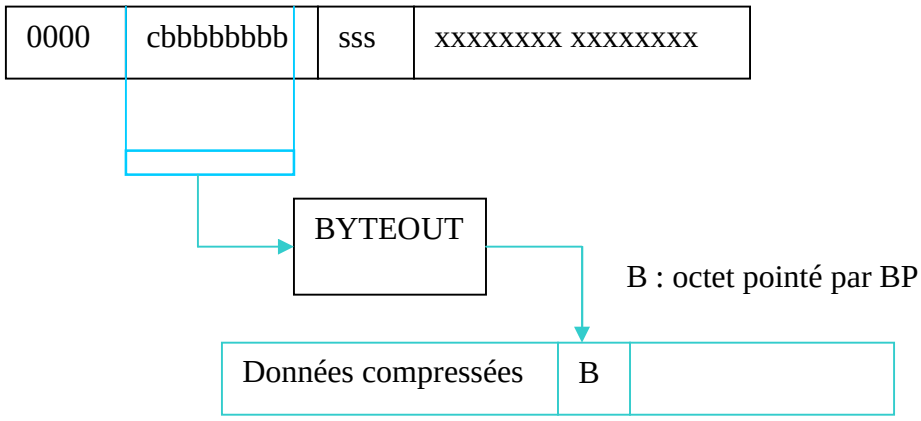


Figure (III-20) : La procédure de transfère.

Remarque

La multiplication par 2 en binaire s'effectue par un décalage des bits du coefficient à multiplier d'un bit vers la gauche.

III-7-2-4. Description du décodeur arithmétique JPEG2000 [42]

La figure (III- 21) montre un schéma fonctionnel simple d'un décodeur arithmétique adaptatif binaire. Après initialisation du décodeur par INITDEC. Les données comprimées et les contextes (CX) stockés pour les bits codés sont chargés au décodeur. Muni de ces contextes, il peut déterminer la probabilité Qe associée à chaque bit. La routine de DECODE décode la décision binaire D et renvoie une valeur de 0 ou de 1.

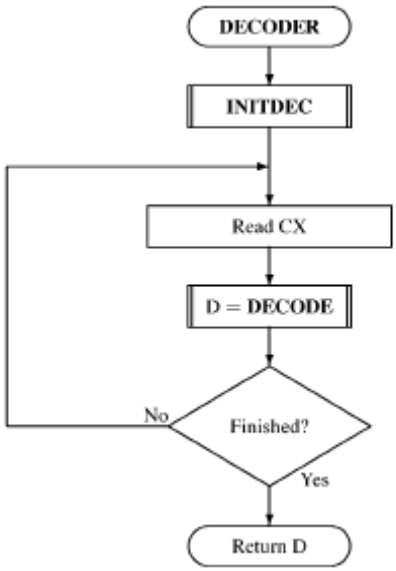


Figure (III- 21) : Schéma général du décodeur MQ.

III-8. Codage Tier 2 (Organisation du bitstream)

A ce stade, La séquence binaire issue du MQ est divisée en un certain nombre de paquets. Chacun d'eux contient le bitstream correspondant à une même composante, à un même niveau de résolution, à une même couche de qualité et à une même zone spatiale du niveau de résolution [33]. Chacun de ces paquets est précédé d'un en-tête contenant des informations permettant d'identifier très précisément les données véhiculées par ce paquet. L'ordre dans lequel les paquets apparaissent dans le codestream est appelé ordre de progression [38], [39]. Quatre ordres de progression différents sont définis dans la norme JPEG2000. Ils permettent lors du décodage d'obtenir en priorité soit les données d'une même composante, soit celle d'une progression en niveau de résolution, soit celles d'une progression en couche de qualité, soit celles d'une même zone spatiale de l'image. Dans le cas d'une compression avec perte, certains passes de codage seront écartés c.a.d. ne seront pas inclus dans les couches de qualité, dans ce cas le contrôleur de taux décide quels passes va être inclus dans le flux de données final (codestream). L'allocation de débit consiste à trouver, dans la séquence correspondant à chaque code-bloc, le point de troncature optimal permettant d'atteindre le débit spécifié dans les paramètres de codage [38].

Une particularité de JPEG 2000 est que plusieurs débits peuvent être spécifiés au codage. Dans ce cas, la séquence binaire (bitstream) de chaque code-bloc sera coupée en plusieurs endroits. La réunion des premières portions des séquences binaires de chaque code-bloc forme une première couche (layer) de qualité. L'ajout des secondes portions de chaque code-bloc forme la

deuxième couche apportant ainsi un incrément de qualité à la totalité de l'image (ou tile), et ainsi de suite (voire figure (III-22)) [33]. Et on peut remarquer que dans certaines couches le bitstream d'un code bloc peut être vide. La totalité du flux binaire codé est ordonnée pour former le code-stream [42]. Ce dernier est ensuite encapsulé dans un fichier au format JP2 selon les spécifications de la norme.

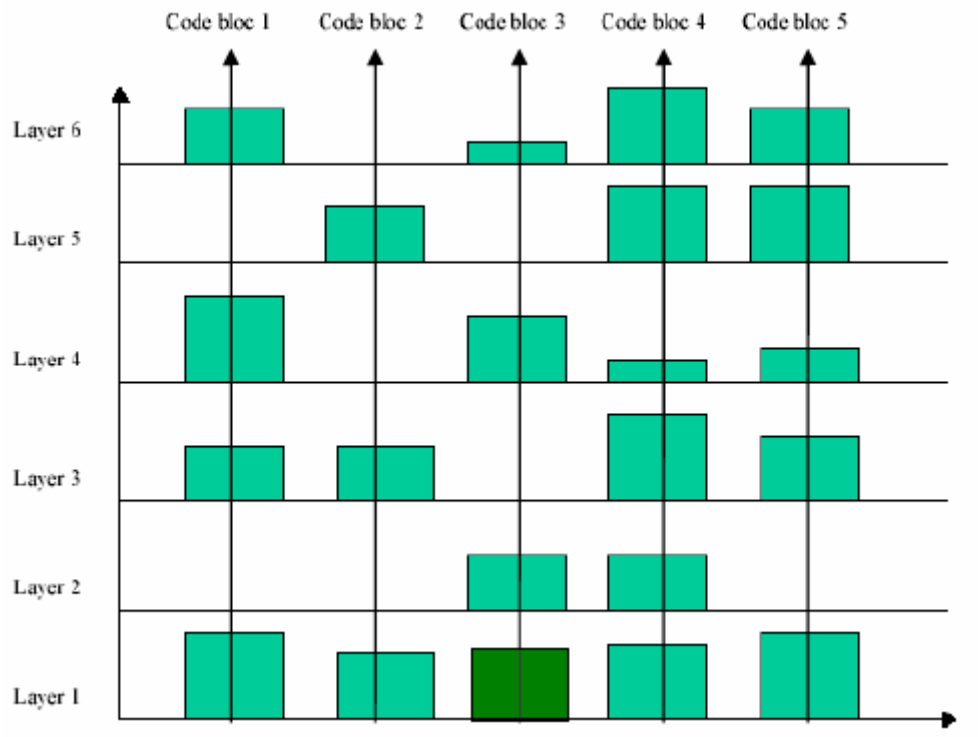


Figure (III-22) : Illustration du concept de couches de qualité.

La figure (III-23) montre un exemple d'organisation d'un fichier JPEG2000 d'une image constituée d'une seule composante compressée sans utilisation de tiles à trois couches de qualité :

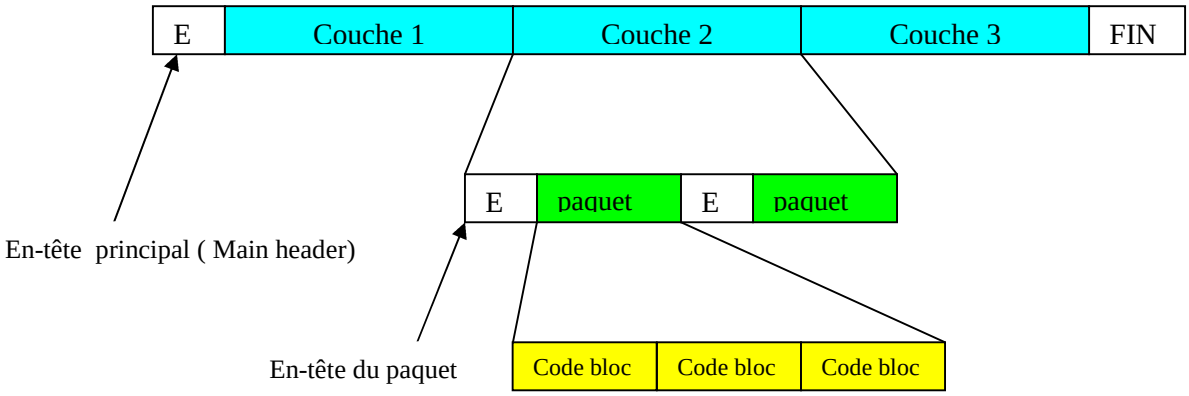


Figure (III-23) : Organisation d'un fichier JPEG2000.

III-9. Régions d'intérêt

Une région d'intérêt (ROI pour Region of Interest) est une région de l'image qui est codée avec une plus grande précision, en général parce que cette région présente un intérêt particulier (ex : visage, plaque d'immatriculation, …etc.). Cette plus grande précision se fait au détriment des autres zones de l'image qui sont alors compressées à un taux inférieur et donc dégradées [44], [45].

Certains coefficients après la transformée en ondelettes sont identifies comme appartenant à la région d'intérêt, les autres appartenant à la région de moindre intérêt appelée arrière plan. Le codage de la région d'intérêt s'opère entre la quantification scalaire et le codage entropique. JPEG2000 utilise la méthode dite MaxShift. Le principe est de surélever de la valeur Max de tous les coefficients correspondant à la région d'intérêt. Cette valeur Max est appelée facteur d'échelle est notée 'S' comme le montre la figure (III-24). On a ainsi augmenté artificiellement la dynamique des coefficients de la région d'intérêt et donc leur importance en terme de signal. La forme de la région d'intérêt n'est pas signalée au décodeur, seulement la présence d'une région d'intérêt et la valeur du décalage opéré sont signalées. Au décodage, les données seront redécalées de façon à retrouver la dynamique d'origine.

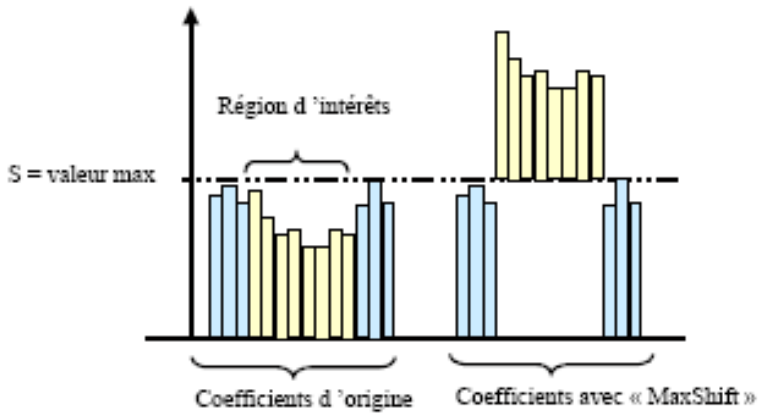


Figure (III-24) : Principe d'élévation des coefficients correspondant à une région d'intérêt.

III-10. Le contrôle de taux [32]

Il s'agit ici de fixer un taux de bits que le codeur ne doit pas dépasser en respectant la contrainte d'une distorsion minimale. Dans le codeur, le contrôle du taux est accompli par deux mécanismes distincts :

• Choix des pas de quantification : il fixe des pas de quantification plus ou moins grands afin d'arriver au débit binaire voulu. Cependant cette méthode n'est pas la meilleure parce qu'à chaque fois que l'on modifie le pas de quantification il faut refaire intégralement l' étage de codage.

• Sélection des passes de codage qui seront inclus dans le flux de données final (codestream): cette méthode est la meilleure car elle permet d'enlever rapidement des données superflues et elle n'affecte que le deuxième étage de codage. Le standard ne spécifie pas comment ces mécanismes devraient être employés, il est possible d'utiliser l'un ou l'autre ou même tous les deux ensembles.

III-11. Résistance aux erreurs

Une originalité de JPEG 2000 est d'inclure des outils de résistance aux erreurs de transmission. Le problème vient essentiellement du codeur arithmétique car un seul bit erroné entraîne le décodage d'une mauvaise séquence.Par défaut, le codage arithmétique agit en effet sur un code-bloc. En cas d'erreur (un seul bit erroné suffit) c'est l'ensemble du code-bloc qui est perdu. Afin de limiter les effets de ces erreurs, la norme propose plusieurs outils, dont la philosophie est essentiellement de compartimenter les mots de codes ou de réduire leur longueur afin d'éviter la propagation des erreurs ou de limiter leurs effets.Les outils proposés sont les suivants :

Marqueur de segment

Ces marqueurs sont insérés après chaque plan de bits et sont codés arithmétiquement. Leur bon décodage indique que le plan de bit courant a été correctement décodé. Inversement, si le marqueur n'est pas trouvé, le plan de bits sera considéré comme erroné et donc supprimé.

Terminaison à chaque passe

C'est un moyen de limiter la propagation des erreurs, en compartimentant les données de façon fine (à chaque passe). Le décodeur arithmétique peut ainsi continuer le décodage en cas d'erreur.

Marqueur de resynchronisation (SOP/EPH)

Ces deux marqueurs indiquent le début et la fin de chaque paquet et permettent au décodeur de se synchroniser grâce au numéro de paquet inclus dans ces marqueurs. Il s'agit ici de marqueurs de syntaxe (non codés arithmétiquement) et dont la gestion est propre à chaque décodeur.

IV-12. Conclusion

Dans ce chapitre nous avons fait un tour d'horizon des principales fonctionnalités proposées dans JPEG2000 partie I. Nous avons pu voir que JPEG2000 possède tous les outils nécessaires pour devenir une norme universelle. Parmi les caractéristiques principales de JPEG2000 sont les suivantes:

• Compression avec ou sans perte dans le même algorithme.

• Accès aléatoire aux données.

• Progressivité en qualité ou en résolution ou les deux en même temps.

• Si l'image source est multi- composantes, chaque composante sera traitée de manière indépendante. Par exemple, une image couleur RVB sera décomposée et convertie en composante YUV.

• Les images sont découpées (optionnel) en imagettes rectangulaires dites tiles.La composante imagette est l'unité de base de l'image originale ou reconstruite.

• Une transformation en ondelettes est appliquée sur chaque imagette. Une imagette (tile) est décomposée en differents niveaux de résolution.

• Les niveaux de décomposition d'ondelettes sont en formes de sous-bandes des coefficients qui décrivent les caractéristiques de fréquence des zones locales des composantes imagettes.

• Les coefficients des sous-bandes sont quantifies et classes dans des blocs de coefficients (code-blocs).

• Les plans de bits des coefficients dans un code-bloc sont codés entropiquement.

• Le codage peut être réalisé de telle manière que certaines régions d'intérêt peuvent être codées à une meilleure qualité que le fond de l'image.

• Des marqueurs sont ajoutées au flux de bits (bitstream) pour permettre de créer une meilleur résistance aux erreurs de transmission.

• Le code-stream possède un en-tête principal qui décrit l'image originale et les différents modèles de décomposition et codage utilisés pour localiser, extraire, décoder et reconstruire l'image avec la résolution désirée, qualité souhaitée, les régions d'intérêt et autres caractéristiques.

Chapitre IV

Etude de performance de la technique de compression JPEG2000 par rapport à JPEG

IV-1. Introduction

Ce chapitre vise à étudier les performances de la norme de compression JPEG2000 par rapport à la norme de compression classique JPEG. Pour réaliser cette étude, nous utilisons le logiciel libre "Kakadu", comme une implémentation du JPEG2000 [46]. Ce logiciel est écrit en utilisant le langage de programmation C++. Pour le JPEG, nous avons utilisé le logiciel libre développé par le Groupe Indépendant de JPEG (Indépendant JPEG Group) qui est écrit en utilisant le langage de programmation C [47]. La compilation des fichiers contenant les codes sources des normes JPEG2000 et JPEG, avec le Compilateur Microsoft Visual C++ version 6, nous donne des fichiers exécutables sous DOS. Ces fichiers nous permettent de réaliser la compression et décompression des images. Des Programmes de liaison et des fonctions sous Matlab ont été développés afin de faciliter la compression et la décompression des images et l'évaluation des résultats de tests sur les différentes images utilisées.

IV-2. Description des programmes et fonctions utilisés dans Matlab

Les programmes réalisés sous Matlab pour évaluer la compression par les deux normes JPEG2000 et JPEG peuvent être résumé en trois blocs essentiels comme le montre la figure (IV-1) :

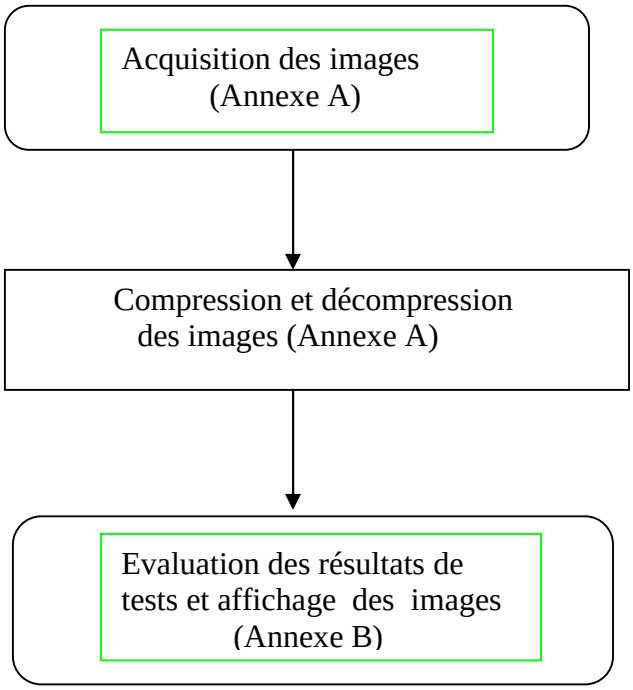


Figure (IV-1) : Principe des programmes développés.

Pour l' évaluation des tests de la compression, les fichiers exécutables sous Dos de ces codecs et les images de tests ainsi que les programmes développées sous Matlab doivent être placés dans le dossier work de Matlab.

IV-2-1. Bloc Acquisition des images (Annexe A)

Dans cette partie. Les fonctions réalisées permettent de saisir le nom de l'image à compresser (d'origine), le nom de l'image compressée, le nom de l'image décompressée (reconstruite) et la ligne de commande de chaque codec utilisé.

IV-2-2. Bloc compression et décompression des images (Annexe A)

Ce sont des programmes qui servent de passerelle entre Matlab et DOS. Ils vont exécuter les fichiers exécutables des codecs sous Dos et retournent les résultats dans le dossier work de Matlab.

IV-2-3. Bloc évaluation des résultats de tests et affichage des images (Annexe B)

Dans ce bloc, Pour chaque image en niveau de gris ou pour chaque composante d'image en cas d'image couleur, les programmes réalisés permettent :

1) D'afficher en même temps sur l'écran de l'ordinateur l'image à compresser (d'origine), l'image décompressée (reconstruite) et l'image d'erreur (différence entre image (origine et son image reconstruite)) avec leurs histogrammes respectifs.

2) De calculer les paramètres suivants :

1- PSNR et SNR et QEM de chaque composante image d'origine et reconstruite.

2- Taux de bits (Tb) en bits par pixel (bpp).

3- Quotient (Taille image originale/Taille image compressée).

4- Taille de l'image à compresser et taille de l'image compressée en Octets.

5- Informe l'utilisateur de type de format d'image utilisé.

Remarque

Ce programme réalisé est valable uniquement pour les images en niveau de gris codées avec 8 bits par pixel et couleurs codées avec 24 bpp. Les logiciels kakadu et celui du groupe JPEG travaillent mieux avec des images en niveau de gris de format .pgm et de format couleur de type .ppm.

IV-3. Tests d'applications

Rappel

PSNR (Peak Signal to Noise Ratio) : C'est le rapport signal sur bruit crête qui est très utilisé en traitement d'images pour témoigner de la qualité de restitution, notamment en compression des images. Le PSNR est très facile à calculer, il est basé sur l 'erreur quadratique moyenne entre l'image originale et l'image reconstruite et s'exprime dans le cas des images en niveaux de gris par :

$$PSNR = 10 \times \log_{10}\left(\frac{(255)^2}{EQM}\right) dB \qquad \text{(IV -1)}$$

$$EQM = \frac{1}{M \times N} \times \sum_{n=1}^{N}\sum_{m=1}^{M}\left(I(n,m) - \tilde{I}(n,m)\right)^2 \qquad \text{(IV - 2)}$$

Où

EQM : est l'erreur quadratique moyenne.

N × M : est la taille de l'image.

$I(n,m)$: désigne la valeur du pixel de l'image d'origine.

$\tilde{I}(n,m)$: désigne la valeur du pixel de l'image reconstruite.

Il faut avoir à l'esprit que le PSNR n'est pas une mesure rigoureuse de la qualité visuelle d'une image reconstruite car l'erreur quadratique est une mesure globale qui ne renseigne aucunement sur les distorsions locales (disparition de détails, dégradation des contours …) La validité du PSNR est relative et dépend donc du type d'utilisation envisagée des image traités. A défaut d'un critère général quantifiant la qualité perceptuelle, le PSNR demeure malgré tout très utilisé, l'aspect visuel peut être très apprécié directement par des observateurs humains.

IV-3-1. Tests d'images au niveau de gris

Pour l'évaluation de la compression au moyen des techniques JPEG et JPEG2000, nous allons tout d'abord appliqué le test sur une image de référence « Lena », image très utilisée en compression d'image, de résolution 512 * 512 pixels et codée sur 256 niveaux de gris (voir figure (IV-2)) et ensuite sur trois autre images : Boat, Barbara, Baboon aussi de résolution 512 * 512 pixels codée sur 256 niveaux de gris pour chaque une (Voir (figure (IV-3), figure (IV-4) et figure (IV-5))).

Figure (IV-2) : Image originale ‘Lena .PGM’.

Figure (IV-3) : Image originale ‘Boat .PGM’.

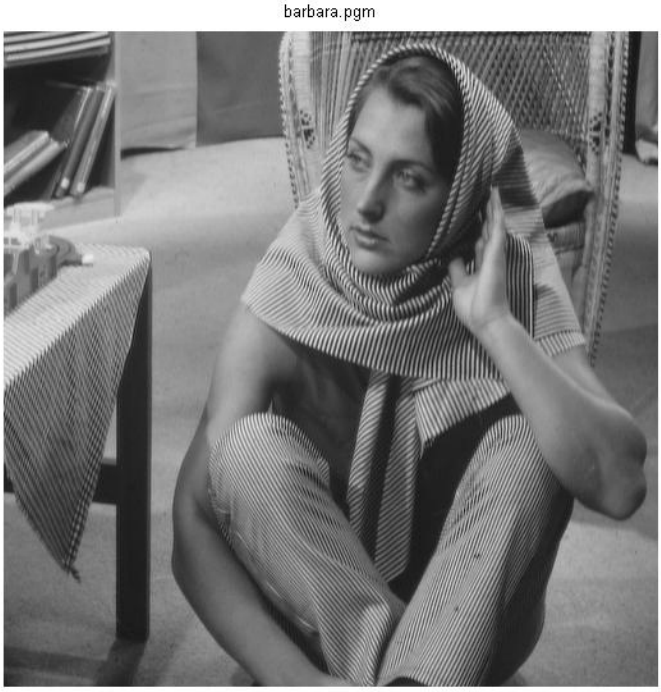


Figure (IV-4) : Image originale ‘Barbara .PGM’.

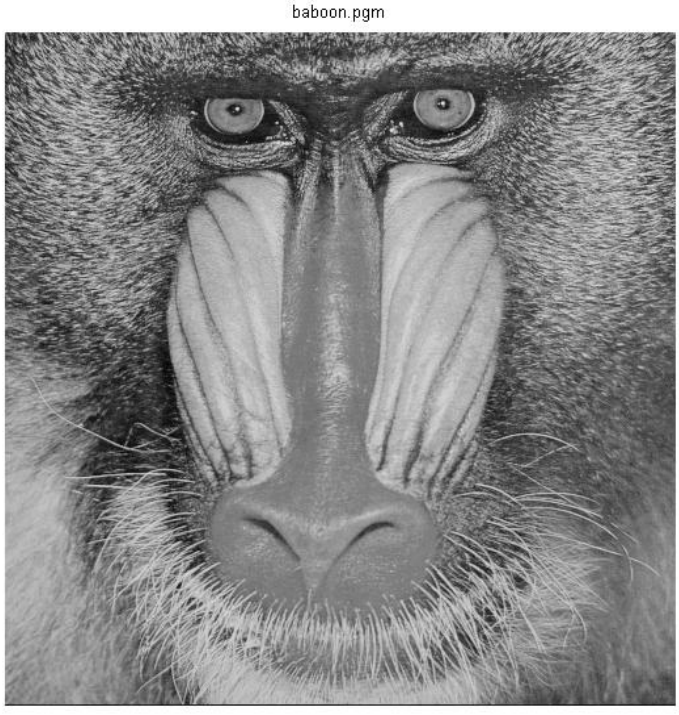


Figure (IV-5) : Image originale ‘Baboon .PGM’.

Les expériences sont effectuées avec les deux types de filtres d'ondelettes du codec JPEG2000, le filtre 5-3 réversible (JPEG 2000 R) et le filtre 9-7 irréversible (JPEG2000 NR). Le tableau (IV-1) présente les résultats obtenus en terme de PSNR(dB) sur les quatre images de tests : "Lena" ,"Boat" , "Barbara", " Baboon " avec différents taux de bits (Tb) en bits par pixel (bpp) : 0.125 bpp, 0.185 bpp, 0.25 bpp, 0.35 bpp, 0.5 bpp, 1.0 bpp .

Les figures ((IV-6) à (IV-9)) : présentent les résultats des PSNRs(dB) obtenus par les deux codecs (JPEG et JPEG2000(R et NR)) en forme de bâtonnets en fonction des différents Tb (bpp) pour chaque image de test .

Les figures ((IV-10) à (IV-12)): présentent le résultat du PSNR (dB) obtenu en fonction des Tb(bpp) pour les quatre images compressées par chaque codec (JPEG et JPEG2000(R et NR)).

Les figures ((IV-13) à (IV-26)) : présentent quelques images reconstruites à différents taux de bits par le codec JPEG et JPEG2000NR.

Nous remarquons que, le résultat du PSNR (dB) obtenu sur les différentes images varie pour chaque taux de bits (figures ((IV-10) à (IV-12)) et cela est due au contenu des images qui différent (riche en détails ou en basse fréquences où elles constituent un mélange).

D'après Les figures ((IV-6) à (IV-9) et (IV-13) à (IV-26)) il apparaît clairement que la norme JPEG 2000 est la plus performante en terme de PSNR et qualité d'image visuelle reconstruite, surtout lorsqu'on l'utilise avec le filtre irréversible qui permet de compresser et décompresser l'image avec perte. La qualité visuelle obtenue avec JPEG 2000 surpasse largement la norme JPEG pour les faibles valeurs du taux de bits (inférieur ou égal à 0.25 bpp). En dessous du Tb = 0.125 bpp l'image JPEG n'est pas reconnaissable alors que l'image JPEG2000 est reconnaissable (figures ((IV-13) et (IV-14))). Pour ces faibles et très faibles valeurs du taux de bits, JPEG offre une très mauvaise qualité visuelle (les limites des blocs sont visibles). Ces derniers apparaissent d'autant plus fréquemment lorsque on traite une image présentent de nombreux détails très fins ou des contours. La qualité visuelle des images JPEG génère des défauts de blocs (effets de blocs) alors que JPEG2000 a tendance à filtrer les images par un filtre passe bas et à supprimer les détails fins ce qui résulte comme conséquence la présence d'un léger floue dans l'image qui est acceptable si on le comparant à l'effet de bloc généré par JPEG.

A des valeurs du taux de bits moins faibles la différence de plus en plus est moins importante et ne serait de toute façon pas visible sur la version imprimée. Les résultats peuvent varier selon le contenu de l'image.

Pour les valeurs du taux de bits supérieur à 0.5 bpp, la qualité visuelle des images reconstruites par JPEG et JPEG2000 est presque identique. A l'oeil nu c'est très difficile de

remarquer la différence malgré que le PSNR donné par JPEG2000 reste meilleur que celui donné par JPEG.

		JPEG		JPEG2000R		JPEG2000NR	
Image	**Taux de bits (Tb) en (bpp)**	**PSNR en (dB)**	**Quotient**	**PSNR en (dB)**	**Quotient**	**PSNR en (dB)**	**Quotient**
Lena	0.125	21.92	64	30.16	64	30.94	64
	0.185	28.23	43	31.90	43	32.61	43
	0.25	30.77	32	33.20	32	34.12	32
	0.35	32.78	23	34.74	23	35.61	23
	0.5	34.75	16	36.36	16	37.30	16
	1	38.25	8	39.31	8	40.40	8
Boat	0.125	21.45	64	27.73	64	28.03	64
	0.185	26.02	43	29.25	43	29.63	43
	0.25	27.94	32	30.50	32	30.99	32
	0.35	30.17	23	32.09	23	32.68	23
	0.5	32.33	16	33.95	16	34.72	16
	1	37.10	8	38.52	8	39.28	8
Barbara	0.125	20.49	64	25.10	64	25.74	64
	0.185	23.59	43	26.55	43	27.18	43
	0.25	24.84	32	27.82	32	28.82	32
	0.35	26.27	23	29.37	23	30.61	23
	0.5	28.34	16	31.48	16	32.85	16
	1	34.13	8	36.58	8	38.05	8

Tableau (IV-1): Résultats des PSNRs obtenus par les deux codecs pour différents Tb (bpp) sur les images de tests.

		JPEG		JPEG2000R		JPEG2000NR	
Image	**Taux de bits (Tb) en (bpp)**	**PSNR en (dB)**	**Quotient**	**PSNR en (dB)**	**Quotient**	**PSNR en (dB)**	**Quotient**
Baboon	0.125	19.09	64	21.47	64	21.83	64
	0.185	20.45	43	22.22	43	22.62	43
	0.25	21.62	32	22.96	32	23.33	32
	0.35	22.94	23	23.96	23	24.30	23
	0.5	23.79	16	25.27	16	25.76	16
	1	26.66	8	28.82	8	29.30	8

Tableau (IV-1): Résultats des PSNRs obtenus par les deux codecs pour différents Tb (bpp) sur les images de tests suite

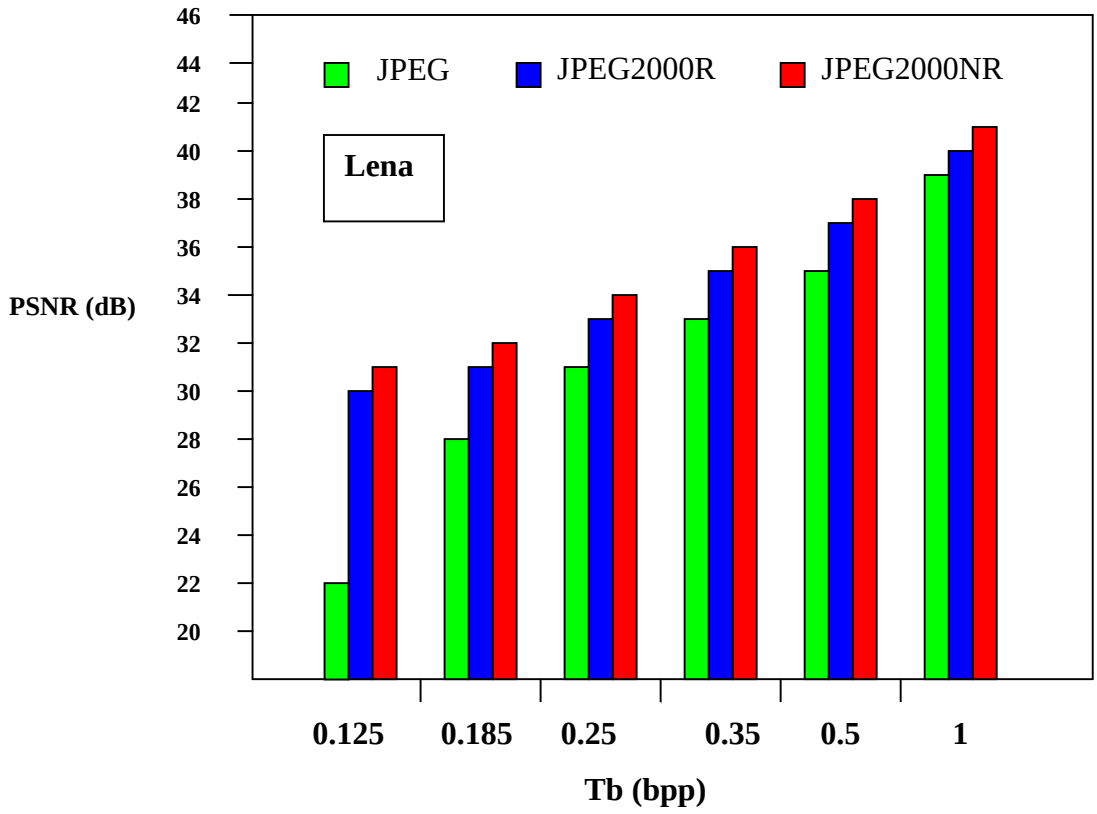


Figure (IV-6) : PSNR (dB) en fonction des différents Tb (bpp) pour l'image Lena.

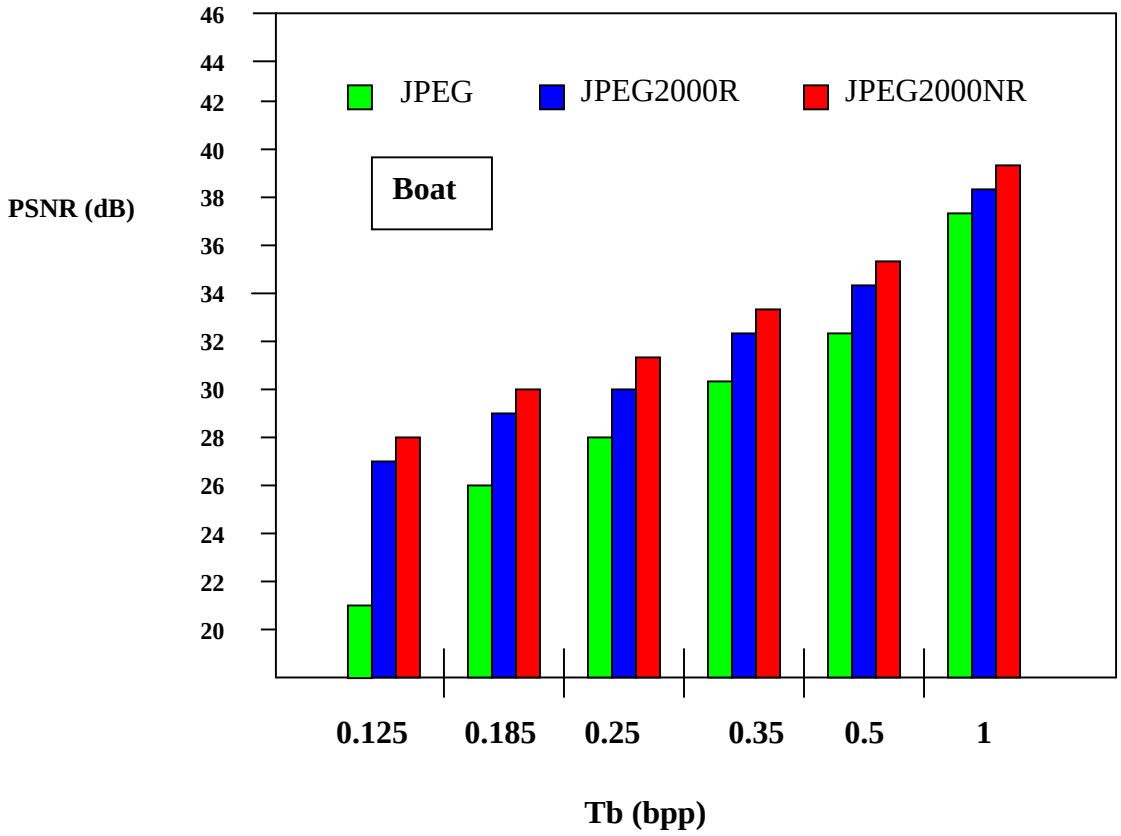


Figure (IV-7) : PSNR (dB) en fonction des différents Tb (bpp) pour l'image Boat.

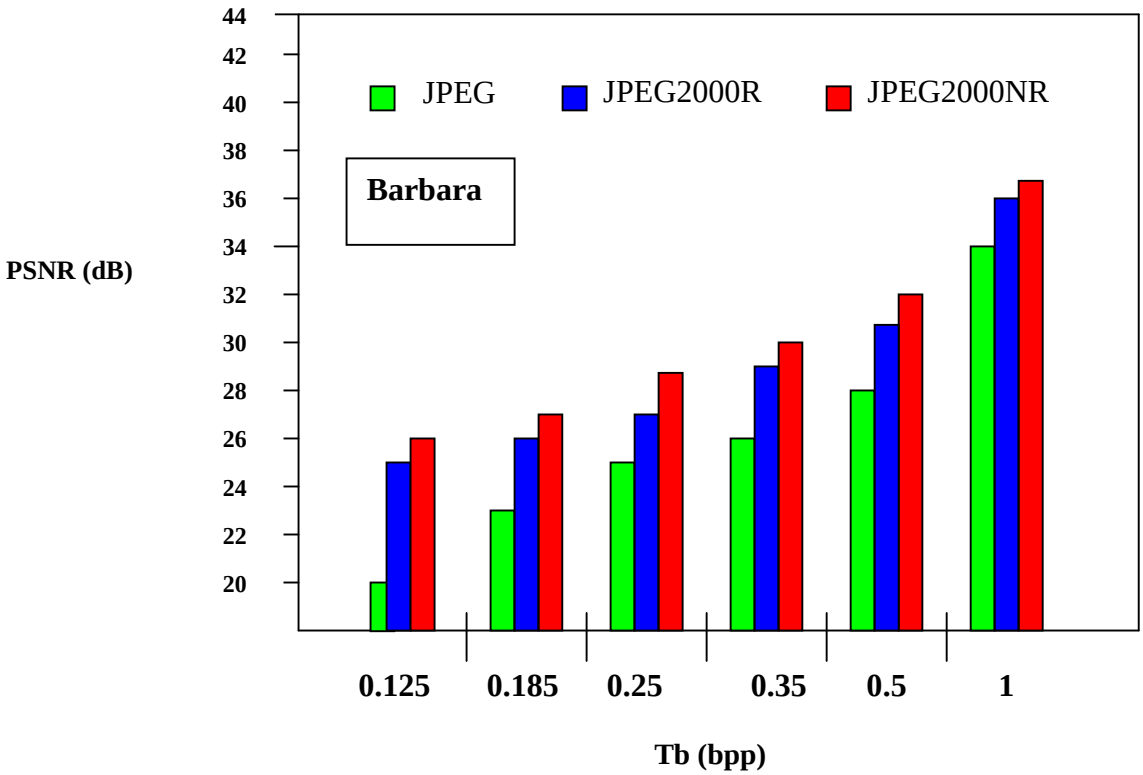


Figure (IV-8) : PSNR (dB) en fonction des différents Tb (bpp) pour l'image Barbara.

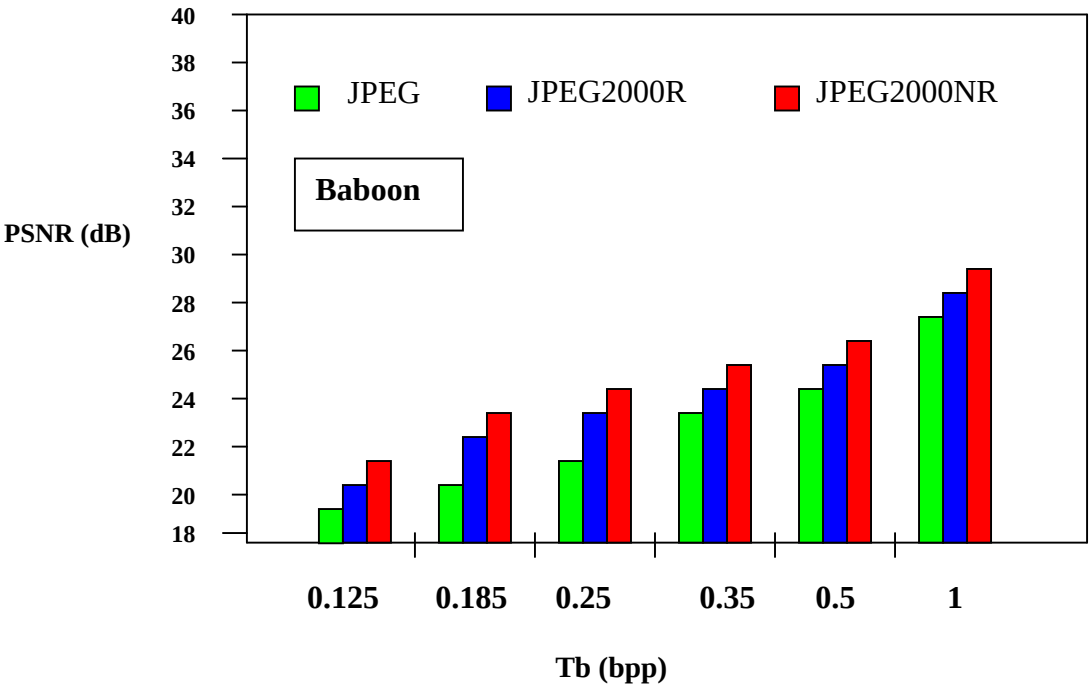


Figure (IV-9) : PSNR (dB) en fonction des différents Tb (bpp) pour l'image Baboon.

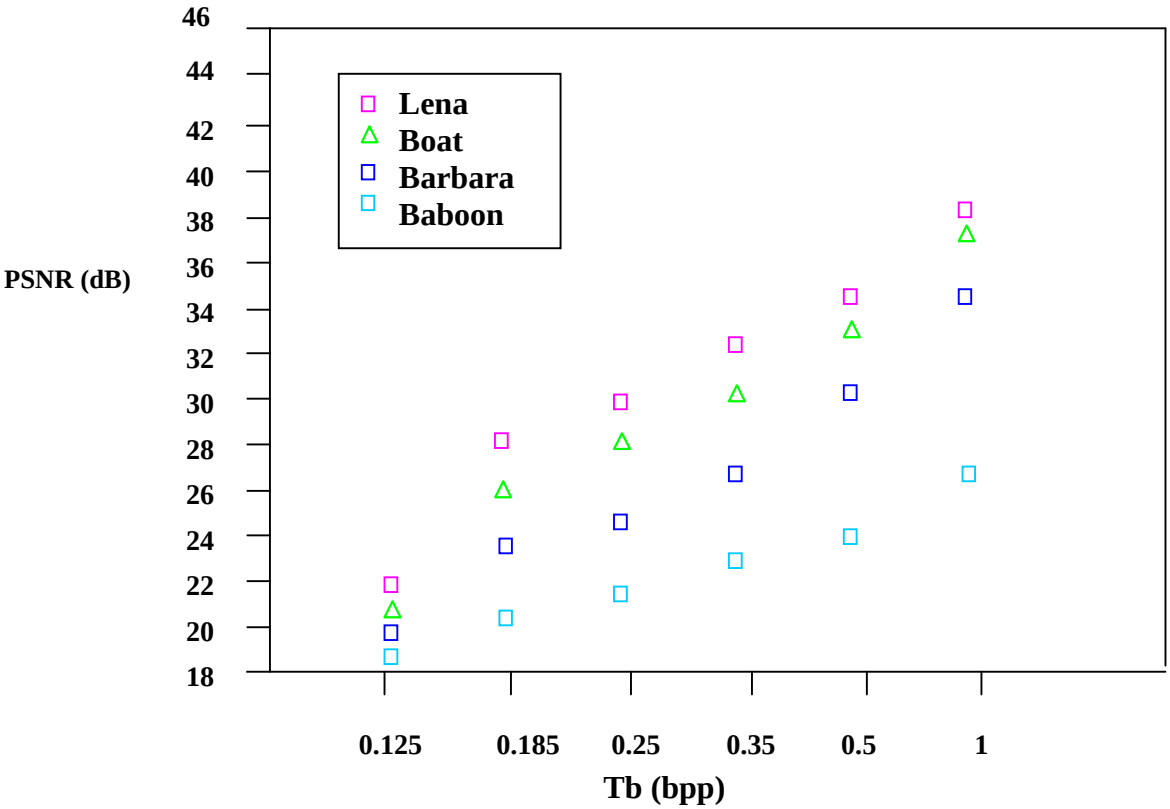


Figure (IV-10) : PSNR (dB) en fonction du Tb (bpp) pour les différentes images obtenu par JPEG.

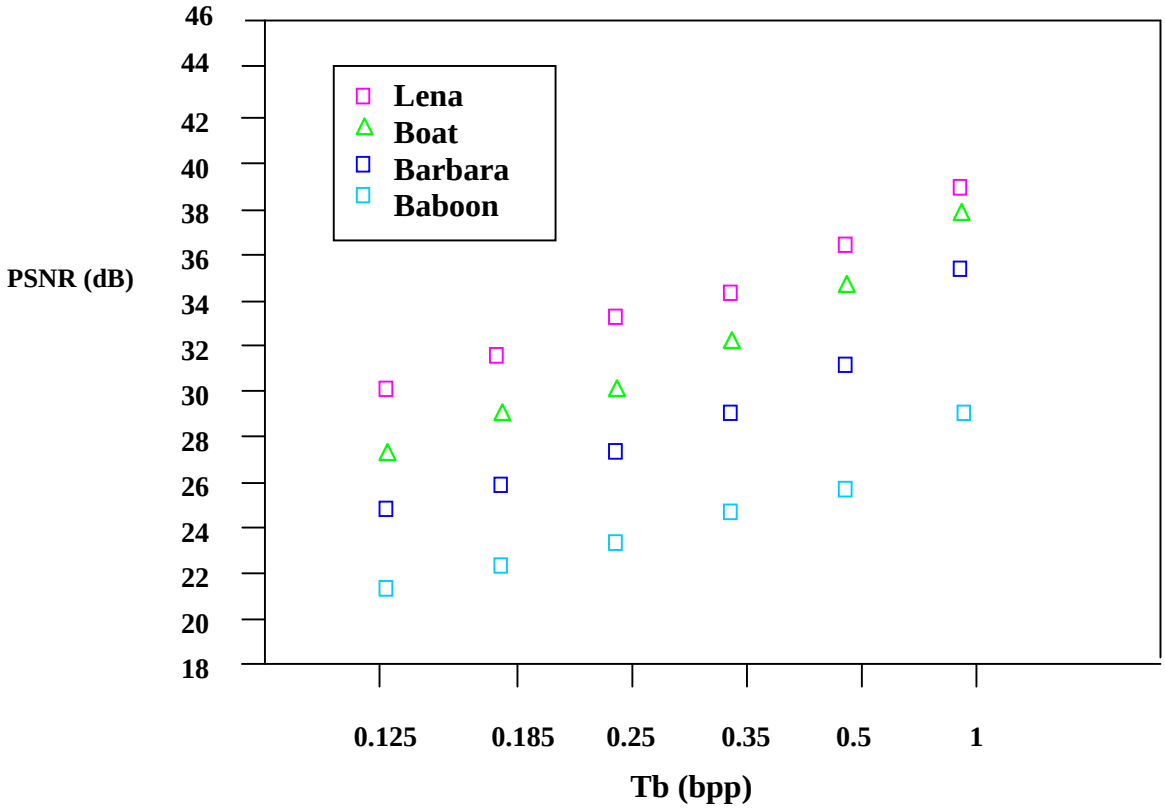


Figure (IV-11) : PSNR (dB) en fonction du Tb (bpp) pour les différentes images obtenu par JPEG2000R.

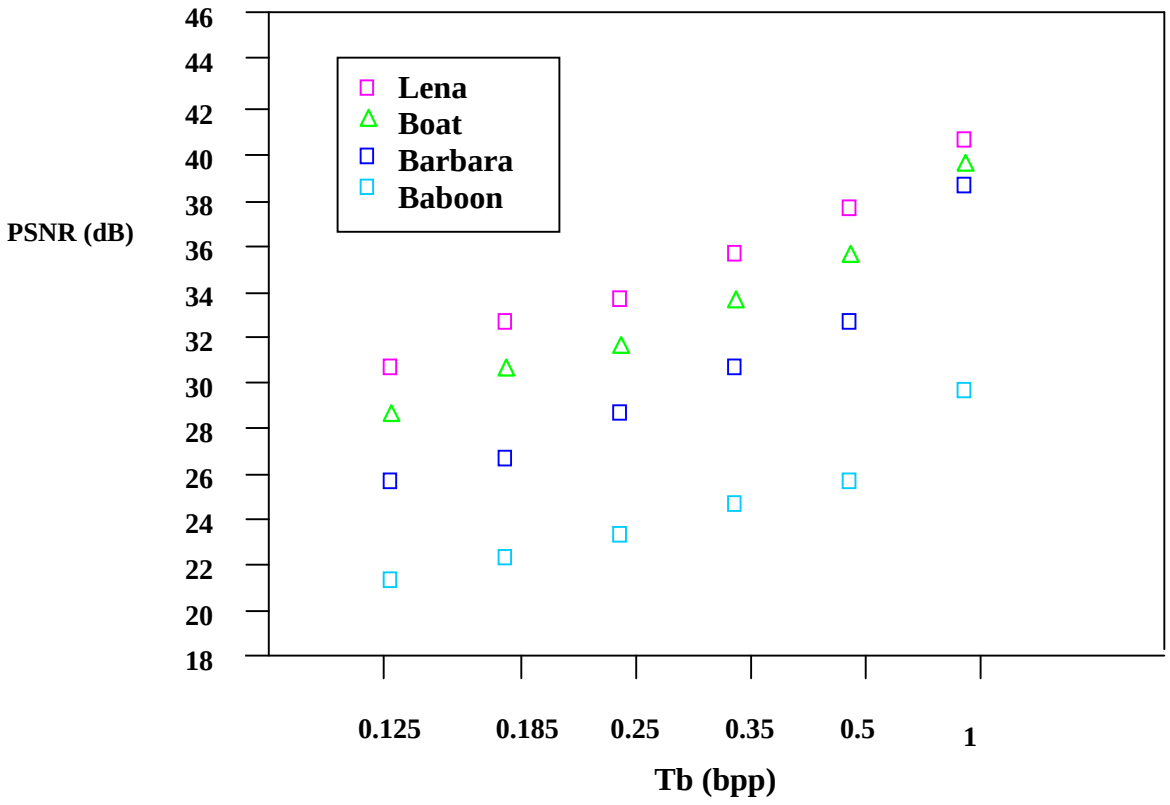


Figure (IV-12) : PSNR (dB) en fonction du Tb (bpp) pour les différentes images obtenu par JPEG2000NR.

PSNR= 27.92 db, Tb= 0.0625 bpp,
Quotient = 128. JPEG2000NR.

Figure (IV-13) : Image 'Lena 'reconstruite après une compression à 0.0625 bpp par le codeur JPEG2000NR.

PSNR= 18 db, Tb= 0.11 bpp,
Quotient = 71. JPEG.

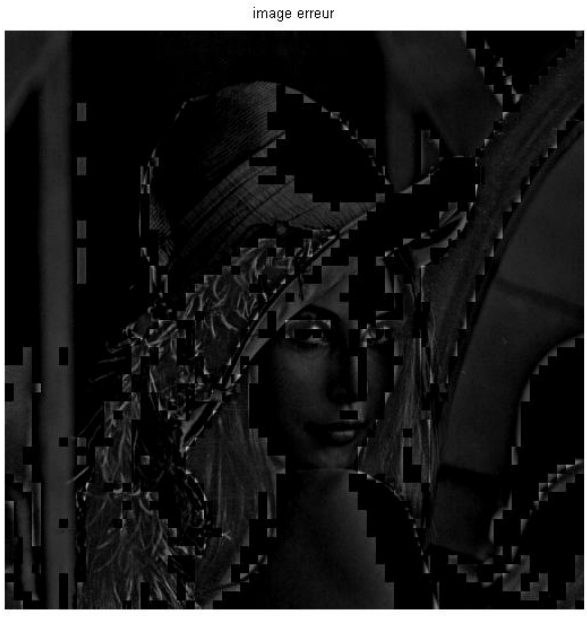
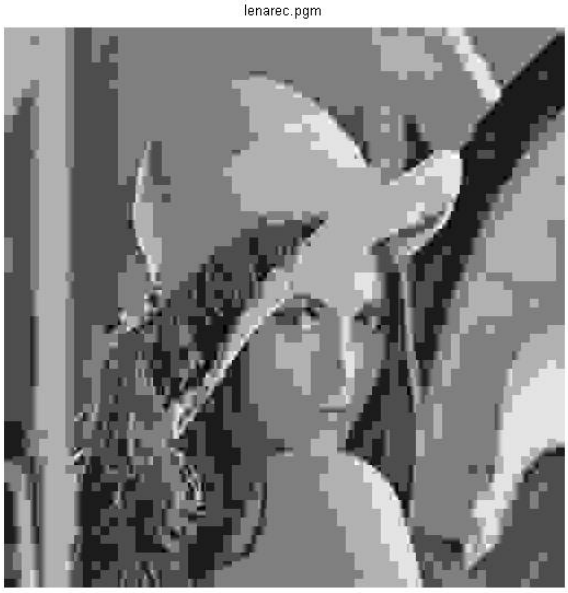


Image erreur = Différence entre image d'origine et son image reconstruite

Figure (IV-14) : Image 'Lena 'reconstruite après une compression à 0.11 bpp par le codeur JPEG et son image d'erreur associée.



PSNR=21.92 db, Tb=0.125 bpp,
Quotient = 64. JPEG.

PSNR = 30.94 db, Tb= 0.125 bpp,
Quotient = 64. JPEG2000NR.

Figure (IV-15) : Image 'Lena 'reconstruite après une compression à 0.125 bpp à gauche par le codeur JPEG et à droite par le codeur JPEG2000NR.

PSNR=28.23 db, Tb= 0.19 bpp,
Quotient = 42. JPEG.

PSNR=32.61 db, Tb= 0.185 bpp,
Quotient = 43. JPEG2000NR.

Figure (IV-16) : Image 'Lena 'reconstruite après une compression à 0.19 bpp à gauche par le codeur JPEG et à 0.185 bpp à droite par le codeur JPEG2000NR.

PSNR=30.77 db, Tb= 0.25 bpp,
Quotient = 32. JPEG.

PSNR= 34.12 db, Tb= 0.25 bpp,
Quotient = 32. JPEG2000NR.

Figure (IV-17) : Image 'Lena 'reconstruite après une compression à 0.25 bpp à gauche par le codeur JPEG et à droite par le codeur JPEG2000NR.

PSNR = 21.45 db, Tb= 0.125 bpp,
Quotient = 64. JPEG.

PSNR = 28.03 db, Tb= 0.125 bpp,
Quotient = 64. JPEG2000NR.

Figure (IV-18) : Image 'Boat 'reconstruite après une compression à 0.125 bpp à gauche par le codeur JPEG et à droite par le codeur JPEG2000NR.

boatrec.pgm

PSNR= 26.02 db, Tb= 0.185 bpp, Quotient = 42. JPEG.

boatdec.pgm

PSNR= 29.63 db, Tb= 0.185 bpp, Quotient = 42. JPEG2000NR.

Figure (IV-19) : Image 'Boat 'reconstruite après une compression à 0.185 bpp à gauche par le codeur JPEG et à droite par le codeur JPEG2000NR.

boatrec.pgm

PSNR=27.94 db, Tb= 0.25 bpp, Quotient = 32. JPEG.

boatdec.pgm

PSNR=30.99 db, Tb= 0.25 bpp, Quotient = 32. JPEG2000NR.

Figure (IV-20) : Image 'Boat 'reconstruite après une compression à 0.25 bpp à gauche par le codeur JPEG et à droite par le codeur JPEG2000NR.

PSNR = 20.49 db, Tb= 0.125 bpp, Quotient = 64. JPEG.

PSNR = 25.74 db, Tb= 0.125 bpp, Quotient = 64. JPEG2000NR.

Figure (IV-21) : Image 'Barbara 'reconstruite après une compression à 0.125 bpp à gauche par le codeur JPEG et à droite par le codeur JPEG2000NR.

PSNR= 23.59 db, Tb= 0.185 bpp, Quotient = 42. JPEG.

PSNR= 27.18 db, Tb= 0.185 bpp, Quotient = 42. JPEG2000NR.

Figure (IV-22) : Image 'Barbara 'reconstruite après une compression à 0.185 bpp à gauche par le codeur JPEG et à droite par le codeur JPEG2000NR.

barbararec.pgm

PSNR=24.84 db, Tb= 0.25 bpp, Quotient = 32. JPEG.

barbaradec.pgm

PSNR= 28.82 db, Tb= 0.25 bpp, Quotient = 32. JPEG2000NR.

Figure (IV-23) : Image 'Barbara 'reconstruite après une compression à 0.25 bpp à gauche par le codeur JPEG et à droite par le codeur JPEG2000NR.

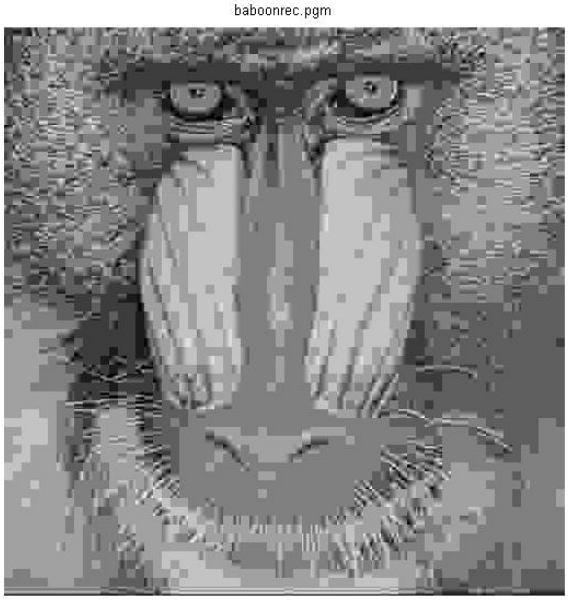
baboonrec.pgm

PSNR = 19.09 db, Tb= 0.125 bpp, Quotient = 64. JPEG.

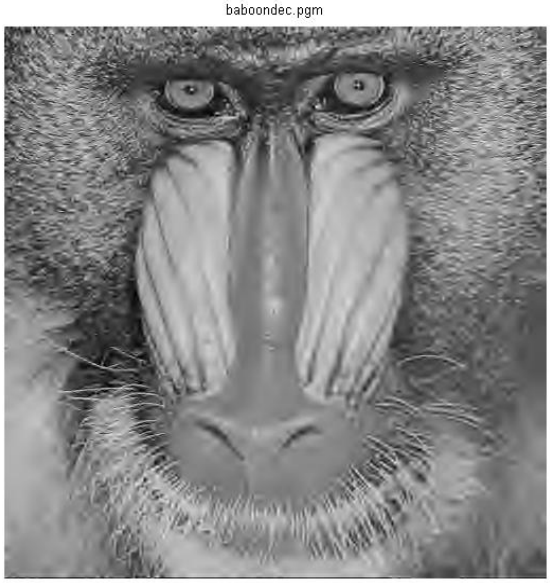
baboondec.pgm

PSNR = 21.83 db, Tb= 0.125 bpp, Quotient = 64. JPEG2000NR.

Figure (IV-24) : Image 'Baboon 'reconstruite après une compression à 0.125 bpp à gauche par le codeur JPEG et à droite par le codeur JPEG2000NR.

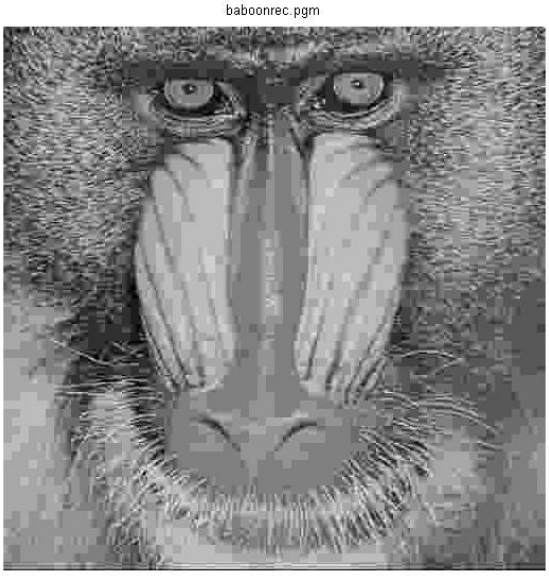


PSNR= 20.45 db, Tb= 0.185 bpp,
Quotient = 42. JPEG.

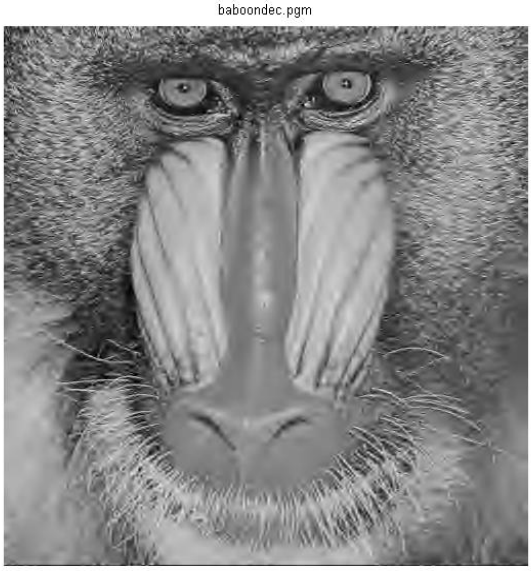


PSNR= 22.62 db, Tb= 0.185 bpp,
Quotient = 42. JPEG2000NR.

Figure (IV-25) : Image 'Baboon 'reconstruite après une compression à 0.185 bpp à gauche par le codeur JPEG et à droite par le codeur JPEG2000NR.



PSNR= 21.62 db, Tb= 0.25 bpp,
Quotient = 32. JPEG.



PSNR=23.33 db, Tb= 0.25 bpp,
Quotient = 32. JPEG2000NR.

Figure (IV-26) : Image 'Baboon 'reconstruite après une compression à 0.25 bpp à gauche par le codeur JPEG et à droite par le codeur JPEG2000NR.

IV-3-2. Test avec différents niveaux de décomposition en ondelette et code-blocs

Afin d'aboutir aux résultats qui permettent un meilleur compromis PSNR / taux de bits (Tb), nous avons proposé de varier la taille des code-blocs utilisés pour la compression des images et le niveau de résolution de la décomposition en ondelettes. Nous avons sélectionné le type de filtre 9/7 d'ondelette irréversible et le test est effectué sur l'image **' Lena .PGM '** pour un Tb de (**0.5 bpp)**. Les résultats sont illustrés dans le tableau (IV-2)**.**

Image	**Taille des blocs**	**2éme résolution**	**3éme résolution**	**4éme résolution**	**5éme résolution**	**6éme résolution**	**7éme résolution**	**8éme résolution**	**9éme résolution**	**10éme résolution**
		PSNR en (dB)	**PSNR en (dB)**	**PSNR en (dB)**	**PSNR en (dB)**	**PSNR en (dB)**	**PSNR en (dB)**	**PSNR en (dB)**	**PSNR en (dB)**	**PSNR en (dB)**
Lena (0.5 bpp)	**4*4**	34.21	35.26	35.38	35.40	35.39	35.38	35.37	35.37	35.37
	8*8	35.77	36.37	36.43	36.44	36.43	36.42	36.42	36.41	36.41
	16*16	36.44	36.92	36.97	36.98	36.96	36.95	36.94	36.93	36.93
	32*32	36.70	37.17	37.22	37.22	37.20	37.19	37.19	37.18	37.18
	64*64	36.76	37.27	37.30	37.30	37.29	37.28	37.27	37.27	37.27

Tableau (IV-2): Résultats du PSNR avec variation de la taille des code-blocs et avec différents niveaux de décomposition en ondelettes.

Après l'analyse des résultats du tableau (IV-2) nous constatons que le PSNR augmente avec l'augmentation de la taille des code-blocs et cela pour tous les niveaux de résolution de la décomposition en ondelettes. Les résultats obtenus pour la taille des code-blocs de (64 * 64) sont appréciables en donnant un bon rapport taux de bits qualité visuelle (PSNR).

IV-3-3. Test sur une image médicale

Les résultats des tests de compression sur l'image médicale Thorax de résolution de 288 * 376 pixels codée sur 256 niveaux de gris [48] (figure (IV-27)) avec différents taux de bits (Tb) sont rapportés dans le tableau (IV-3) :

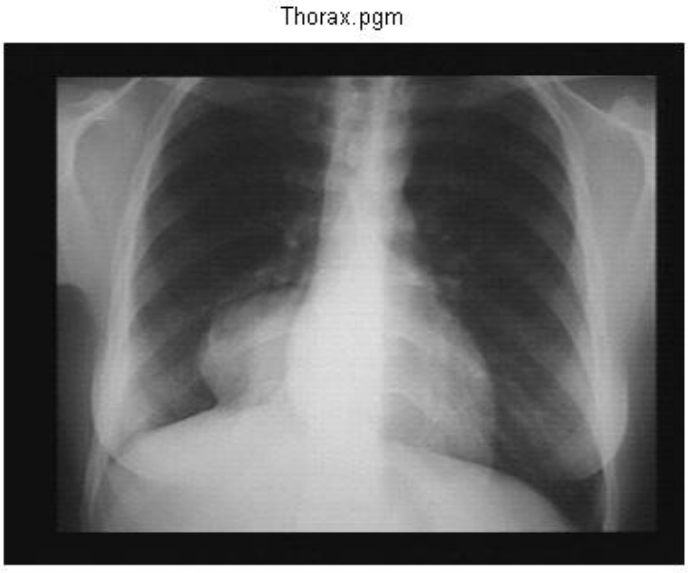


Figure (IV- 27) : Image originale 'Thorax. PGM'.

		JPEG		JPEG2000NR	
Image	**Taux de bits (Tb) en (bpp)**	**PSNR en (dB)**	**Quotient**	**PSNR en (dB)**	**Quotient**
Thorax	0.125	19.08	64	37.77	64
	0.185	31.80	43	39.31	43
	0.25	35.75	32	40.60	32
	0.35	38.21	23	41.87	23
	0.5	41.28	16	43.22	16
	1	46.70	8	47.14	8

Tableau (IV-3): Résultats du PSNR pour différents Tb (bpp) avec les deux codecs sur l'image Thorax.

La figure (IV-28) présente les résultats des PSNRs(dB) en forme de bâtonnets obtenus sur l' image de test Thorax par les deux codecs (JPEG et JPEG2000NR) pour les différents Tb (bpp) utilisés.

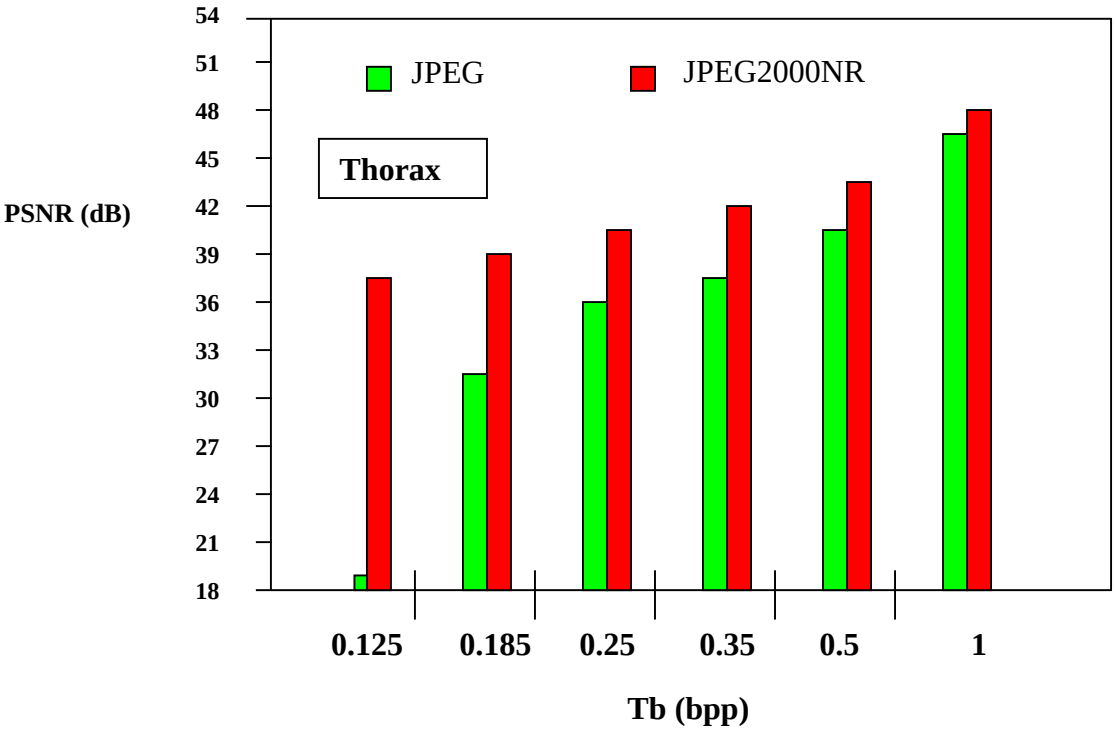
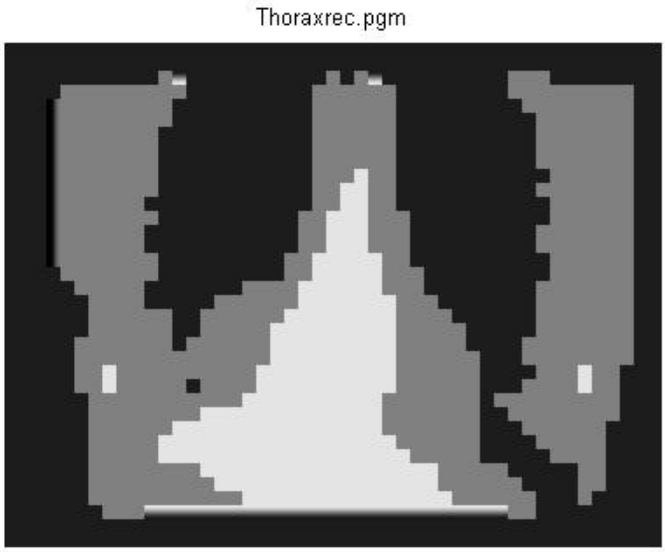


Figure (IV-28) : PSNR (dB) en fonction des différents Tb (bpp) pour l'image Thorax.

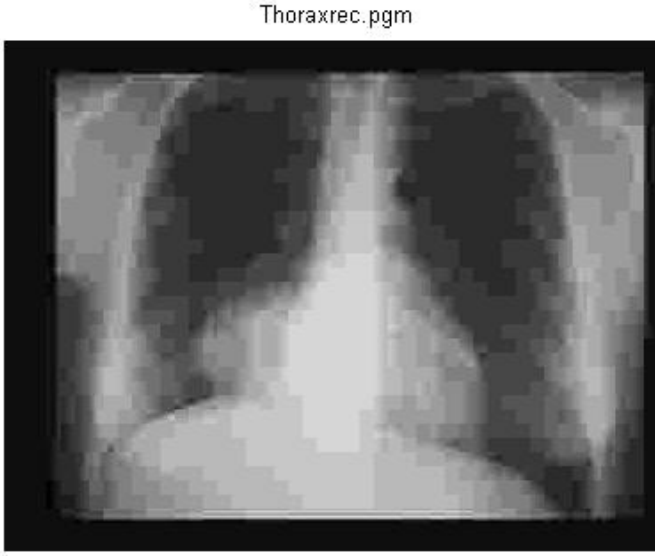
D'après la figure (IV-28) nous remarquons que les PSNRs(dB) obtenus par JPEG2000NR sont meilleurs que ceux obtenus par JPEG et cela pour tout les taux de bits utilisés. Afin d'évaluer la qualité perceptuelle des images décompressées, nous avons présentées dans les figures ((IV-29) à (IV-31)) quelques images restituées. Ces figures représentent les images restituées respectivement à des Tb inférieur ou égal à 0.25 bpp. Nous voyons clairement que les images restituées par JPEG sont dégradées, elles sont d'une mauvaise qualité visuelle due à la présence des effets de blocs. Par contre pour les images restituées par JPEG2000NR nous remarquons qu'aucun effet de blocs n'est visible et la qualité des images reconstruites est très bonne et portent des gains d'importantes valeurs de PSNR par rapport à JPEG. Exemple 18.69 dB pour un Tb = 0.125 bpp et 4.85 dB pour un Tb = 0.25 bpp .



PSNR=19.08 db, Tb = 0.125 bpp,
Quotient = 64. JPEG.

PSNR=37.77 db, Tb = 0.125 bpp,
Quotient = 64. JPEG2000NR.

Figure (IV-29) : Image 'Thorax' reconstruite après une compression à 0.125 bpp à gauche par le codeur JPEG et à droite par le codeur JPEG2000NR.

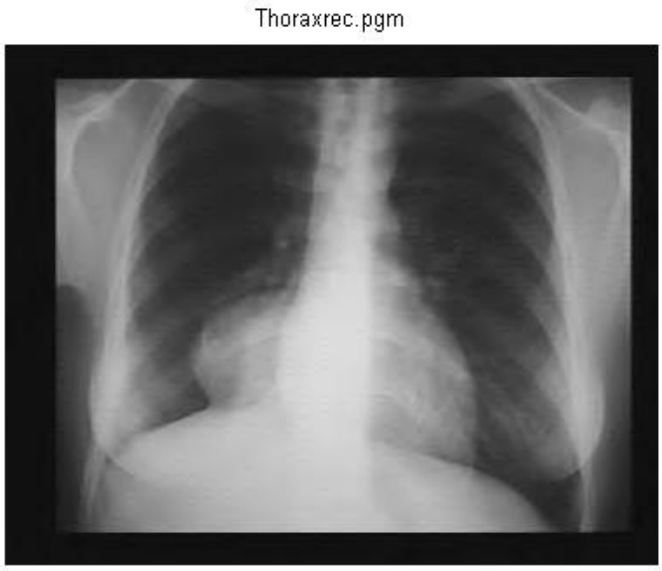


PSNR= 31.80 db, Tb = 0.185 bpp,
Quotient = 43. JPEG.

PSNR= 39.31 db, Tb = 0.185 bpp,
Quotient = 43. JPEG2000NR.

Figure (IV-30) : Image 'Thorax 'reconstruite après une compression à 0.185 bpp à gauche par le codeur JPEG et à droite par le codeur JPEG2000NR.

PSNR=35.75 db, Tb = 0.25 bpp, Quotient = 32. JPEG.



PSNR= 40.60 db, Tb = 0.25 bpp, Quotient = 32. JPEG2000NR.

Figure (IV-31) : Image 'Thorax'reconstruite après une compression à 0.25 bpp à gauche par le codeur JPEG et à droite par le codeur JPEG2000NR.

IV-3-4. Test sur une image couleur

Les résultats des tests de compression sur l'image couleur Lena de résolution 512 * 512 pixels codée sur 24 bits / pixel (figure (IV-32)) avec différents taux de bits (Tb) sont rapportés dans le tableau (IV-4). Nous avons calculé le PSNR entre la composante rouge de l'image d'origine et la composante rouge de l'image décompressée et entre la composante verte de l'image d'origine et la composante verte de l'image décompressée et entre la composante bleu de l'image d'origine et la composante bleu de l'image décompressée et cela après décomposition de l'image Lena couleur (d'origine et décompressée) en trois composantes (Rouge(R), Bleu(B) ,Verte(V)) bien sur.

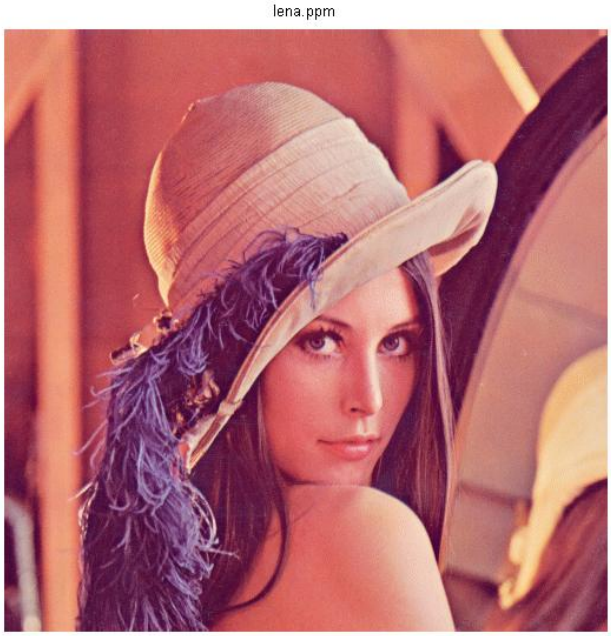


Figure (IV-32) : Image originale 'Lena .PPM'.

			JPEG			JPEG2000NR		
Image	**Taux de bits (Tb) en (bpp)**	**Quotient**	**PSNR en (dB) Pour chaque composante**			**PSNR en (dB) Pour chaque composante**		
			R	**V**	**B**	**R**	**V**	**B**
Lena	0.125	192	11.70	15.95	15.53	27.78	27.93	26.11
	0.185	130	21.27	22.83	19.39	28.86	29.12	26.95
	0.25	96	25.41	26.71	24.00	29.56	30.09	27.43
	0.35	68	27.80	29.11	25.92	30.22	30.70	27.94
	0.5	48	29.22	30.74	27.24	30.66	31.40	28.41
	1	24	30.50	32.88	28.57	31.51	33.01	29.14

Tableau (IV-4): Résultats des PSNRs sur l'image Lena obtenus avec les deux codecs pour différents Tb (bpp).

La figure (IV-33) présente les résultats des PSNRs(dB) en forme de bâtonnets obtenus sur l' image de test Lena couleur par les deux codecs (JPEG et JPEG2000NR) pour les différents Tb (bpp) utilisés.

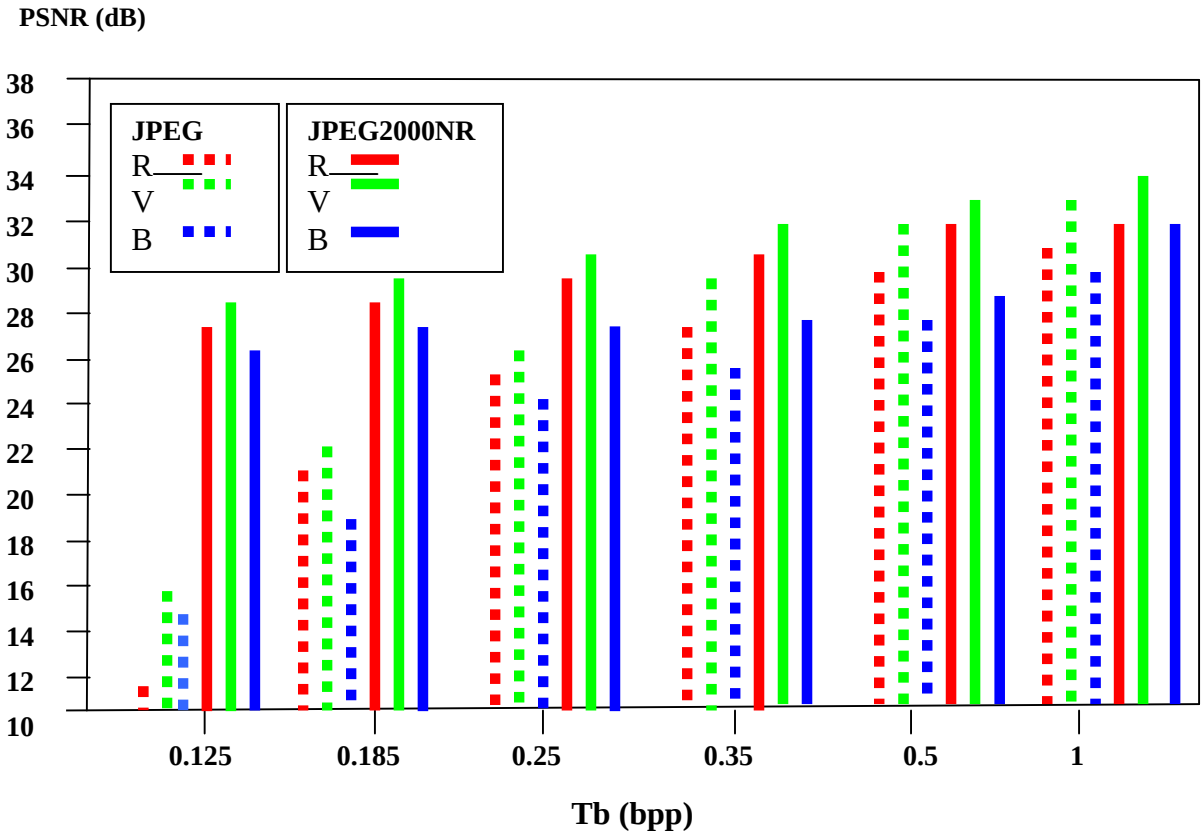


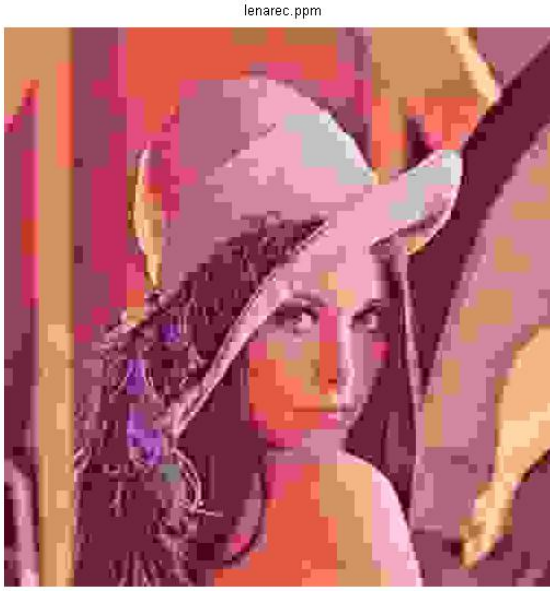
Figure (IV-33) : PSNRs(dB) pour chaque composante (rouge , verte, bleu) en fonction des différents Tb (bpp) obtenus sur l'image Lena .

D'après la figure (IV-33), il apparaît clairement la supériorité de JPEG2000NR sur JPEG en terme de PSNR et cela pour chaque composante (Rouge(R), Bleu (B), Verte (V)) de l'image. En faisant une comparaison visuelle entre les images reconstruites (figures ((IV-34) à (IV- 37)), nous remarquons que les résultats obtenus pour les taux de bits ($\leq$ 0.35bpp) par le codec JPEG2000NR sont meilleurs que ceux obtenus par le codec JPEG qui apparaît détérioré par les matrices des blocs 8x8 pixels. Ces matrices deviennent légèrement visibles et ont tendance à disparaître de plus en plus au fur et à mesure qu'on augmente le taux de bits. Le défaut de la compression JPEG2000NR présente un légère floue dans l'image qui est acceptable si on le compare à l'effet de bloc généré par JPEG. Pour des taux de bits (> 0.5 bpp), JPEG et JPEG2000 sont sensiblement identiques visuellement.

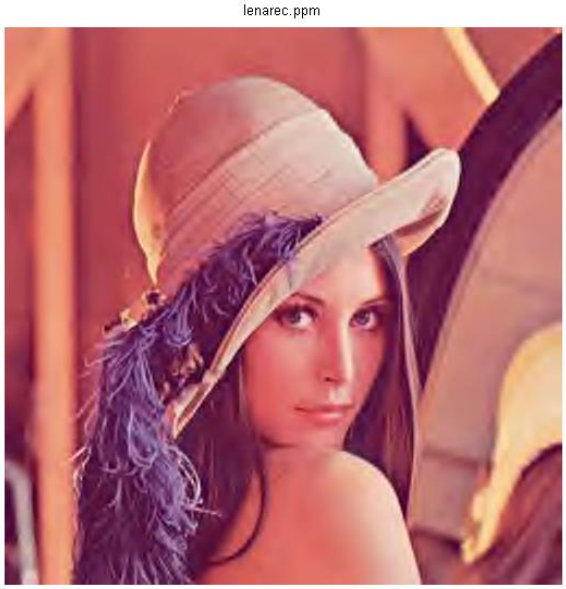
Tb= 0.125 bpp,
Quotient =192. JPEG.

Tb = 0.125 bpp,
Quotient =192. JPEG2000NR.

Figure (IV-34) : Image 'Lena' reconstruite après une compression à 0.125 bpp à gauche par le codeur JPEG et à droite par le codeur JPEG2000NR.

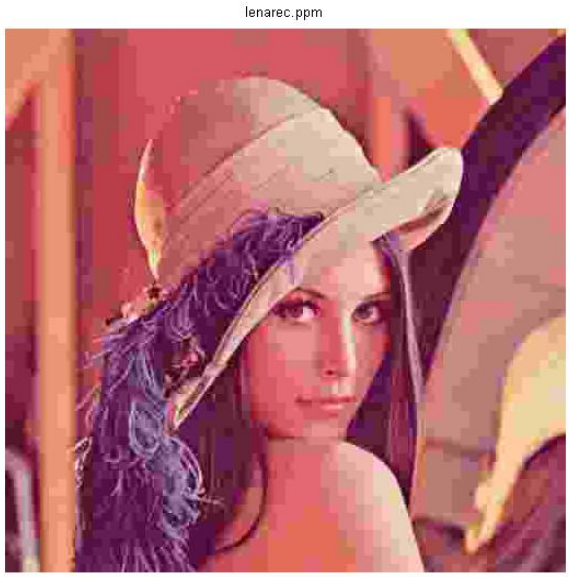


Tb = 0.185 bpp,
Quotient = 130. JPEG.

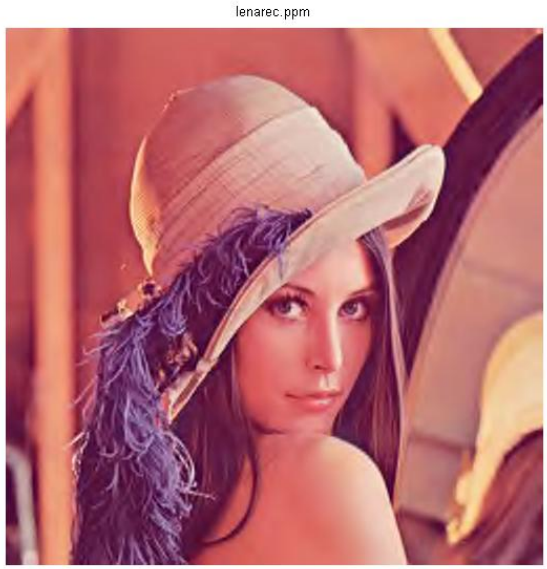


Tb = 0.185 bpp,
Quotient = 130. JPEG2000NR.

Figure (IV-35) : Image 'Lena' reconstruite après une compression à 0.185 bpp à gauche par le codeur JPEG et à droite par le codeur JPEG2000NR.

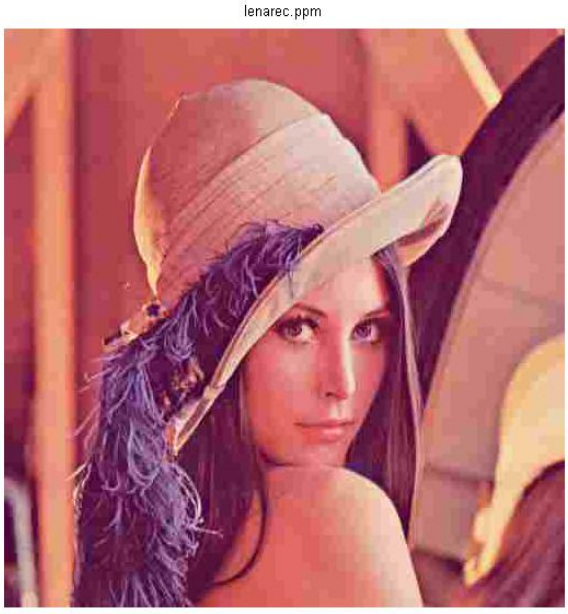


Tb = 0.25 bpp,
Quotient = 96. JPEG.

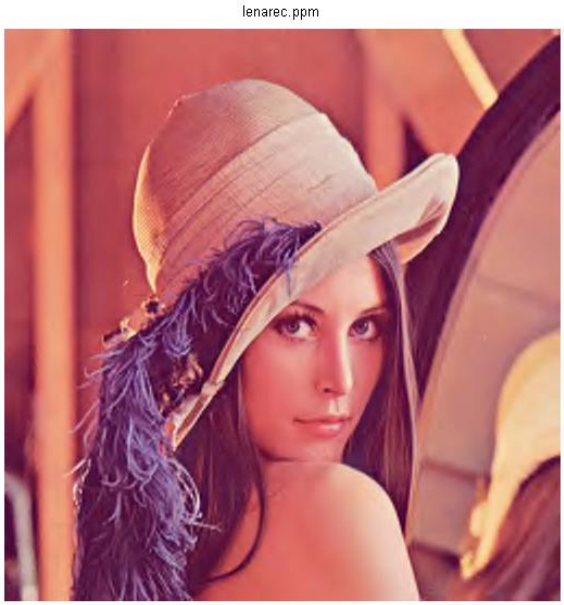


Tb = 0.25 bpp,
Quotient = 96. JPEG2000NR.

Figure (IV-36) : Image 'Lena' reconstruite après une compression à 0.25 bpp à gauche par le codeur JPEG et à droite par le codeur JPEG2000NR.



Tb = 0.35 bpp,
Quotient = 68. JPEG.



Tb = 0.35 bpp,
Quotient = 68. JPEG2000NR.

Figure (IV- 37) : Image 'Lena' reconstruite après une compression à 0.35 bpp à gauche par le codeur JPEG et à droite par le codeur JPEG2000NR.

Remarque

Une image compressée par les ondelettes peut être décompressée de trois manières différentes : sa résolution (taille) est fixe mais sa qualité augmente progressivement(Scalabilité en qualité), sa qualité est fixe mais sa résolution augmente progressivement (Scalabilité en résolution) , sa qualité et sa résolution peuvent augmenter progressivement(Scalabilité en résolution et Qualité).

IV-3-5. Test Scalabilité en qualité sur une image au niveau de gris Lena

La technique JPEG2000 permet de créer un seul fichier qui contient l'image compressée à différents niveaux de qualité (différents taux de bits en même temps) et permet aussi de décompresser l'image à ces différents niveaux de qualité. L'image de test Lena est compressée avec ces trois taux de bits (0.125 bpp**,** 0.25 bpp**,** 0.5 bpp) en même temps et les résultats des images de reconstructions sont donnés par les figures ((IV- 38) à (IV- 40)) :

Figure (IV- 38) : Image ‘ Lena ’ reconstruite à 0.125 bpp par le codeur JPEG2000.

Figure (IV- 39) : Image ‘ Lena ’ reconstruite à 0.25 bpp par le codeur JPEG2000.

Figure (IV- 40) : Image ‘ Lena ’reconstruite à 0.5 bpp par le codeur JPEG2000.

V-3-6.Test Scalabilité en résolution sur une image au niveau de gris Lena

JPEG2000 nous permet aussi de décompresser l’image compressée à différents niveaux de résolution. Le test Scalabilité en résolution effectué sur l’image Lena compressée à 0.25 bpp est illustré par les figures ((IV- 41) à (IV- 44)) :

Résolution 32 x 32 pixels
(a)

Résolution 64 x 64 pixels
(b)

Figure (IV- 41) : Image ‘ Lena ’ reconstruite divisée horizontalement et verticalement par 16 (a) et par 8 (b) par rapport à l’image d’origine par le codeur JPEG2000.

Résolution 128 x 128 pixels

Figure (IV- 42) : Image ‘ Lena ’ reconstruite divisée horizontalement et verticalement par 4 par rapport à l’image d’origine par le codeur JPEG2000.

Résolution 256 x 256 pixels

Figure (IV-43) : Image ‘ Lena ’ reconstruite divisée horizontalement et verticalement par 2 par rapport à l’image d’origine par le codeur JPEG2000.

Résolution 512 x 512 pixels

Figure (IV- 44) : Image ‘ Lena ’ reconstruite par sa taille d’origine par le codeur JPEG2000.

IV-3-7.Test Scalabilité en résolution et Qualité sur une image au niveau de gris Lena

En plus de la décompression de l'image avec différents niveaux de résolution et différents niveaux de qualité, la technique permet aussi de décompresser l’image à différents niveaux de résolution et qualité en même temps. Le test effectué sur l’image Lena compressée à ces taux de bits (0.125 bpp**,** 0.25 bpp**,** 0.5 bpp) en même temps est illustré par la figure (IV- 45) et la figure (IV- 46)) :

Résolution 256 x 256 pixels
(a)

Résolution 256 x 256 pixels
(b)

Résolution 256 x 256 pixels
(c)

Résolution 256 x 256 pixels
(d)

Figure (IV- 45) : Image ‘ Lena ’ reconstruite divisée horizontalement et verticalement par 2 par rapport à l’image d’origine et à différents niveaux de qualité : (a) 0.125 bpp, (b) 0.20 bpp et (c) 0.25 bpp et (d) 0.5 par le codeur JPEG2000.

Résolution 128 x 128 pixels
(d)

Résolution 128 x 128 pixels
(e)

Résolution 128 x 128 pixels
(f)

Figure (IV- 46) : Image ‘ Lena ’reconstruite divisée horizontalement et verticalement par 4 par rapport à l’image d’origine et à différents niveaux de qualité : (d) 0.125 bpp, (e) 0.20 bpp et (f) 0.25 bpp par le codeur JPEG2000.

IV-3- 8. Test Scalabilité en qualité sur une image couleur poivron

Le test de Scalabilité en qualité sur l’image couleur poivron de résolution 512 * 512 pixels codée sur 24 bits / pixel compressée à ces trois taux de bits (0.1 bpp**,** 0.2 bpp**,** 0.5 bpp) en même temps est donné par les figures ((IV- 47) à (IV- 49)) :

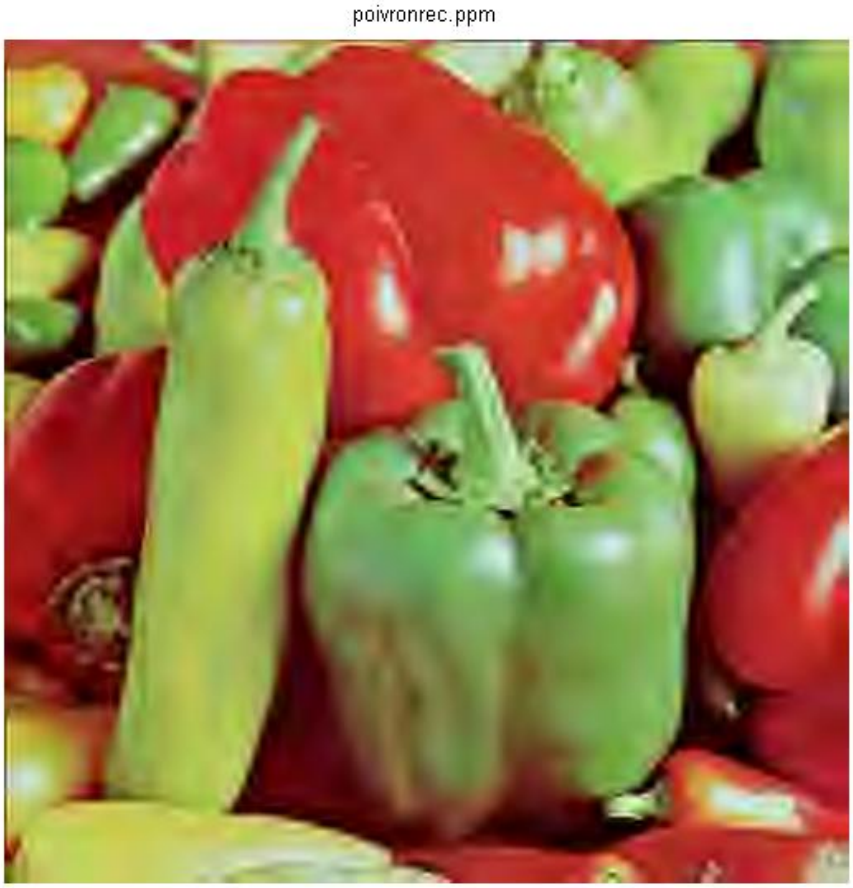


Figure (IV-47) : Image ‘poivron ’reconstruite à 0.1 bpp par le codeur JPEG2000.

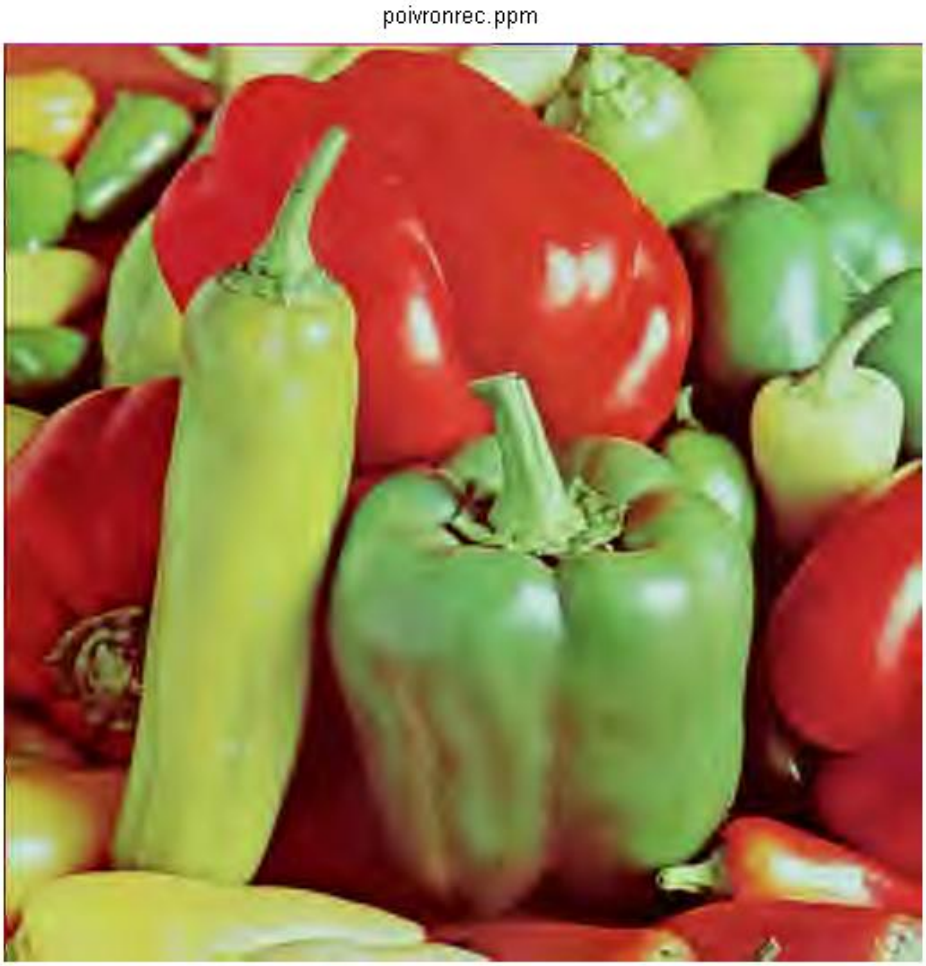


Figure (IV-48) : Image ‘poivron ’reconstruite à 0.2 bpp par le codeur JPEG2000.

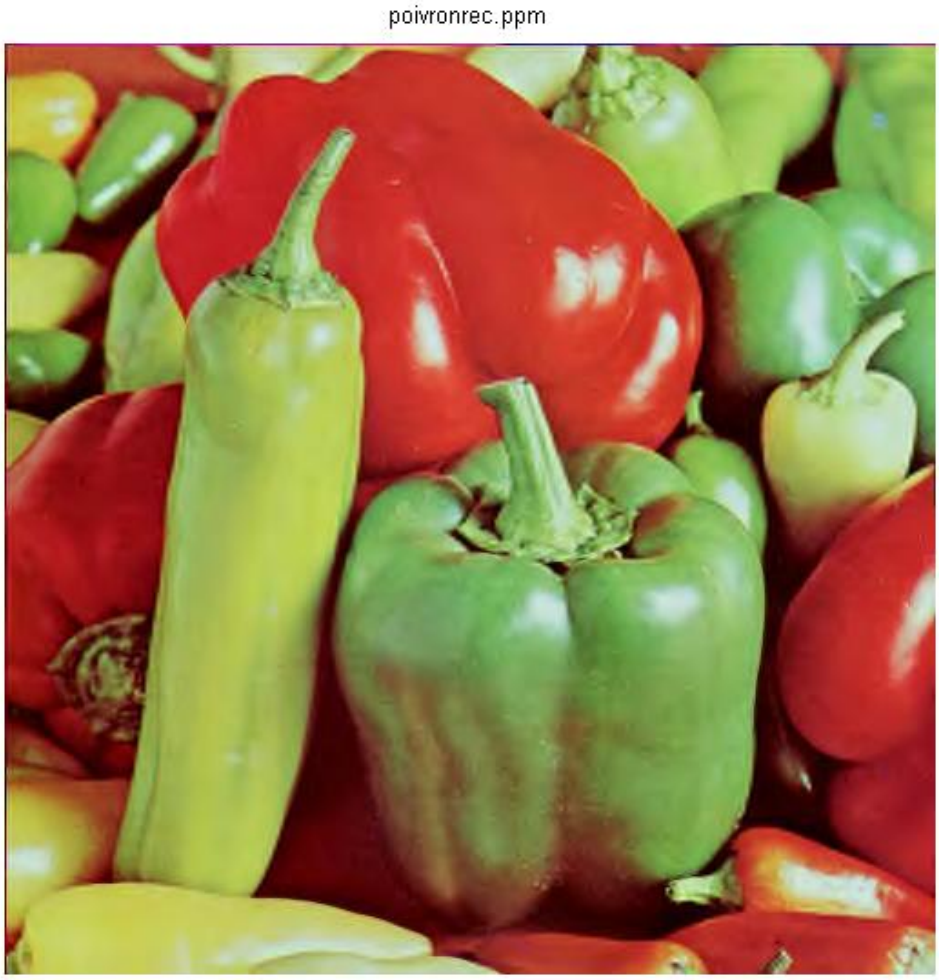


Figure (IV-49) : Image ‘poivron ’reconstruite à 0.5 bpp par le codeur JPEG2000.

IV-3-9. Test Scalabilité en résolution sur une image couleur poivron

Le test de Scalabilité en résolution sur l'image couleur poivron de résolution 512 * 512 pixels codée sur 24 bits / pixel compressée à 0.2 bpp est montré par les figures ((IV- 50) à (IV- 54)) :

Résolution 32 x 32 pixels

Figure (IV-50) : Image 'poivron 'reconstruite divisée horizontalement et verticalement par 16 par rapport à l'image d'origine par le codeur JPEG2000.

Résolution 64 x 64 pixels

Figure (IV-51) : Image 'poivron 'reconstruite divisée horizontalement et verticalement par 8 par rapport à l'image d'origine par le codeur JPEG2000.

Résolution 128 x 128 pixels

Figure (IV-52) : Image 'poivron 'reconstruite divisée horizontalement et verticalement par 4 par rapport à l'image d'origine par le codeur JPEG2000.

Résolution 256 x 256 pixels

Figure (IV-53) : Image ‘poivron ’reconstruite divisée horizontalement et verticalement par 2 par rapport à l’image d’origine par le codeur JPEG2000.

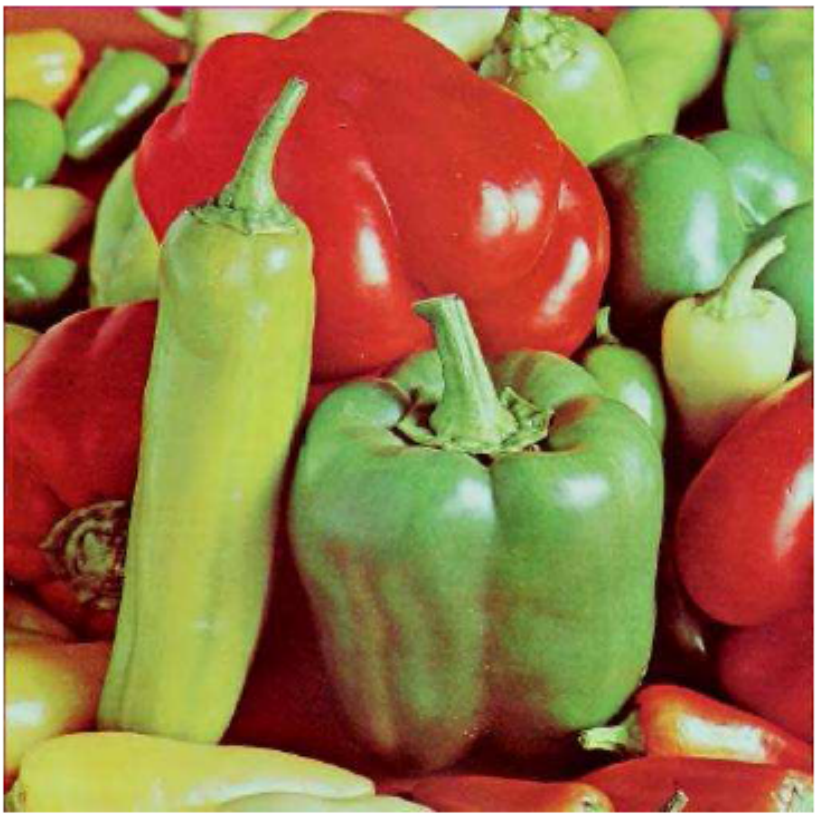


Figure (IV-54) : Image ‘poivron ’reconstruite par sa taille d’origine par le codeur JPEG2000.

IV-3-10. Test Scalabilité en résolution et Qualité sur une image couleur poivron

Le test de Scalabilité en résolution et qualité sur l'image couleur poivron de résolution 512 * 512 pixels codée sur 24 bpp compressée à ces trois taux de bits (0.1 bpp**,** 0.2 bpp**,** 0.5 bpp) en même temps est illustré par la figure (IV- 55) et figure (IV- 56) :

Résolution 256 x 256 pixels
(a)

Résolution 256 x 256 pixels
(b)

Résolution 256 x 256 pixels
(c)

Figure (IV-55) : Image ' poivron 'reconstruite divisée horizontalement et verticalement par 2 par rapport à l'image d'origine et à différents niveaux de qualité : (a) 0.1 bpp, (b) 0.2 bpp et (c) 0.5 bpp par le codeur JPEG2000.

Résolution 128 x 128 pixels
(d)

Résolution 128 x 128 pixels
(e)

Résolution 128 x 128 pixels
(f)

Figure (IV-56) : Image ‘ poivron ’reconstruite divisée horizontalement et verticalement par 4 par rapport à l’image d’origine et à différents niveaux de qualité : (d) 0.1 bpp, (e) 0.2 bpp et (f) 0.5 bpp par le codeur JPEG2000.

IV-3-11. Test Région d’intérêt (ROI) sur une image au niveau de gris Lena

La qualité d'une image peut ne pas être gérée de manière globale , on peut désormais attribuer à certaines régions de l'image un poids plus important que d’ autres régions. L’exemple suivant de la figure (IV-57) montre une image Lena compressée avec un taux de bits de 0.25bpp dans une zone de Région d'intérêt rectangulaire qui est ici le visage de Lena.

Figure (IV-57) : Image ‘Lena ’reconstruite après une compression à 0.25 bpp de la région d’intérêt par le codeur JPEG2000.

IV-3-12. Test Région d'intérêt (ROI) sur une image couleur Lena

Le résultat de test de Région d'intérêt sur l'image couleur Lena de résolution 512 * 512 pixels codée sur 24 bits / pixel compressée à 0.25 bpp dans une zone de Région d'intérêt rectangulaire est donné par la figure (IV-58) :

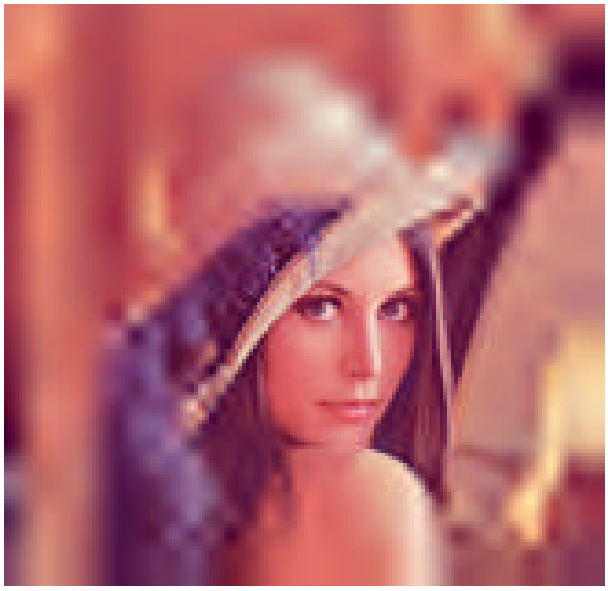


Figure (IV-58) : Image 'Lena 'couleur reconstruite après une compression à 0.25 bpp de la région d'intérêt par le codeur JPEG2000.

IV- 4. Conclusion

D'après les résultats obtenus, nous pouvons affirmer que la norme de compression JPEG2000, basée sur la transformée en ondelettes et pour les faibles valeurs du taux de bits , est plus performante que JPEG basé sur la TCD qui opère sur des blocs de taille 8x8 pixels. De même, lorsque nous augmentons la valeur du taux de bits, JPEG2000 surpasse JPEG. En effet, la différence entre les deux normes lorsque nous augmentons les valeurs du taux de bits n'est pas visuellement perceptible. En plus, cette nouvelle norme pour la compression des images fixes offre de nouvelles fonctionnalités telle que la compression avec perte ou sans perte avec le même standard, la Scalabilité en qualité, la Scalabilité en résolution , la Scalabilité en résolution et Qualité en même temps, et la compression par Région d'intérêt (ROI). Toutes ces fonctionnalités permettant de faire face aux besoins actuels dans de nombreux domaines tel que le domaine de l'imagerie médicale et l'Internet, la photographie numérique. Parmi certaines de ces fonctionnalités peuvent être utilisées en scanner, CD-ROM, et les applications d'archivage sur un Micro ordinateur personnel ou d'affaires. A titre d'exemple une image peut être stockée dans un fichier JPEG 2000 au moyen d'un scanner ou d'un CD-ROM. L'image peut ensuite être récupérée à des résolutions différentes selon l'application. Par exemple, une qualité d'impression élevée de l'image peut exiger une résolution de 1280× 1024 pixels, tandis que l'image ne peut être affichée à l'écran à plus 1024 ×800 pixels, et le processeur d'image peut avoir besoin d'accéder à une image sans toute erreur due à la compression. Toutes ces applications de JPEG-2000 peuvent être réalisées très facilement en utilisant le même fichier image. Cependant, la norme JPEG-2000 à un inconvénient qui nécessite plus de puissance de calcul que JPEG, ce qui n'est pas un problème pour un Micro ordinateur ordinaire, mais qui en est un pour un appareil aux ressources limitées comme téléphone portable. L'autre inconvénient est que JPEG2000 n'est pas encore utilisé par assez de constructeurs.

Conclusion générale

Conclusion générale

Dans ce travail, nous avons étudié différentes méthodes de compression d'images fixes à niveau de gris et couleur. Dans le premier chapitre nous avons étudié les méthodes de compression sans perte d'informations. Ces méthodes ont généralement un faible gain de compression qui est généralement lié aux propriétés statistiques de l'image.

Dans le deuxième chapitre nous avons fait une étude approfondie de la norme de compression JPEG (Baseline).

Dans le troisième chapitre nous avons étudié la technique de compression des images par la transformation en ondelettes, qui est la norme de compression JPEG2000, L'objectif de cette norme est d'apporter, en plus des nouvelles fonctionnalités nécessaires aux applications nouvelles, une nette amélioration de la qualité de reconstruction des images par rapport à JPEG-DCT et une meilleure résistance aux erreurs de transmissions. Cette norme à deux modes de compression :

· Un mode sans perte, réversible qui utilise des filtres (5,3) pour réaliser la transformation en ondelettes. Ces filtres sont associées à la transformée couleur réversible. Il n'y a évidemment pas de quantification.

· Un mode avec pertes, irréversible qui utilise des filtres (9,7) plus performants en termes de décorrélation de l'image mais qui n'ont pas une reconstruction parfaite. Ces filtres sont associées à la transformée couleur irréversible. Ce mode permet d'utiliser de la quantification.

Nous avons aussi étudié le principe de codeur entropique utilisé dans la norme pour coder les coefficients résultants de la transformation d'ondelette de l'image. Dans le dernier chapitre, une étude de performance de la technique de compression JPEG 2000 par rapport à JPEG a été faite. Des Programmes et des fonctions sous MATLAB 7 ont été développés afin de faciliter la compression et la décompression des images et l'évaluation des tests. Nous avons surtout évalué les performances en termes de qualité visuelle de l'image après décompression entre le codec JPEG2000 par rapport à JPEG dans le cas du mode avec perte pour différents taux de bits et pour différentes images (au niveau de gris , couleur et image médicale). Les résultats obtenus montrent que la norme JPEG2000 est meilleure que la norme classique JPEG. Mais cette supériorité de la performance de JPEG2000 diminue à mesure qu'on augmente la valeur du taux de bits. On peut dire que la compression d'images par ondelettes permet d'atteindre de très bons résultats. Le résultat en terme du PSNR et taux de bits diffère d'une image à l'autre et cela

est due au contenu des images qui différe (riche en détails ou en basse fréquences ou constitue un mélange). Et aussi nous avons évalué les nouvelles fonctionnalités apportées par JPEG2000 telle que la compression par Région d'intérêt (ROI) qui peut être exploitée dans le domaine d'imagerie médicale, le codage avec ou sans perte d'informations avec un seul standard, Décodage de l'image avec différents niveaux de résolution ou avec différents niveaux de qualité et aussi avec différents niveaux de résolution et qualité en même temps. Des tests ont été effectués pour déterminer un compromis entre les niveaux de décomposition d'ondelettes et la taille des code-blocs utilisés pour la compression des images. Les tests démontrent qu'une taille des code-blocs de 64x64 donne de bons résultats en terme de PSNR et cela pour tout les niveaux de décomposition d'ondelettes utilisés. Enfin, le domaine de la compression reste encore un sujet de recherche, et le but est d'arriver à une compression optimale en terme de temps de compression, de taux de bits et en terme de qualité de l'image reconstruite est toujours en course. La transformation en ondelettes reste une méthode à développer et à améliorer afin d'atteindre de meilleurs résultats par rapport à ce qui existe déjà .

Annexes

Programme principal d'utilisation du logiciel kakadu (Acquisition et compression et décompression des images et évaluation des tests)

```
% function KAKADU (p, name_in,nom_j2c,nom_sortie,ligne_de_commande_com,ligne_de_commande_dec);
% fonction compression / décompression d'images de type niveaux de gris (grayscale) au format PGM et de
% type couleur au format PPM avec le codeur Kakadu.
% Exemple d'utilisation sous Matlab, mettre les exécutables Dos du codec Kakadu dans le dossier work de
% Matlab avec les images à compresser.
% Signification des variables de la fonction KAKADU :
% p : choisir cette valeur = 1 si on veut compresser une image en niveaux de gris (grayscale ) ou p = 2 si on veut
% compresser une image couleur.
% name_in : Entrer le nom de l'image à compresser (d'origine).
% nom_j2c : Donner le nom de l'image compressée au format JPEG2000.
% nom_sortie : Donner le nom de l'image décompressée (reconstruite).
% ligne_de_commande_com: Spécifier les paramètres du codec pour effectuer la compression.
% ligne_de_commande_dec : Spécifier les paramètres du codec pour effectuer la décompression.
% Auteur : Sahir mourad.
% Université de Sétif.
function KAKADU(p, name_in,nom_j2c, nom_sortie, ligne_de_commande_com , ligne_de_commande_dec);
switch p
   case 1,
c = imread (name_in); % ouverture de l'image
imwrite(c,'graypgm.pgm'); % création de fichier au format PGM
% création d'une chaîne de caractères comprenant la ligne de commande
% du compresseur qui sera exécutée par la commande « dos » de MATLAB.
chaine_compresseur = ['kdu_compress -i graypgm.pgm -o ',nom_j2c,' ', ligne_de_commande_com]
dos (chaine_compresseur)
% chaîne de caractères pour la ligne de commande du décompresseur.
chaine_decompresseur = [' kdu_expand -i ',nom_j2c,' -o ',nom_sortie,' ', ligne_de_commande_dec]
dos (chaine_decompresseur)
compare (name_in,1,nom_j2c,nom_sortie,2)
   Case 2,
  c = imread (name_in); % ouverture de l'image
imwrite(c,'couleurppm.ppm'); % création de fichier au format PPM
% création d'une chaîne de caractères comprenant la ligne de commande
% du compresseur qui sera exécutée par la commande « dos » de MATLAB.
chaine_compresseur =['kdu_compress -i couleurppm.ppm -o ',nom_j2c,' ', ligne_de_commande_com]
dos(chaine_compresseur)
% chaîne de caractères pour la ligne de commande du décompresseur.
chaine_decompresseur = [' kdu_expand -i ',nom_j2c,' -o ',nom_sortie,' ', ligne_de_commande_dec]
dos (chaine_decompresseur)
compare2 (name_in, 1, nom_j2c, nom_sortie, 2)
 otherwise
  disp('Erreur du choix, il faut choisir p = 2 pour compresser une image couleur de type ppm ou choisir p = 1
pour compresser une image au niveaux de gris de type pgm.')
end
```

Programme d' évaluation des résultats des tests des images compressées et leurs affichages sur l'écran de l'ordinateur :

```
% function compare (name_in, num1, name_compresse_j2k, nom_sortie, num2) ;
% fonction comparant la dégradation entre une image d'origine et son image reconstruite (décompressée) et
% affiche l'image d'origine et décompressée et l'image erreur sur l'écran de l'ordinateur en même temps et
% l'histogramme de l'image d'origine et de l'image décompressée.
% image erreur = différence entre image originale et son image décompressée (reconstruite).
% la fonction calcul aussi EQM et SNR et PSNR entre l'image originale et l'image décompressée (reconstruite)
% et entre l'image originale et l'image erreur. La fonction calcul aussi la Taille image à compresser en Octets,
% taille image compressée en Octets, Quotient (Taille image à compresser /Taille image compressée), taux de
% bits en (bpp) et affiche les résultats sur l'espace de travail de Matlab.
% Exemple d'utilisation : Mettre l'image à compresser et compressée et décompressée (reconstruite) dans le
% dossier work de Matlab et dans l'espace de travail de Matlab saisir la commande suivante :
% compare ('lena.pgm ',1,'lena.j2k ','lenna.pgm ',2)
% lena.pgm : image d'origine à compresser.
% num1=1 : pour indiquer le numéro de la figure de l'image d'origine lena.pgm.
% lena.j2k : image compressée au format JPEG2000.
% lenna.pgm : image décompressée (reconstruite).
% num2= 2 : pour indiquer le numéro de la figure de l'image reconstruite lenna.pgm lors de l'affichage sur
% l'écran de l'ordinateur.
% Auteur : Sahir mourad.
% Université de Sétif.
function compare(name_in,num1,name_compresse_j2k,nom_sortie,num2)

[M_in,nb_comp1]=init_image(name_in);
[M_out,nb_comp2]=init_image(nom_sortie);

size(M_in);
size(M_out);
n1 = double (nb_comp1) ;
n2 = double (nb_comp2) ;
aff_image(num1,name_in); title(name_in);
aff_image(num2,nom_sortie); title(nom_sortie);
aff_image_err(name_in,nom_sortie);

if n1 ~= n2
  disp('erreur de Taille')
  return
end

fprintf('*** image à compresser = %s / image reconstruite(décompressée) = %s ***\n',name_in, nom_sortie);
for i=1:nb_comp1

        [A(i),B(i),C(i)] = mesure_PSNR(M_in(:,:,i),M_out(:,:,i));
        [A(i),B(i),C(i)] = mesure_PSNR(M_in(:,:,i),M_out(:,:,i));
        [A(i),B(i),C(i)] = mesure_PSNR(M_in(:,:,i),M_out(:,:,i));
        fprintf('composante %d : EQM = %f, SNR(dB) = %f, PSNR(dB) = %f \n',i,A(i),B(i),C(i));

end

[M_ou,nb_comp3]=init_image('image_erreur.pgm');
```

```
fprintf('*** image à compresser = %s / image de différence(image à compresser - image reconstruite) = %s
***\n',name_in, 'image_erreur');

for i=1:nb_comp1
        [A(i),B(i),C(i)] = mesure_PSNR(M_in(:,:,i),M_ou(:,:,i));
        [A(i),B(i),C(i)] = mesure_PSNR(M_in(:,:,i),M_ou(:,:,i));
        [A(i),B(i),C(i)] = mesure_PSNR(M_in(:,:,i),M_ou(:,:,i));
fprintf('composante %d : EQM = %f, SNR(dB) = %f, PSNR(dB) = %f \n',i,A(i),B(i),C(i));

end
Taille = taille_fichier(name_compresse_j2k);
Taille2 = taille_fichier(name_in);

[M,N] = size(M_in(:,:,i)) ;

Quotient = nb_comp1*M*N/Taille ;

fprintf('Taille image originale à compresser en Octets =  % d \n',Taille2);

fprintf('Taille de l' image compressée en Octets =  % d \n',Taille);
fprintf('Quotient(Taille de l' image originale/Taille de l' image compressée) = %g \n',Quotient);
fprintf('Taux de bits en bit par pixel =  (% g bpp) \n',Taille*8/(M*N));

%==========================================================
```

Utilisation du logiciel Kakadu Sous DOS

Les programmes «Kdu_compress .exe » et «Kdu_expand .exe » sont les exécutables sous DOS du logiciel Kakadu.

1. Compression

Les commandes suivantes sont utilisées dans une fenêtre de commande DOS afin de compresser une image en niveaux de gris pour un taux de bits de 2 bpp :

Kdu_compress -i lena.pgm -o lena.jp2 -rate 2

Avec -i identifie le fichier image à compresser ici lena et -o identifie le nom du fichier image compressée ici lena . L'extension. jp2, qui est le format pour la norme JPEG2000. Le -rate définit le taux de bits de l'image en bpp, ici l'image est compressée à 2bpp.

2. Décompression

Après la compression d'une image que l'on peut stocker le flux de bits dans une bibliothèque ou l'envoyer à travers un canal. La commande pour extraire l'image est:

Kdu_expand -i lena.jp2 -o lenna.pgm - reduce 1

Avec -i identifie les flux de bits avec l'extension. jp2. -o identifie le nom du fichier image décompressée ici lenna de format.pgm. - reduce permet de décoder l'image à plusieurs résolutions, ici dans notre cas - reduce 1 signifie récupérer l'image divisée en deux en horizontal et en vertical.

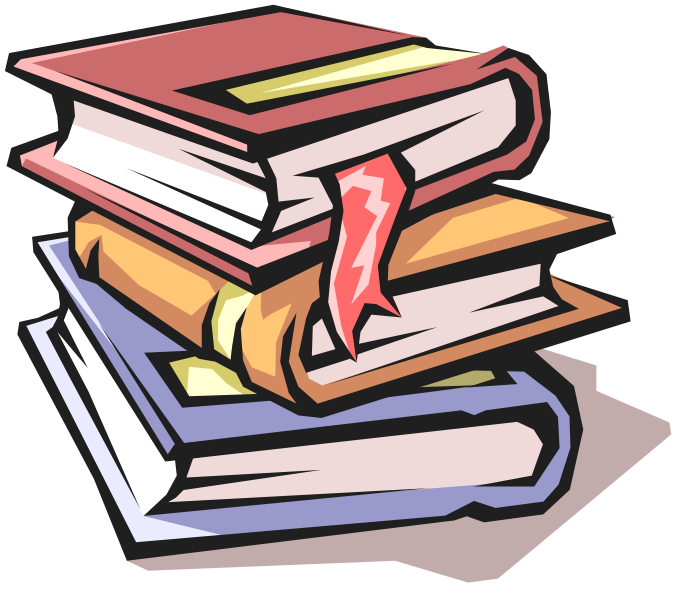
Tableau C-1 propose plus d'options et leur explication.

Commande	Explication
-i	Donner le nom de l'image à compresser
-o	Donner le nom du fichier de l'image décompressée
-rate	Spécifier le taux de compression
-reduce	Décoder l'image à plusieurs niveaux de résolution
Stiles	Décomposition de l'image en tilles
Rlevels	Choisir le nombre de décomposition en ondelettes
Cblk	Choisir la taille des code- blocs (par défaut = 64x64)

Tableau C-1 : Commandes du codeur JPEG2000.

Des explications plus détaillées peuvent être trouvées dans Usage_Examples.txt du logiciel.

Bibliographie



Bibliographie

[1] M. Nelson, “ La compression de données textes, Image, son ’’, Edition Dunod, 1993.

[2] A. N. Skodras, C. A. Christopoulos and T. Ebrahimi, “ JPEG2000: Still Image Compression Standard”, Proceedings Recpa00d 20, pp. 359-366, May 2000.

[3] Léger, A,“ Implementations of fast discrete cosine transform for full color videotex services and terminals”, In Proceedings of the IEEE Global Telecommunications Conference, IEEE Communications Society, pp. 333-337, 1984.

[4] Y. Wang and Q. Zhu,“ Error control and concealment for video communication: A review”, Proceedings of the IEEE, vol. 86, pp. 974-997. May 1998.

[5] M. Zhao and T.-M. Le,“ Robust Transmission of JPEG-2000 Images over Peer- to-Peer Mixed Wired and Wireless Links”, IEEE, Wireless Communications and Networking Conference (WCNC), 2005.

[6] N. Morceau, “ Techniques de compression des signaux ’’, Edition Masson, 1995.

[7] M. Kunt, “ Traitement numérique des signaux ’’, Edition Dunod, 1981.

[8] Alain et Ursula Bouteveille,“ La télévision numérique ’’, 4^{éme} Edition Dunod, Paris, 2006.

[9] D. A. Huffman, “ A method for the construction of minimum - redundancy codes” , Pro - ceedings of the Institute of Radio Engineers, vol. 40, pages 1098–1101. sept. 1952.

[10] P.G.Howard, J.S.Vitter, “ Arithmetic coding for Data Compression ” , Proc. of the IEEE, vol. 82, pp. 857-865, June 1994.

[11] R . J.Clarke, “ Digital Compression of Still Images and Video ” , Acadimic Press,1995.

[12] N. Jayant, J. Johnston, R. Safranek, “ Signal Compression Based on Models of Humain Perception ”, vol. 81, pp 1385-1422,October 1993.

[13] Sonja Grgic, Mislov Grgic and Branka Zorko-cihlar, “ Performance Analysis of Image Compression Using Wavelets” ,IEEE Trans on Industrial Electronics, vol 48, no. 3, June 2001.

[14] G.Sadashivappa, and K.V.S. AnandaBabu,“ Performance analysis of Image Coding Using Wavelets”, IJCSNS International Journal of Comp. Science and Network Security, Oct.2008.

[15] Min-Mo Sung, and Hee-Joung Kim,“ Clinical Evaluation of JPEG2000 Compression for Digital Mammography ” , IEEE Trans on Nuclear Science, vol .49, no. 3, June 2002.

[16] A. K. Jain, “ Image Data Compression: A Review”, Procedings of IEEE, vol. 69, no. 3, March 1981.

[17] G.K.Wallance, “ The JPEG Still Picture Compression Standard ”, IEEE Trans. On Consumer Electronics, December1991.

[18] Mitchell, J.L., and W.B. Pennebaker,“ Evolving JPEG Color Data Compression Standard ”, in Standards for Electronic Imaging Systems, Neir, M. and M.E. Courtot, eds., vol. CR37, SPIE Press, pp. 68-97, 1991.

[19] Wallace, G.K,“ Overview of the JPEG still image compression standard. Image Processing Algorithms and Techniques” , In Proceedings of the SPIE, vol. 1244 , pp. 220-233, Feb.1990.

[20] Wallace, Gregory K, “ The JPEG Still Picture Compression Standard ”, Communications of the ACM, vol. 34, no. 4, pp. 30-44, April 1991.

[21] Rao, K.R., and Yip, P,“ Discrete Cosine Transform - Algorithms, Advantages” , Appli -cations. Academic Press, Inc. London, 1990.

[22] C. Huang, S. Ravi , A. Raghunathan , and N. K. Jha ,“ Eliminating Memory Bottlenecks for a JPEG Encoder Through Distributed Logic-Memory Architecture and Computation-unit Integrated Memory ” , IEEE Custom Integreted Circuits Conference , 20005.

[23] Léger, A., J. Mitchell, and Y. Yamazaki, “ Still Picture Compression Algorithms Evaluated for International Standardization ” , Proceedings of the IEEE Global Telecommunications Conference, pp. 1028-1032, November 1988.

[24] Léger, A., T. Omachi, and T.K. Wallace,“ JPEG Still Picture Compression Algorithm”, Optical Engineering, vol. 30, no. 7, pp. 947-954, July 1991.

[25] Wallace, Gregory K,“ The JPEG still picture compression standard” ,IEEE Transa - ctions on Consumer Electronics. vol 38, no .1, pp. 18-34. 1992.

[26] Gregory K. Wallace , “ Information Technology – Digital Compression and Coding of Continuous-Tone Still Image – Requirements and Guidelines”, Copyright by ITU, 1993.

[27] W. B. Pennebaker, J. L. Mitchell,“ JPEG Still Image data compression standard ”, Van Nostrand Reinhold, NY, 1993.

[28] Klaus Jung and Ruedi Seiler, “ Segmentation and Compression of Documents with JPEG2000 ”, IEEE Trans. On Consumer Electronics, vol. 49, no. 4, November 2003.

[29] S.G. Mallat, “ A theory for multiresolusion signal decomposition : the wavelet representation’’, IEEE trans.Pattn.Anal. Mach. Intell. vol. 11, no7. pp.674-693. July 1989.

[30] D. Santa-Cruz, R. Grosbois, T. Ebrahimi, “ JPEG 2000 performance evaluation and assessment ”, Signal Processing: Image Communication 17 .pp.113–130,2002.

[31] B. E. Usevitch, “ A tutorial on modern lossy wavelet image compression: Foundations of JPEG 2000”, IEEE Signal. Processing Mag., vol. 18, pp .22-35, Sept. 2001.

[32] M.D. Adams, “ The JPEG2000 Still Image Compression Standard”, ISO/IEC JTC1 / SC29/ WG1 (ITU-TSG16), December 2005.

[33] C. Christopoulos, A. Skodras, T. Ebrahimi, " The JPEG2000 Still Image Coding System : An Overview", IEEE Transactions on Consumer Electronics, Vol. 46, No. 4,pp.1103-1127, November 2000.

[34] Y. Kumar Jain, S. Jain, " Performance Evaluation of Wavelets for Image Compression", Asian Journal of International Technology 5(10) , pp. 113–130,2002.

[35] D. Le Gall and A.Tabatabai, "Subband Coding of Digital Images Using Symmetric Short Kernel Filters and Arithmetic Coding Techniques", Proc IEEE Int. Conf. ASSP, NY, pp. 761-765, 1988.

[36] M. D. Adams and F. Kossentini, "Reversible Integer- To-Integer Wavelet Transforms For Image Compression: Performance Evaluation And Analysis", IEEE Trans. Image Processing, Vol. 9, No. 6, pp. 1010-1024, June 2000.

[37] M. Antonini, M. Barlaud, P. Mathieu and I.Daubechies: "Image Coding Using theWavelet Transform", IEEE Trans. Image Proc., pp. 205-220, April 1992.

[38] D. Taubman, " High Performance Scalable Image Compression with EBCOT", IEEE Transactions on Image Processing, vol. 9, no. 7, July 2000.

[39] D. Taubman, E. Ordentlich, M. Weinberger, G.Seroussi, I. Ueno, and F. Ono, " Embedded block coding in JPEG2000", in Proc. IEEE Int. Conf. Im- age Processing,Vancouver, Canada, vol. 2, pp. 33-36, Sept. 2000.

[40] A. Skodras, C. Christopoulos, and T. Ebrahimi, " The JPEG 2000 still image compression standard", IEEE Signal. Processing Mag., vol. 18, pp 36-58, Sept. 2001.

[41] M. Boliek, C. Christopoulos, et E. Majani," JPEG 2000 Part I Final Committee Draft Version 1.0 ", FCD15444-1, ISO/IECJTC1/SC29WG1 N1523 – décembre 1999.

[42] M. Boliek, C. Christopoulos, et E. Majani, " JPEG 2000 Part I Final Committee Draft Version 1.0 ", FCD15444-1, ISO/IEC JTC1/SC29 WG1 N1646R, april 2000.

[43] M. Boliek, C. Christopoulos, et E. Majani, " JPEG 2000 Part I Final Committee Draft Version 1.0 ", FCD15444-1, ISO/IEC JTC1/SC29 WG1 N164R, juillet 2001.

[44] C. Christopoulos, J. Askelof, and M. Larsson," Efficient methods for encoding regions of interest in the upcoming JPEG 2000 still image coding standard", IEEE Signal Processing Letters, vol. 7, no. 9, pp. 247–249,Sept. 2000.

[45] J. Askelof, M. Larsson Carlander, and C. Christopoulos, " Region of interest coding in JPEG 2000 " ,Signal Processing: Image Communication,vol. 17, no. 1, pp. 105–111, Jan.2002.

[46] http://www.kakadusoftware.com/

[47] http://www.ijg.org/

[48] http://www.med.univ-rennes1.fr/cerf/iconocerf/idx/T/THORAX.html.

ملخص:

الهدف من هذا العمل هو لدراسة ضغط الصور الرقمية الملونة و الغير الملونة بإستعمال طريقة الموجات ، و على وجه التحديد طريقة ضغط الصور JPEG2000 التي تعتمد علي أساس الموجات. لقد تلقت هذه الطريقة الجديدة إهتمام خاصا من العديد من الباحثين في السنوات الأخيرة . طريقة الضغط JPEG ناجحة جدا وتبقى وسيلة لضغط الصور فعالة جدا في مجال الكاميرات الرقمية و الانترنت، لكن بالنسبة للضغوط العالية تتدهور جودة الصورة بسرعة و بالإضافة إلى ذلك فهي قليلة المقاومة للأخطاء الناتجة عن الإرسال لسبب نوع الترميز المستعمل هوفمان . من الأهداف JPEG2000 الرئيسية هي تلبية إحتيجات متنوعة من التطبيقات الجديدة وتوفير تحسن واضح من حيث جودة الصورة ومقاومة لأخطاء الإرسال بالنسبة لي JPEG . اختبارات أداء المقارنة بين هذة الطريقة و طريقة JPEG من حيث نسبة الضغط وجودة الصور بعد بناؤها لعدة صور(الملونة و الغير الملونة و الطبية) تحت Matlab7 قد أجريت. النتائج المتحصل عليها بينت أن JPEG2000 هو الأفضل ، وكما قمنا أيضا بإجراء إختبارات علي التطبيقات الجديدة التي تقدمها JPEG2000 مثل المنطقة ذات الفائدة والترميز مع وبدون خسارة بخوارزمية واحدة ، فك الصورة مع مستويات دقة مختلفة أو مع مستويات مختلفة من الجودة وكذلك مع مختلف مستويات الجودة والدقة معا في نفس الوقت.

Abstract :

The objective of this work is to study the compression of digital still images to grayscale and color with wavelet technique , specifically JPEG 2000 (Joint Photographic Expert Group) still image compression standard based on wavelets . This new method, recently proposed in the literature, has received special attention of many researchers in recent years. The JPEG compression standard based on the DCT (Discrete Cosine Transform) is very successful and remains a method of image compression very efficient in the field of Internet and digital cameras. For high compression, the image quality degrades rapidly. In addition, JPEG offers very little resistance to errors due to variable length coding type Huffman used. JPEG2000 had as main objectives to meet the diverse needs of new applications and provide a clear improvement over JPEG in terms of quality and resistance to transmission errors. Comparative performance tests of this method in terms of bitrate and quality of the reconstructed images with the JPEG compression method (Baseline) in mode losses were done with varying test images (in grayscale , color and Medical) under Matlab7. The results show that JPEG2000 is better than the conventional JPEG standard and more tests were also done on the new features provided by JPEG2000 as the Region of Interest (ROI), coding with and without loss image with a single algorithm, decoding the image with different resolution levels or with different levels of quality and also with different levels of resolution and quality at the same time.

Keywords : JPEG2000, JPEG, DCT, color image coding, image compression, wavelet ,image quality.