République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Cheikh Larbi Tbessi –Tébessa-
Faculté des Sciences et de Technologies
Département Informatique
Ecole **D**octorale en **S**ciences et **T**echnologie de l'**I**nformation et de la **C**ommunication (INI)

Spécialité : **Informatique**
Option : **SIC**

Soutenance de Magister

Aura lieu le samedi 13 novembre 2010 à 8h30 au centre du calcul

---

Compression des images par l'algorithme SPIHT : Applications aux images fixes contenant des régions d'intérêt

---

Présenté par : **Souad SAADI**

Jury composé des personnes suivantes :

Président	**Pr.**	**Mohamed BENMOHAMED**	**Université de Constantine**
Rapporteur	**Pr.**	**Hamid SERIDI**	**Université de Guelma**
Examinateur	**Dr.**	**Med Kheiredine KHOLADI**	**Université de Constantine**
Examinateur	**Dr.**	**Allaoua CHAOUI**	**Université de Constantine**
Examinateur	**Dr.**	**AZEDINE BELAMI**	**Université de Batna**

2010

Dédicace

À mes parents,

à mes frères et sœurs,

À tous mes amis et collègues,

Je dédie ce travail.

Remercîments

Je remercie tout d'abord mon dieu le tout puissant qui m'a donné le courage, la paissance, et la volonté pour pouvoir terminer ce travail, ALHAMDO-LI- LELLEH.

Je remercie très sincèrement mon directeur de mémoire le professeur Hamid SERIDI, pour m'avoir dirigé pendant cette période de recherche ainsi pour Les conseils, la confiance et la liberté qu'il m'a accordés au cours de ce travail, m'ont permis d'entreprendre de nombreuses expériences qui ont grandement contribué à la réalisation de ce mémoire.

Mes remerciements vont également au membres de jury pour m'avoir honorer par leur participation à l'évaluation de mon travail.

J'aimerais bien remercie mes parents qui m'ont toujours aidé, soutenu, encouragé au cours de mes études.

J'aimerais adresser également un grand merci à mes amis, qui m'ont beaucoup encouragé afin de pouvoir continuer ce travail.

Je réserve un remercîment chaleureux à mes deux copines ***Rima*** *et* ***Zahra****.*

Je remercie tout ceux qui ont contribué à l'élaboration de ce mémoire.

Résumé

Le sujet central de ce mémoire est l'application de la méthode de codage SPIHT à la compression d'images fixes.

Dans la plupart des applications, surtout en imageries biomédicale ou satellitaire, L'utilisateur s'intéresse à l'extraction de l'information de certaines régions de l'image uniquement. Ceci suggère la localisation précise de ces régions que l'on qualifie de régions d'intérêt pour des fins de transmission, d'interprétation, etc.

Nous proposons dans le cadre de compression des images fixes une méthode de codage sélective. La sélection des régions d'intérêt utilise des techniques de segmentation par région. Ces régions subissent une compression en appliquant des techniques réversibles tel que SPIHT sans pertes. L'arrière plan, qui ne nécessite pas d'être codé en plaine résolution, est compressé à l'aide de JPEG2000 irréversible. Le but de la méthode est d'améliorer le taux de compression tout en conservant l'information qui se trouve dans les zones d'intérêt. La méthode a été testée sur un ensemble d'images incluant des régions d'intérêt de différentes natures et elle a montré des performances supérieures aux méthodes classiques telles que les méthodes définies dans la norme de JPEG2000 et JPEG.

Mots clefs Compression avec perte, compression sans perte, compression sélective, régions d'intérêt, extraction supervisée, SPIHT.

Abstract

The main subject of this work is the application of SPIHT to still image compression. In most applications, especially in biomedical or satellite imagery, the user is interested in extracting information from some parts of the image only. This suggests the precise location of these regions described as regions of interest for purposes of transmission, interpretation, etc..

The work presented the adaptive still image compression. We propose an adaptive still image compression method based on supervised extraction of the region of interest. The method consists in extracting the regions of interest using region segmentation techniques. These regions are thereafter compressed using lossless coding methods such as lossless SPIHT. The background is compressed using lossy JPEG2000 coding since it does not need to be coded in full resolution. The aim of the method is to improve the compression ratio and to save the information transmitted through region of interest. Our method has been tested on a set of images containing regions of interest and revealed higher performances compared to conventional methods.

Keywords, Lossless compression, lossy compression, adaptative compression, region of interest, supervised extraction, SPIHT.

الملخص

الموضوع الرئيسي لهذا العمل هو تطبيق طريقة لضغط الصور.
في معظم التطبيقات وخاصة في مجال الصور الطبية، الخبراء يهتمون أكثر لاستخراج المعلومات الموجودة في مناطق معينة فقط من الصورة، وهنا يفترض تحديد هذه المناطق التي تسمى بالمناطق ذات الأهمية وذلك لأغراض النقل و الترجمة.

لضغط الصور الثابتة نقترح الضغط الانتقائي للصور الثابتة. لاستخراج المناطق ذات الأهمية تستعمل عدة طرق لتجزئة الصور لعدة مناطق. هذه المناطق تخضع للضغط باستعمال طرق استرجاعية مثل طريقة SPIHT الاسترجاعية. الخلفية ليست بحاجة للضغط مع الاحتفاظ بكل المعلومات ، لهذا فهي تضغط بطريقة JPEG2000 غير الاسترجاعية. الهدف من الطريقة هو تحسين نسبة الضغط مع الاحتفاظ بالمعلومات الموجودة في المناطق ذات الاهمبة.
هذه الطريقة اختبرت على مجموعة من الصور تحتوي على مناطق ذات أهمية و قد أثبتت كفاءة عالية مقارنة مع الطرق الكلاسيكية.

الكلمات الرئيسية الضغط الإسترجاعي، الضغط غير الإسترجاعي، الضغط الانتقائي، مناطق الأهمية، الاستخراج الاشرافي، SPIHT.

Table des matières

1 Introduction générale **1**
2 Compression d'images numériques fixes **4**
2.1 Généralités sur l'image numérique 4
2.1.1 Types d'image numérique 4
2.1.1.1 Modes de représentation de l'image 4
2.1.1.2 Image binaire 5
2.1.1.3 Image à niveaux de gris 6
2.1.1.4 Image en couleurs 6
2.1.2 Caractéristiques de l'image numérique 7
2.1.2.1 Définition d'une image 7
2.1.2.2 Résolution d'une image 7
2.1.2.3 Profondeur de bits d'une image 7
2.1.3 Espaces colorimétriques 8
2.1.3.1 Synthèse additive RGB 8
2.1.3.2 Synthèse soustractive CMY 8
2.1.3.3 L'espace XY Z 9
2.1.3.4 Les systèmes luminance-chrominance YCbCr 10
2.1.3.5 Les systèmes perceptuels 10
2.2 Théorie de l'information 11
2.2.1 Information de source 11
2.2.2 Quantité d'information 11
2.2.3 Entropie 12
2.2.4 Codage source - codage canal 12
2.3 Formats image bruts 13
2.3.1 Les formats PNM 13
2.3.2 Le format BMP 13
2.3.3 Autres formats 13
2.4 Compression réversible généraliste 13
2.4.1 Codage par longueur de séquence 14
2.4.2 Codage à longueur variable 14

2.4.3 Codage arithmétique 19
2.4.4 Codage à base de dictionnaire 21
2.4.4.1 Fenêtre coulissante : LZ77 21
2.4.4.2 Construction dynamique de dictionnaire : LZ78 21
2.4.4.3 Extension de LZ78 : LZW 23
2.4.5 Compression sans pertes d'image 25
2.4.5.1 Application du RLE à la décomposition en plans de bits 25
2.4.5.2 Codage prédictif sans pertes 25
2.5 Compression avec pertes d'image 26
2.5.1 Quantification d'image 26
2.5.1.1 Quantification scalaire uniforme 26
2.5.1.2 Quantification de Lloyd-Max 26
2.5.1.3 Quantification vectorielle 27
2.5.2 Transformée de Fourier discrète : DFT 28
2.5.3 Transformée en cosinus discrète : DCT 28
2.5.4 Transformée en ondelettes discrète : DWT 29
2.6 Normes de compression d'image fixe 31
2.6.1 Norme JPEG 31
2.6.1.1 Préparation de l'image 32
2.6.1.2 Transformée en cosinus discrète 32
2.6.1.3 Quantification 33
2.6.1.4 Codage des DC et parcours en ZigZag 33
2.6.1.5 Codage entropique 33
2.6.2 Norme JPEG 2000 34
2.6.2.1 Prétraitement de l'image 35
2.6.2.2 La transformée en ondelettes discrète 35
2.6.2.3 La quantification 35
2.6.2.4 Le codage entropique 36
2.6.2.5 Formation du train binaire 36
2.7 Conclusion 36
3 Compression par régions d'intérêt des images fixes 37
2.1 Aperçu des méthodes existantes 38
3.3 Compression des zones d’intérêt 39
3.3.1 Extraction des zones d’intérêt 39
3.3.2 La compression par la méthode SPIHT 41
3.3.2.1 Introduction 41
3.3.2.2 Structures d’arbres 41

3.3.2.3 Principes généraux de EZW et SPIHT 42
3.3.2.4 Adaptation de EZW 43
3.3.2.5 Adaptation de SPIHT 44
3.3.2.6 Exemples de compression par la méthode SPIHT 49
3.3.2.7 Eléments pratiques de comparaison des différentes méthodes de compression 50
3.4 Méthode de compression proposée 52
3.4.1 Application à l'image avec une ROI 53
3.5 Conclusion 64
4 Conclusion générale **65**

Bibliographie **65**

Table des figures

Figure 2.1 A gauche une image matricielle, à droite une partie de l'image agrandie 5
Figure 2.2 A gauche une image matricielle, à droite une image vectorielle 6
Figure 2.3 La base des vecteurs RGB et le cube des couleurs dans l'espace XY Z 9
Figure 2.4 Diagramme de chromaticité de l'espace XY Z 10
Figure 2.5 Construction de l'arbre de Shannon-Fano 16
Figure 2.6 Construction de l'arbre du code de Huffman 17
Figure 2.7 Construction de code de Huffman en une seule passe pour la séquence "COMPRESSIONSANSPERTES" 18
Figure 2.8 Construction de code de Huffman en une seule passe pour la séquence 19
Figure 2.9 Exemple de codage d'une séquence d'ADN avec LZ77 22
Figure 2.10 Voisinage de DPCM 26
Figure 2.11 Quantification scalaire uniforme 27
Figure 2.12 Schéma de décomposition d'une image en sous-bandes 30
Figure 2.13 DWT de l'image « Lena » 30
Figure 2.14 Diagramme bloc du codeur JPEG 32
Figure 2.15 Découpage en blocs de 8X8 33
Figure 2.16 Quantification des coefficients DCT 34
Figure 2.17 Diagramme bloc du codeur JPEG2000 35
Figure 3.1 Processus d'échange des données entre deux niveaux 39
Figure 3.2 Génération des masques. 40
Figure. 3.3 Sélection des Zones d'Intérêt. 41
Figure 3.4 Illustration du principe des arbres de zéros 42
Figure 3.5 Ordre de parcours des coefficients pour EZW. 44
Figure 3.6 Terminologie SPIHT pour les descendants. 46
Figure. 3.7 Différences pour la descendance des coefficients basses fréquences entre EZW 46
Figure 3.8 Schéma de compression de la méthode SPIHT. 47
Figure 3.9 Lena image compression result at 0.3bpp 49
Figure 3.10 Scenery image compression result at 0.3bpp 49
Figure 3.11 City image compression result at 0.3bpp 50

figure 3.12 Comparaison de deux images codées au même débit par ondelettes (SPIHT) et par DCT (JPEG). 50

figure 3.13 (a) Goldhill (b) Baboon 51

figure 3.14 Comparaison de différentes méthodes de codage sur deux images 52

Figure 3.15 Schéma de codage appliqué aux images contenant une ROI 53

Figure 3.16 Image **poumons de face (originale)** : Dimensions=438x387 pixels.. Taille du fichier 92.90ko. Type de fichier=PNG 54

Figure 3.17 Image contenant la région d'intérêt compressée sans contexte: Dimensions=438x387 pixels. Taille de fichier=34.7 Ko 55

Figure 3.18 Image contexte compressée sans région d'intérêt : Dimensions=438x387 pixels. 66.60 ko. 55

Figure 3.19 Image **poumons de face** reconstruite 56

Figure 3.20(a) Image scanner présentant un abcès du cerveau : Dimensions=404x535 pixels. Taille de fichier=90,90Ko. Type de fichier=PNG. 56

Figure 3.21 Image contenant la région d'intérêt compressée sans contexte : Dimensions=404x535 pixels. Taille de fichier=15.8 Ko. 57

Figure 3.22 Image contexte compressée sans région d'intérêt : Dimensions=404x535 pixels. Taille de fichier=79.70 Ko. 57

Figure 3.23 Image scanner présentant un abcès du cerveau reconstruite. 58

Figure 3.23 Image de Lena Dimensions=512x512pixels. Taille du fichier 218ko. Type de fichier=PNG. 59

Figure 3.24 Image contenant la région d'intérêt compressée sans contexte : Dimensions=512x512 pixels. Taille de fichier=30.8 Ko. 60

Figure 3 .25. Image contexte compressée sans région d'intérêt : Dimensions=512x512 pixels. Taille de 155ko. 60

Figure 3.26 Image Lena reconstruite 61

Figure 3.27 Image de Stefan Dimensions=351x286pixels. Taille du fichier 94.8ko. Type de fichier=PNG. 61

Figure 3.28 Image contenant la région d'intérêt compressée sans contexte : Dimensions=351x286 pixels. Taille de fichier=23.5 Ko. 62

Figure 3.29 Image contexte compressée sans région d'intérêt : Dimensions=351x286 pixels. Taille de 79.7ko. 62

Figure 3.30 Image de Stefan reconstruite 63

Liste des tableaux

Table 2.1 Exemple de codes préfixés pour une source aléatoire 15
Table 2.2 Code de Shannon-Fano pour la séquence "CODAGEDESHANNONFANO" 16
Table 2.3 Construction du code arithmétique pour la séquence CARICATURE 20
Table 2.4 Codage LZ78 23
Table 2.5 Décodage LZ7 23
Table 2.6 Codage LZW 24
Table 2.7 Décomposition en plan de bits 25
Table 3.1 Résultats de la compression uniforme des images " **poumons de face** " et "Cerveau". *Compression sans perte. **Compression avec maximum de qualité. Ko :Kilo Octet. 58
Table 3.2 Résultats de la compression sélective des images " **poumons de face** "et "**Cerveau**".*Compression sans perte. **Compression avec maximum de qualité. Ko :Kilo Octet. R : Taux de compression. 59
Table 3.3 Résultats de la compression uniforme des images "**Lena**"et "**Stefan**". *Compression sans perte. **Compression avec maximum de qualité. Ko :Kilo Octet. 63
Table 3.4 Résultats de la compression sélective des images "**Lena**" et "Stefan".*Compression sans perte. **Compression avec maximum de qualité. Ko :Kilo Octet. R : Taux de compression. 64

Chapitre 1

Introduction générale

Les évolutions récentes des technologies de l'information ont agité le domaine des télécommunications.

Ainsi, l'apparition des techniques numériques a favorisé une diversification des utilisations du multimédia ainsi que des fonctionnalités qui leur sont associées. Cela implique l'usage de volumes de données sans cesse croissants. Il est donc nécessaire de disposer d'outils performants pour la transmission et le stockage de ces énormes quantités d'information. Les progrès techniques de ces dernières années, qui ont permis une augmentation conséquente des débits des réseaux ainsi qu'une baisse du prix de revient de la mémoire, apportent une partie de la réponse à ce problème. Parallèlement, on peut également chercher à diminuer la taille des fichiers de description des données.

C'est dans cette optique qu'ont été développées au cours des dernières décennies de nombreuses méthodes de compression de données déduites de la théorie de l'information et faisant appel à de nombreux domaines des mathématiques et de l'informatique.

Parmi les données les plus volumineuses, on compte les images et vidéos numériques qui ont de ce fait un besoin particulièrement important d'un traitement adapté à leurs spécificités. Les méthodes de compression de ce type de documents ont fait l'objet de nombreux travaux de recherche.

Dans la littérature, deux types de schémas de codage sont possibles conduisant à la compression avec pertes ou la compression sans perte des données. La compression sans perte

s'avère être une opération totalement réversible : l'image reconstruite correspond intégralement à l'image originale. Le codage avec pertes apparait quant à lui comme une opération irréversible, l'image reconstruite étant dégradée par rapport à l'image originale.
La compression avec pertes est largement utilisée dans les applications qui nécessitent un taux de compression élevé et qui ne requiert pas une qualité visuelle identique à la source, telles que la photographie numérique, la transmission à bas débit, les séquences vidéo, etc... Par contre, dans des domaines d'application comme l'imagerie biomédicale ou satellitaire, les pertes de données risquent de fausser des interprétations décisives. Les algorithmes de compression sans pertes sont, par conséquent, préférés dans ces types d'applications. Il est à noter que les images utilisées dans ce type d'application sont caractérisées par des résolutions très grandes. Ce qui génère des fichiers énormément volumineux à cause des limitations de la compression sans pertes en terme de taux de compression.

Cependant, l'utilisateur n'a besoin qu'à une partie de l'information transmise à travers l'image dans ce type d'application. A titre d'exemple, les mammographies numérisées et les images satellitaires mesurent des dizaines de mégaoctets, alors que les régions d'intérêt constituent moins de 30% de l'image. Pour remédier à ceci il est recommandé de traiter les différentes zones de l'image par des approches différentes réversibles ou irréversibles, pour cela nous avons utilisé la méthode de codage SPIHT pour coder les régions d'intérêt parce que elle permet de coder et reconstruire une image quasiment identique à l'originale et nous avons choisi la méthode compression JPEG200 pour coder le reste de l'image. La question qui intervient à ce niveau est comment appliquer l'extraction des régions d'intérêt de manière à cerner les parties intéressantes de l'image? Et comment améliorer le processus de codage par régions d'intérêt pour éviter toutes pertes d'information au niveau de celles-ci ?

Les objectifs visés par ce travail sont :

- Faire un tour d'horizon des méthodes de compression d'images.
- Coder les différentes régions de l'image avec des qualités différentes. Pour cette raison il est nécessaire d'exploiter la méthode de compression SPIHT pour coder la partie intéressante de l'image sans perte et la méthode de compression JPEG2000 pour coder le reste de l'image avec une dégradation de qualité.

Ce mémoire est organisé selon le plan suivant : Nous avons commencé, dans le deuxième chapitre, par introduire les notions de base liées à l'image numérique et aux différentes techniques de compression de données.

Nous détaillons dans le troisième chapitre le principe de base des méthodes introduites pour la compression adaptative , nous présentons également dans ce chapitre la méthode SPIHT pour la compression des images fixe qui sera utilisée par la suite pour coder les ROIs et ensuite, nous décrivons la méthode de compression par régions d'intérêt proposée .A la fin de ce chapitre, nous donnons les résultats obtenus en testant cette méthode sur différents types d'images. Enfin, nous terminons notre mémoire par une conclusion générale et des perspectives.

Chapitre 2

Compression d'images numériques fixes

La compression d'image est une technique qui exploite les redondances de données dans l'image pour réduire le nombre de bits nécessaires au codage de celle-ci afin de faciliter son stockage ou sa transmission. La compression d'image peut être effectuée avec ou sans pertes. La compression sans pertes consiste à coder les données de façon réversible. Autrement dit, les données sont reproduites de façon exacte après le cycle de codage/décodage. La compression avec pertes consiste à coder les données de façon irréversible en utilisant des techniques qui produisent des approximations de ces données. Les données ne sont pas reproduites de façon exacte après décodage. La compression avec pertes permet d'augmenter significativement le taux de compression par rapport à la compression sans pertes. Cependant, cette augmentation peut être accompagnée d'une dégradation de la qualité visuelle de l'image.

Ce chapitre va introduire les notions de base liées à l'image numérique et aux différentes techniques de compression de données.

2.1 Généralités sur l'image numérique

2.1.1 Types d'image numérique

2.1.1.1 Modes de représentation de l'image

L'image matricielle

L'image numérique est représentée dans la majorité des applications sous forme matricielle. La scène est échantillonnée horizontalement et verticalement en un ensemble de points élémentaires appelés pixels. Ce qui donne un tableau à deux dimensions de pixels.

Cette représentation est directement adaptée pour l'affichage sur un écran ou sur n'importe quel dispositif d'affichage. Cependant, la représentation matricielle nécessite un espace de stockage relativement élevé. En plus, l'agrandissement de l'image entraîne une distorsion de la qualité visuelle, on parle d'une image pixellisée. (Figure 2.1)

Figure 2.1 A gauche une image matricielle, à droite une partie de l'image agrandie

L'image vectorielle

L'image vectorielle est codée sous forme d'une description mathématique des formes géométriques qui la constituent (exemple : un disque est décrit par son rayon, les coordonnées de son centre et éventuellement les couleurs de bordure et de remplissage). Ce mode de représentation est caractérisé par le fait de pouvoir redimensionner l'image sans dégradation de la qualité visuelle de celle-ci. En revanche, il n'est adapté qu'aux figures géométriques (dessins, schémas, logos, ...). La représentation d'une scène par ce mode nécessite une puissance de calcul élevée et une mémoire importante. En plus, la visualisation des images vectorielles nécessite leur conversion en matricielle et ceci engendre aussi des calculs sur la machine.

2.1.1.2 Image binaire

Une image binaire est une image matricielle où chaque pixel ne peut prendre que deux valeurs (0 ou 1) qui correspondent généralement aux deux couleurs : le noir et le blanc. Parfois ces deux valeurs peuvent correspondre à d'autres couleurs.

Généralement, un bit est suffisant pour coder chaque pixel d'une image binaire. Cependant, il y a des formats qui ne prennent pas en considération ce type d'image. Dans ce cas chaque pixel est codé sur 1 octet.

L'image binaire est très utilisée dans les applications basées sur le seuillage et la segmentation. Souvent, elle permet de distinguer deux types de régions dans une image par

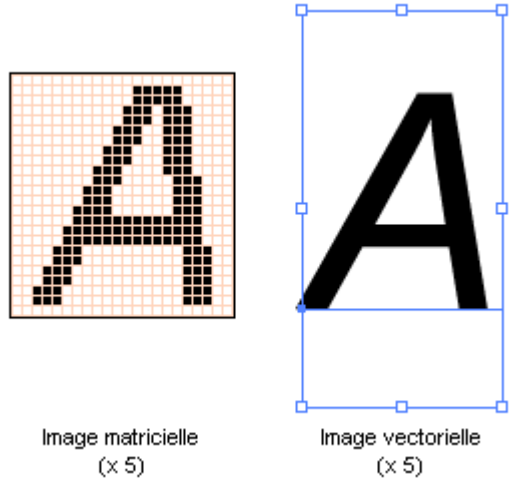


Figure 2.2 A gauche une image matricielle, à droite une image vectorielle

exemple. Elle est aussi utilisée pour le codage de fax. Le CCITT a développé des normes pour ce type de codage. Ces normes sont basées sur des algorithmes spécialisés tels que RLE ou QM-Coder [1].

2.1.1.3 Image à niveaux de gris

Une image à niveaux de gris est une image où chaque pixel est représenté par un niveau d'intensité entre le blanc et le noir. Sachant que l'œil humain perçoit en moyenne 200 niveaux d'intensité, une échelle de gris de 256 valeurs (8 bits par pixel) est suffisante pour représenter un dégradé qui semblera uniforme [1].

2.1.1.4 Image en couleurs

Dans une image en couleurs, chaque pixel est généralement représenté sous forme d'une combinaison de trois composantes pour donner une couleur. Chaque composante est codée sur un octet. Ce qui donne trois octets par pixel. Cependant, il y a des formats qui

acceptent une représentation indexée. Les couleurs utilisées dans une image sont rangées dans une palette, chaque couleur est indexée par un code sur un octet au lieu de représenter la couleur sur trois octets. Celle-ci est remplacée par le code approprié. Ce mode de codage est utilisé dans le cas d'image contenant un nombre de couleurs inférieur à 256. Les composantes d'un pixel sont représentées dans des espaces de couleur bien spécifiques.

2.1.2 Caractéristiques de l'image numérique

2.1.2.1 Définition d'une image

On appelle définition d'une image le nombre de pixels par largeur que multiplie le nombre de pixels par hauteur. C'est-à-dire, le nombre de colonnes d'une image multiplié par le nombre de lignes. Par exemple une image de 512 lignes et 256 colonnes a une définition de 512 pixels par largeur que multiplie 256 pixels par hauteur, et on écrit 512x256.

2.1.2.2 Résolution d'une image

On appelle résolution d'une image le nombre de pixels par unité de surface exprimée en points par pouce ou par centimètre. Cette information est utilisée pour établir un lien entre le nombre de pixels de l'image numérique et la taille réelle de sa représentation dans un support physique comme dans le cas d'impression de l'image sur une feuille. Souvent, les notions de définition et de résolution de l'image sont confondues car elles renseignent toutes les deux sur la précision des informations représentées.

2.1.2.3 Profondeur de bits d'une image

La profondeur de bits d'une image numérique exprime le nombre de bits qui codent un pixel. Elle représente en quelque sorte la troisième dimension de l'image en plus de la largeur et la hauteur. La profondeur de bits renseigne sur le nombre de couleurs maximal que peut contenir l'image. Par exemple, un bit par pixel permet d'avoir seulement une image binaire, 8 bits par pixel permet de coder un maximum de 256 couleurs, tandis qu'on peut arriver jusqu'à 16777216 couleurs avec 24 bits par pixel.
Le taux binaire est aussi exprimé en bits par pixels, mais sa signification est différente. Par exemple, un taux binaire égal à un bit par pixel ne signifie pas qu'il correspond à une image

binaire [2]. Mais que la taille de l'image compressée correspond à une moyenne d'un bit par pixel.

2.1.3 Espaces colorimétriques

2.1.3.1 Synthèse additive RGB

Le système colorimétrique RGB pour Red Green Blue, ou RVB pour Rouge Vert Bleu est le plus utilisé actuellement. Par définition, toute couleur C de coordonnées (r, g, b) peut être exprimée par addition de trois couleurs primaires R, G et B, ce principe est appelé synthèse additive [2].

On obtient : C = rR + gG + bB Cette équation caractérise uniquement la teinte de la couleur résultant de la synthèse additive [3]. Pour caractériser la luminance de cette couleur, il est nécessaire de connaître la luminance de chacune des trois couleurs primaires R, G et B. En stockant séparément le pourcentage de chaque composante (r, g, b). Les couleurs peuvent être codées sur 16 bits (5 bits pour le rouge et le bleu et 6 bits pour le vert, ce qui donne 65536 couleurs) ou 24 bits (8 bits pour chaque composante ce qui donne au total 16 millions de couleurs). Les composantes d'une couleur représentée dans l'espace RGB peuvent être des pourcentages (valeurs entre 0 et 1, exemples : le rouge (1,0,0), le noir (0,0,0), le vert (0,1,0), le blanc (1,1,1)) comme elles peuvent être des valeurs (pour une profondeur de 24 bits, le blanc correspond au (255,255,255), le bleu au (255,0,0), le noir au (0,0,0) et ainsi de suite).

2.1.3.2 Synthèse soustractive CMY

Dans l'espace RGB, les triplets (0,1,1), (1,0,1) et (1,1,0) correspondent respectivement aux couleurs cyan, magenta et jaune qui sont complémentaires des couleurs rouge vert et bleu. Les couleurs cyan, magenta et jaune sont utilisées comme primaires pour l'espace CMY basé sur le principe de la synthèse soustractive, par opposition à l'espace RGB qui est basé sur la synthèse additive [1]. L'espace CMY est le complémentaire de l'espace RGB. L'absence de couleur correspond dans ce cas au blanc (de composante (0,0,0)) tandis que la couleur noir correspond au triplet (1,1,1). L'espace RGB est utilisé pour l'affichage à l'écran qui est basée sur la lumière (absence de couleur correspond au noir) tandis que l'espace CMY est mieux adapté à l'impression (sur un papier blanc, l'absence de couleur correspond au blanc). La transformation qui permet de passer de l'espace RGB à l'espace complémentaire CMY est donnée par :

$$\begin{bmatrix} C \\ M \\ G \end{bmatrix} = \begin{bmatrix} 0\,1\,1 \\ 1\,0\,1 \\ 0\,1\,1 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

2.1.3.3 L'espace XY Z

L'espace tridimensionnel de couleur CIE XY Z est la base de tous les systèmes de gestion de couleur. Cet espace contient toutes les couleurs qui peuvent être perçues par l'œil humain. Parmi ces couleurs, il y a celles qui ne peuvent pas être affichées à l'écran ni imprimées. Pour éviter d'avoir des valeurs négatives dans l'espace RGB, la CIE (Commission Internationale d'Eclairage) a introduit en 1931 un nouveau système de coordonnées XY Z. Le système RGB est défini principalement par trois vecteurs qui constituent une base non orthogonale dans l'espace XY Z (figure 2.3).

Les couleurs dans l'espace XY Z sont déterminées par projection sur un plan en utilisant les coordonnées normalisées (x; y; z) suivantes :

$$x = \frac{X}{X+Y+Z} \qquad y = \frac{Y}{X+Y+Z} \qquad z = \frac{Z}{X+Y+Z}$$

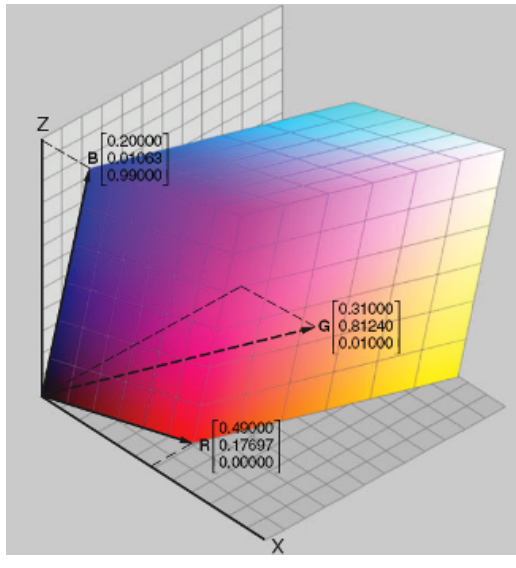


Figure 2.3 La base des vecteurs RGB et le cube des couleurs dans l'espace XY Z

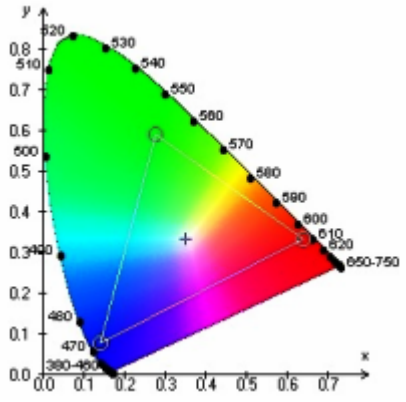


Figure 2.4 Diagramme de chromaticité de l'espace XY Z

Etant donné que x + y + z = 1, z se déduit directement des deux autres. Les coordonnées de ce plan sont appelées coordonnées de chromaticité. Pour visualiser l'espace XYZ, le diagramme de chromaticité est créé pour l'espace couleur CIE XY Z (figure 2.4). La surface délimitée par le triangle représente les couleurs qui peuvent être affichées par un moniteur. Le signe "+" indique le point blanc du moniteur.

2.1.3.4 Les systèmes luminance-chrominance YCbCr

La représentation en luminance-chrominance consiste à séparer la composante luminance Y qui décrit les détails spatiaux de l'image (contours, textures,...) des composantes de chrominance Cb et Cr qui renseignent sur la couleur. En gardant que la luminance on obtient une image en niveaux de gris. Ce système a été à l'origine introduit pour assurer une compatibilité entre les téléviseurs en couleurs et les téléviseurs en noir et blanc [2]. Les coefficients de passage du système RGB vers YCbCr varient selon les standards de télévision (NTSC, PAL ou SECAM) utilisés. Ce qui a donné naissance à plusieurs systèmes de type YCbCr tels que le système YUV qui correspond à la norme PAL, le système Y IQ qui correspond à la norme NTSC et le système Y DbDr qui correspond à la norme SECAM.

2.1.3.5 Les systèmes perceptuels

Les systèmes perceptuels sont des systèmes uniformes au sens de la perception visuelle. C'est-à-dire, ces systèmes peuvent décrire fidèlement, au sens de la perception visuelle, les différences entre couleurs proches. Dans cette famille, on distingue le système CIE L*a*b* qui est considéré parmi les systèmes de référence de la CIE pour calculer des

écarts couleur [2]. La transformation des composantes RGB vers CIE L*a*b* nécessite un passage par le système XY Z. La transformation qui permet de passer de l'espace XY Z à l'espace L*a*b* n'est pas linéaire et prend en compte les coordonnées trichromatiques du blanc de référence. Un autre système CIE L*u*v* appartient à la famille des systèmes perceptuels. Sa structure présente des similitudes avec celle de L*a*b*. Cependant, il existe quelques différences entre ces deux systèmes sur certaines plages de couleur [4]. Les espaces L*a*b*et L*u*v* appartiennent également à la famille des systèmes luminance-chrominance, puisque L* décrit la luminance et a*, b*, u*et v* décrivent la chrominance.

2.2 Théorie de l'information

2.2.1 Information de source

On appelle source ou signal toute information ou ensemble d'événements générés par un processus. Cette information est susceptible d'être codée ou numérisée. Le signal peut prendre la forme d'un texte, d'une voix, d'une image ou de n'importe quelle évolution d'un phénomène naturel. La numérisation des signaux de source permet de profiter des atouts de l'informatique, à savoir la puissance de calcul, la facilité de stockage, la fiabilité de transmission,...

On distingue plusieurs types de source selon la nature des événements et les corrélations entre les événements voisins. Par exemple, une source constante est une source constituée par des événements identiques, une source aléatoire est une source constituée par des événements indépendants les uns des autres, une source markovienne est une source composée par des événements corrélés, c'est-à-dire la production d'un événement est conditionnée par les événements précédents et elle peut influencer la production des événements suivants.

2.2.2 Quantité d'information

La quantité d'information de source est définie par le nombre de bits nécessaires à la représentation binaire des codes attribués aux événements ou symboles qui constituent cette information. Par exemple pour un texte ASCII, chaque caractère est codé sur 7 bits, si ce texte est composé de 1000 caractères alors sa taille sera de 7000 bits. Si on dispose d'une image en couleurs de définition 256x256 et de profondeur égale à 24 bits par pixel, le nombre total de bits sera égal à 256x256x24=1572864.

On appelle source discrète sans mémoire tout signal aléatoire de longueur infinie codé sur un alphabet dénombrable [1]. Supposant qu'un émetteur saisie des caractères au clavier. Ce signal est considéré comme source discrète sans mémoire si les caractères issus du clavier sont indépendants les uns des autres. Cependant, dans la réalité les caractères d'un message sont toujours dépendants les uns des autres [5]. Par exemple si l'émetteur saisie les caractères suivants : m, o, n, s, i, e, u. Un connaisseur de la langue française sait que le prochain caractère sera un "r". Cela montre que le huitième caractère est lié aux sept qui précédent. Dans ce cas, le huitième caractère possède une probabilité égale à 1.

La quantité d'information propre d'un caractère est donnée par :

$$I(c_i) = \log 2\left(\frac{1}{p(c_i)}\right) \qquad (2.1)$$

avec c_i est le $i^{ième}$ caractère ou symbole, i est le numéro du caractère dans l'alphabet complet, comprenant tous les caractères utilisables dans un texte. $p(c_i)$ est la probabilité d'apparition du $i^{ième}$ caractère de l'alphabet complet.

2.2.3 Entropie

La notion d'entropie a été introduite par Shannon pour mesurer l'incertitude sur la nature d'un message donné par rapport au message qui le précède [1]. L'entropie est nulle s'il n'existe pas d'incertitude (signal invariant, ou signal infini constant). Inversement, l'entropie est maximale dans le cas d'un signal aléatoire.

A chaque symbole c_i d'une source S est associée une quantité d'information $I(c_i)$.

La valeur moyenne de $I(c_i)$ définit l'entropie de la source S :

$$H(S) = \sum_{i=1}^{m} p(c_i).I(c_i) = \sum_{i=1}^{m} p(c_i).\log 2\left(\frac{1}{p(ci)}\right) \qquad (2.2)$$

où m est le nombre de caractères constituant la source S.

2.2.4 Codage source - codage canal

Le codage source rassemble l'ensemble des techniques de codage et de compression des symboles de la source avant la transmission dans un canal. Tandis que le codage canal est

l'ensemble des outils qui permettent d'adapter le signal source aux propriétés du canal qui va le supporter.

2.3 Formats image bruts

Les formats image bruts, aussi appelés bitmap permettent de stocker une image sous la formes d'intensités lumineuses facilement lisibles. Elles sont stockées sans pertes, rarement avec une compression.

2.3.1 Les formats PNM

PNM pour Portable aNy Map est une classe de formats lisibles par la plupart des logiciels classiques, indépendamment de la plateforme de travail. Les intensités de l'image sont stockées, après un en-tête, sous forme ASCII ou binaire. Cette classe regroupe les formats : PBM (Portable Bit-Map) pour les images binaires, PGM (Portable Gray-Map) pour les images à niveaux de gris et PPM (Portable Pix-Map) pour les images couleur.

2.3.2 Le format BMP

Le format BMP (BitMap) est le format de base sous environnement Windows. Sa structure est un peu plus complexe que les formats PNM et accepte une compression RLE et l'utilisation d'une palette de couleur : pour une profondeur de 8, 4 ou 1 bits par pixel les couleurs utilisées dans l'image sont indexées dans une palette, chaque couleur est codées sur 24 bits est indexées par un code de 8, 4 ou 1 bit qui sera utilisé à la place des 24 bits.

2.3.3 Autres formats

De nombreux autres formats, proposent également un codage brut. Comme par exemple les formats PNG et TIFF. Cependant, leur en-tête les rend assez volumineux et moins pratiques à l'utilisation que les formats PNM.

2.4 Compression réversible généraliste

La compression réversible dite aussi compression sans pertes ou sans distorsion consiste à représenter les données de la source avec un nombre réduit de bits de façon à avoir

une reproduction exacte des données après le cycle de codage / décodage. La compression réversible exploite les redondances qui existent dans le signal source. Le théorème du codage source sans bruit [1,2] nous assure qu'il est possible de réduire la quantité de bits issus d'un capteur. Ce théorème nous garantit aussi qu'on peut s'approcher aussi très bien du seuil de l'entropie mais que l'on ne pourra pas le dépasser. Les paragraphes qui suivent donnent un aperçu sur les techniques qui mettent en évidence les constatations issues de ce théorème.

2.4.1 Codage par longueur de séquence

Le codage par longueur de séquence, appelé aussi codage par plage en anglais Run Length Encoding (RLE), exploite le fait que les signaux, surtout les images numériques, ont la propriété de présenter de grandes plages parfaitement uniformes.
Ce codage consiste à représenter une séquence d'intensités de mêmes valeurs par un couple $P_i = \{p_i; l_i\}$ p_i est l'intensité, l_i est la longueur de la plage. Par exemple la séquence 70 70 70 70 53 53 53 70 70 70 42 42 70 43 43 43 sera codée de la façon suivante : $P_0(70, 4)$ $P_1(53, 3)$ $P_2(70, 3)$ $P_3(42, 2)$ $P_4(70, 1)$ $P_5(43, 3)$.

Le codage RLE est utilisé dans plusieurs formats d'image tels que BMP, TIFF, RASTER... Il est relativement performant sur des images synthétiques simples (issues d'un dessin par exemple). Néanmoins, il devient désavantageux sur des images complexes.

2.4.2 Codage à longueur variable

Les codes à longueurs variables sont une mise en œuvre directe et simple du théorème du codage source sans bruit [5]. La construction de tels codes est conditionnée par le fait que le code soit préfixé pour éviter le problème de synchronisation au moment de décodage (rétablissement des frontières entre les mots binaires successifs). Le code est dit préfixé lorsqu'aucun des mots qui le composent n'est le début d'un autre.

Exemple de codes préfixés

Le tableau 2.1 compare un code à longueur fixe optimal et des codes à longueur variable préfixés, construit "à la main", pour le codage en binaire d'une séquence aléatoire (équiprobable) sur l'alphabet a,b,c,d,e,f. On peut remarquer qu'il est possible d'arriver à un code de longueur moyenne de 2,66 bits par symbole. Et on peut arriver à une longueur moyenne encore inférieure si la séquence est issue d'une source non équiprobable :

Symboles	CLF	Exemple 1	Exemple 2	Exemple 3	Exemple 4	Exemple 5
a	000	0	0	0	00	00
b	001	10	10	100	01	01
c	010	110	110	101	10	100
d	011	1110	1110	110	110	101
e	100	11110	11110	1110	1110	110
f	101	111110	11111	1111	111	111
Taux binaire	3.00	3.50	3.33	3.00	2.83	2.66

Table 2.1 Exemple de codes préfixés pour une source aléatoire

Codage de Shannon-Fano

L'algorithme de codage de Shannon-Fano permet de produire automatiquement un codage à longueur variable préfixé en se basant sur les fréquences (probabilités) d'apparition des symboles. L'idée de base de cet algorithme est de coder les symboles les plus fréquents par un nombre réduit de bits. Le codage de Shannon-Fano optimal s'obtient par l'algorithme suivant :

- Classer les symboles de la séquence par probabilités décroissantes.
- Séparer les symboles en deux sous-bloques de sorte que les fréquences cumulées des deux sous-blocs devient très proches.
- Concaténer 0 à droite de tous les symboles du sous-bloc de gauche et 1 à ceux du sous-bloc de droite.
- Recommencer récursivement sur chacun des deux sous-blocs.
- L'algorithme s'arrête lorsqu'un sous-bloc n'a plus qu'un seul élément.

Exemple : Soit la séquence S ="CODAGEDESHANNONFANO", cette séquence contient 19 caractères à 10 symboles A = {N, A, O, D, E, C, G, H, S, F}. Le codage binaire standard (256 caractères) nécessiterait 8 bits par symbole, soit un total de 152 bits pour coder la séquence. Un codage à longueur fixe nécessiterait $E(\log_2(10) + 1) = 4$ bits par symbole, soit 76 bits pour la séquence. L'entropie de la séquence est de 3.11 bits par symbole, soit un total de 60 bits. Le tableau 2.2 montre pour cette séquence un code de Shannon-Fano obtenu à partir de l'arbre illustré dans la figure 2.5.

Symbole $_{occurrence}$	CLF optimal	Codes
N_4	0000	00
A_3	0001	010
O_3	0010	011
D_2	0011	100
E_2	0100	1010
C_1	0101	1011
G_1	0110	1100
H_1	0111	1101
S_1	1000	1110
F_1	1001	1111
	4b/s	3.15b/s

Table 2.2 Code de Shannon-Fano pour la séquence "CODAGEDESHANNONFANO"

Codage de Huffman

Le codage de Huffman est un autre algorithme qui permet de construire des codes à longueur variable préfixés très proches de ceux produits par Shannon- Fano. Le principe de cet algorithme ressemble beaucoup à celui de Shannon-Fano, la différence se trouve au niveau de la construction de l'arbre. Le codage de Huffman peut se dérouler en une passe ou en deux.

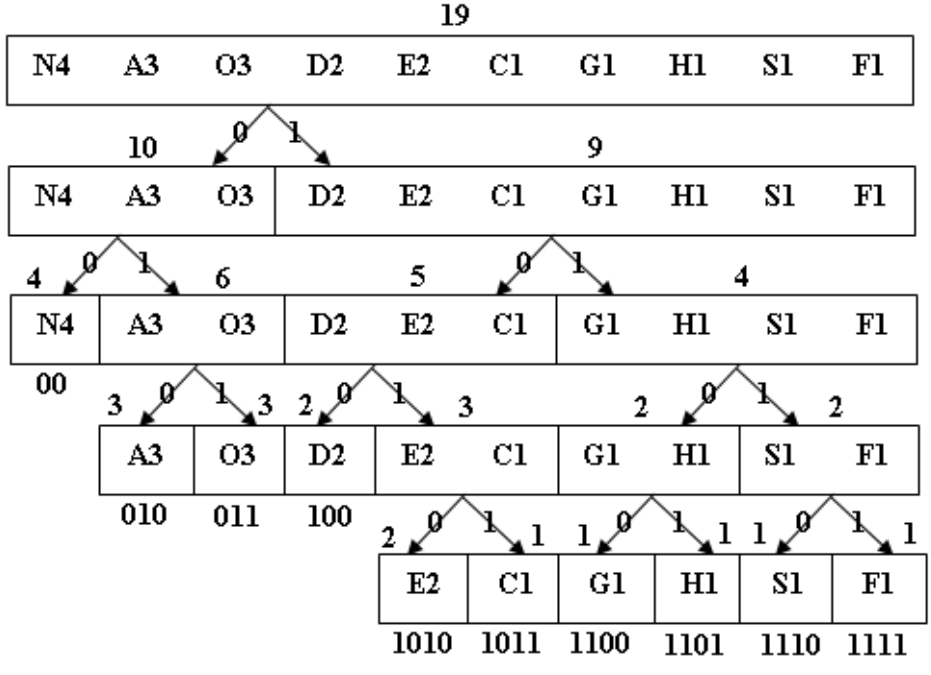


Figure 2.5 Construction de l'arbre de Shannon-Fano

L'algorithme de Huffman en deux phases est le suivant :

1. 1ère phase : Construction de l'arbre.

- Trier les symboles par probabilités croissantes.
- Fusionner les arbres correspondants aux symboles de probabilités minimales pour avoir un nœud de probabilité égale à la somme des deux.
- Supprimer les arbres de probabilités minimales de la table et y introduire l'arbre dont la racine correspond au nœud construit à l'étape précédente.
- Arrêter lorsqu'il reste un seul arbre dont toutes les feuilles sont les symboles de l'alphabet.

2. 2ème phase : Construction de code

   La construction de code est réalisée en parcourant l'arbre obtenu de la racine vers les feuilles en attribuant 0 aux sous-arbres de gauche et 1 aux sous-arbres de droite.

Exemple : La figure 2.6 montre un exemple de la construction de l'arbre et du code pour la séquence "CARICATURE".

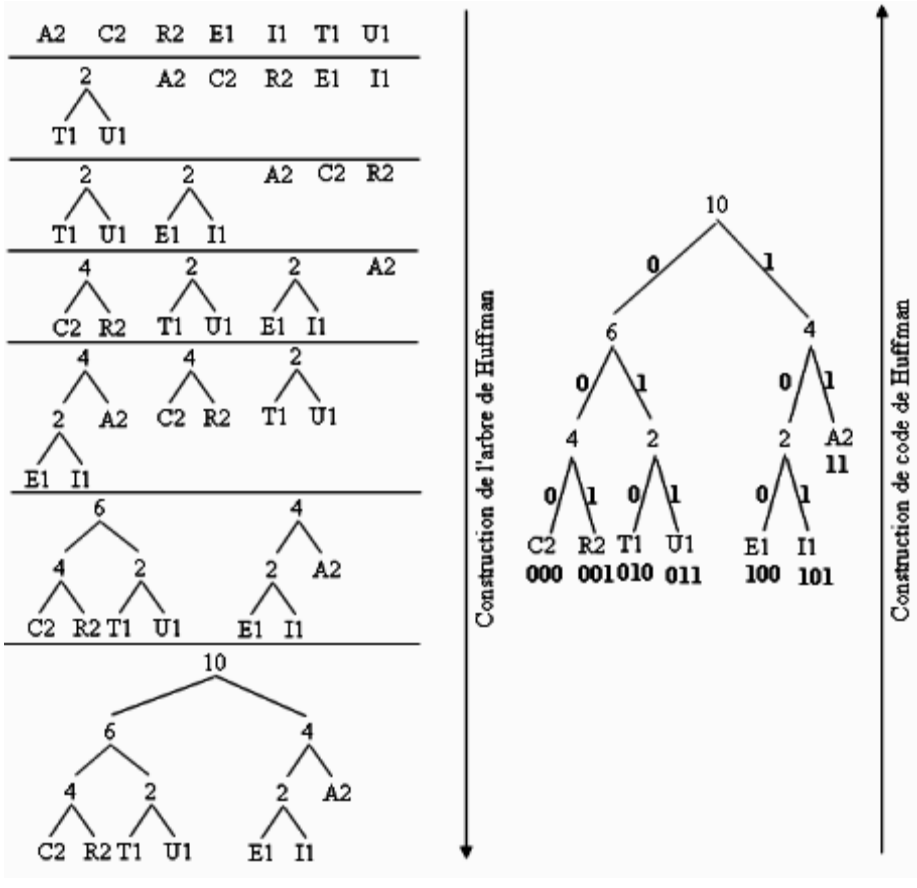


Figure 2.6 Construction de l'arbre et du code de Huffman

La construction du code par l'algorithme de Huffman peut se faire en une seule passe. En effet, on peut construire le code en même temps que l'arbre. A chaque fois qu'un nouvel arbre est créé par fusion de deux sous-arbres, un 0 est ajouté à gauche de toutes les feuilles de l'arbre de gauche et un 1 est ajouté à gauche de toutes les feuilles de l'arbre à droite. Les figures 2.7 et 2.8 illustrent les étapes de construction de code Huffman en une seule passe pour la séquence "COMPRESSIONSANSPERTES".

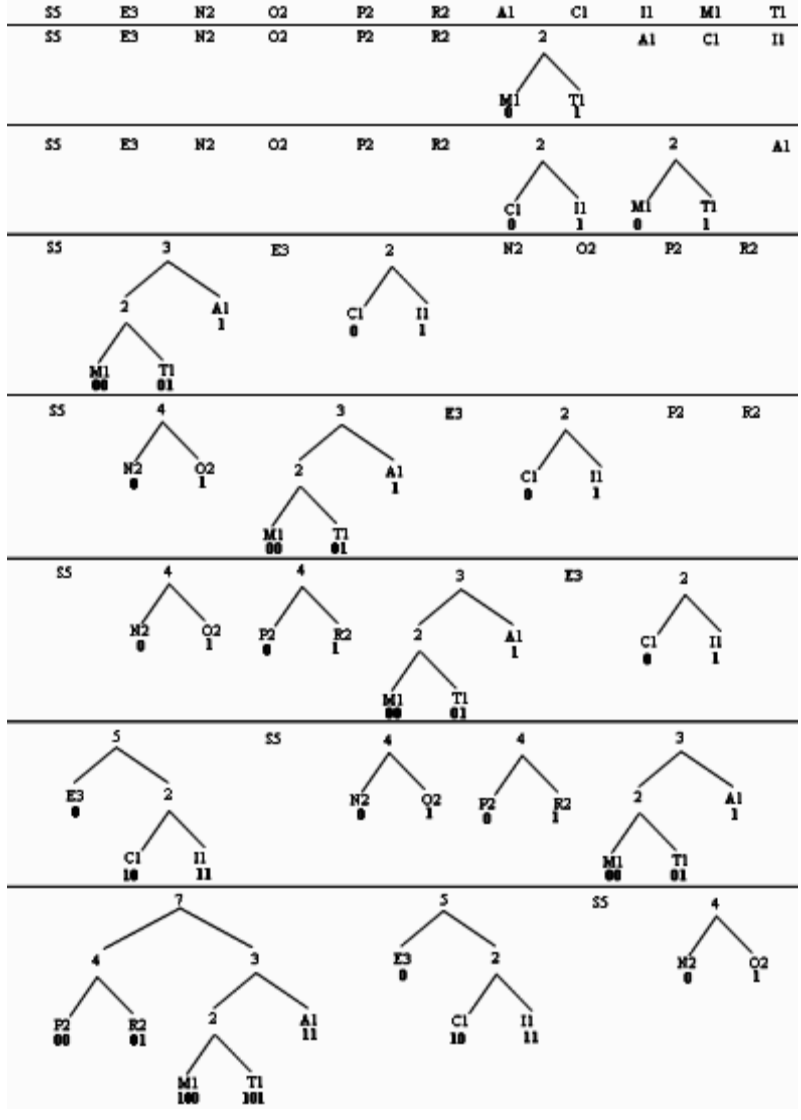


Figure 2.7 Construction de code de Huffman en une seule passe pour la séquence "COMPRESSIONSANSPERTES"

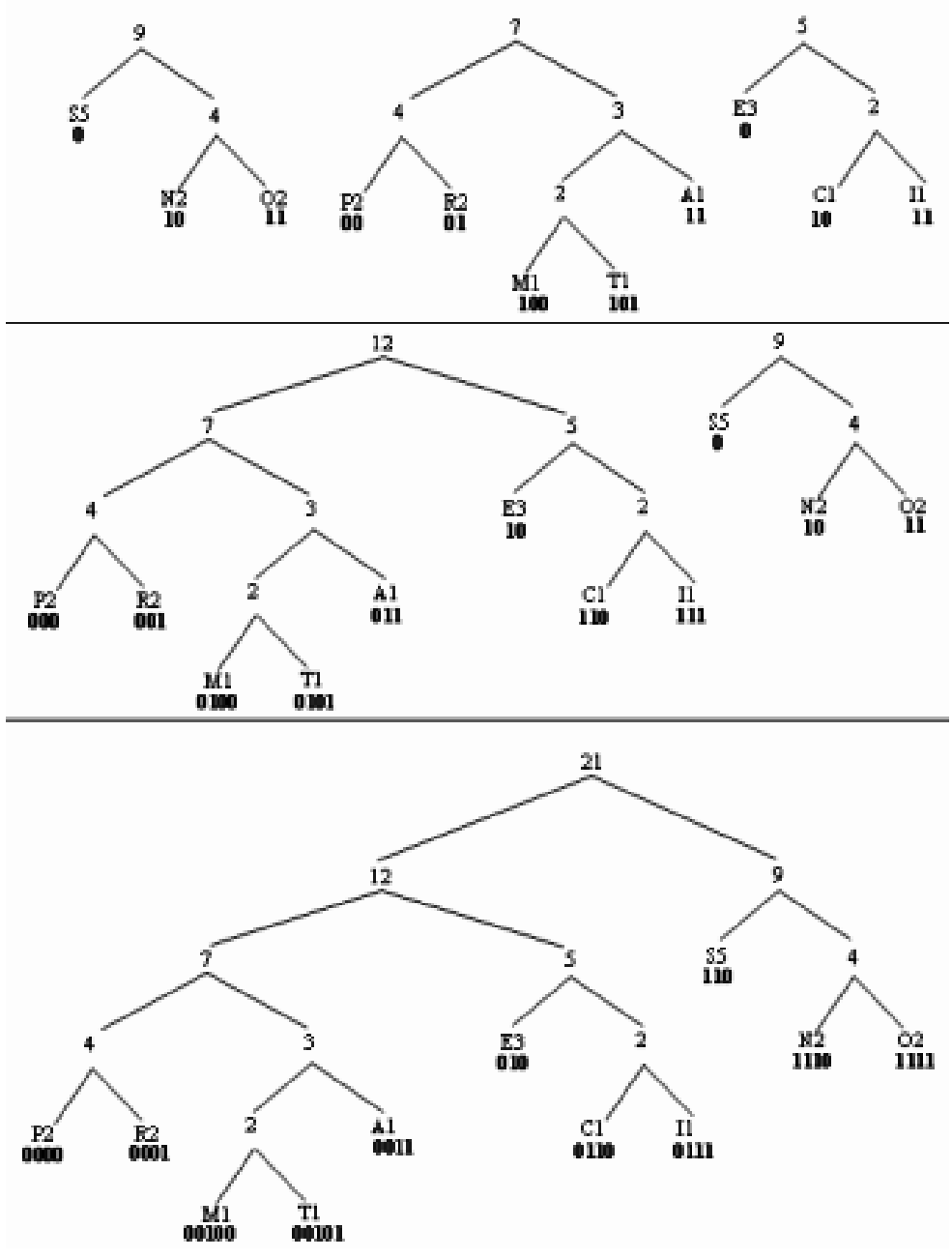


Figure 2.8 Construction de code de Huffman en une seule passe pour la séquence "COMPRESSIONSANSPERTES" – suite

2.4.3 Codage arithmétique

Le codeur arithmétique permet de coder la source entière sous forme d'un seul nombre réel compris entre 0 et 1. La construction du code passe par les étapes suivantes :

1. Parcourir la source pour recueillir la probabilité associée à chaque symbole de la source.

2. Attribuer à chaque symbole un sous-intervalle proportionnel à sa probabilité dans l'intervalle [0,1[le sens de rangement des intervalles est intéressant au décodage [6].
3. La limite inférieure de l'intervalle de travail est initialisée à 0 tandis que la limite supérieure est initialisée à 1.
4. Pour chaque symbole de la séquence :
   - La largeur est égale à la limite supérieure moins la limite inférieure.
   - La limite inférieure est égale à la limite inférieure précédente ajoutée la largeur multipliée par la limite basse du sous-intervalle du symbole.
   - La limite supérieure est égale à la limite inférieure précédente ajourée à la largeur multipliée par la limite haute du sous-intervalle du symbole.
5. La séquence est codée de façon unique par la limite inférieure.

Exemple : Considérons la séquence "CARICATURE" contenant les symboles : {A, C, E, I, R, T, U} nous avons : P(A) = 0.2, P(C) = 0.2, P(E) = 0.1, P(I) = 0.1, P(R) = 0.2, P(T) = 0.1, P(U) = 0.1, les intervalles associés à chaque symbole : I_A =[0,0.2[, I_C =[0.2,0.4[, I_E =[0.4,0.5[, I_I =[0.5,0.6[, I_R =[0.6,0.8[, I_T =[0.8,0.9[, I_U =[0.9,1[. Le tableau 2.4.3 illustre la construction du code arithmétique de la séquence.

Source	Longueur	Borne supérieure	Borne inférieure
C	1	0.4	0.2
A	0.2	0.24	0.2
R	0.04	0.232	0.224
I	0.008	0.2288	0.228
C	0.0008	0.22832	0.22816
A	0.00016	0.228192	0.22816
T	0.000032	0.2281888	0.2281856
U	0.0000032	0.2281632	0.22816288
R	0.00000032	0.228163136	0.228163072
E	0.000000064	0.228163104	0.228163096

Table 2.3 Construction du code arithmétique pour la séquence CARICATURE

Le code réel à transmettre est : **0.2281630976**.

Le décodage arithmétique se fait de manière symétrique, il suffit de connaître les intervalles associés à chaque symbole pour décoder d'une façon unique le code réel reçu. Le principe de décodage est le suivant :

1. On cherche le premier symbole de la séquence qui correspond à l'intervalle qui contient le code réel reçu.
2. Le symbole reçu étant saisi, il va être supprimé du code réel par l'opération inverse : soustraction de la borne inférieure de l'intervalle du symbole reçu puis division par la longueur de cet intervalle.
3. On répète les opérations précédentes jusqu'à ce que le code devienne nul.

2.4.4 Codage à base de dictionnaire

Le codage à base de dictionnaire consiste à coder les symboles par des index pointant sur ceux-ci dans un dictionnaire. Le codage peut être statique ou dynamique.
Dans le cas de codage statique, le dictionnaire est prédéfini et connu par le codeur et le décodeur. Dans le cas de codage dynamique, le dictionnaire est construit au moment de codage.

2.4.4.1 Fenêtre coulissante : LZ77

J. Ziv et A. Lempel [7] ont proposé un algorithme nommé LZ77 à base de dictionnaire qui utilise une fenêtre glissante divisée en deux parties. La première partie à gauche contient le texte codé. La deuxième, à droite, est un tampon contenant les caractères à coder. Si le début du tampon existe déjà dans la première partie, le code à transmettre est composé de la position du bloc dans la première partie, la longueur de ce bloc et le premier symbole différent. Le bloc plus le premier symbole différent sont glissés vers la première partie. Si le premier symbole du tampon ne figure pas dans la première partie, le code à transmettre devient 0,0 suivi du symbole. La figure 2.9 montre un exemple de compression d'une séquence d'ADN en utilisant l'algorithme LZ77.

2.4.4.2 Construction dynamique de dictionnaire : LZ78

LZ78 est une extension de LZ77 proposée par les mêmes auteurs [8] pour surmonter le problème de la mémoire courte : la fenêtre coulissante est d'une taille fixe, dès que les symboles sortent de cette fenêtre, ils ne sont plus pris en considération. L'amélioration qui a été proposée consiste à remplacer le tampon de recherche par un dictionnaire dynamique. Ce dictionnaire va contenir tous les symboles et les blocs répétés dans le message et il sera

construit dynamiquement par le codeur et le décodeur sans avoir besoin d'être transmis, il est alors détruit à la fin du traitement.

Chaque symbole transmis prend un index dans le dictionnaire, si le symbole se trouve déjà dans le dictionnaire le codeur transmet l'index du symbole reconnu suivi du symbole suivant. Si le symbole apparaît pour la première fois, le codeur envoi 0 suivi du symbole lui-même. Dans le premier cas, le symbole reconnu et le symbole suivant constituent un bloc qui sera lui aussi stocké dans le dictionnaire.

De la même manière que dans LZ77 si tout un bloc existe déjà dans le dictionnaire, le codeur transmet son index suivi du premier symbole non reconnu. Le décodage se fait de manière symétrique. Les tableaux 2.4 et 2.5 donnent les étapes de codage et décodage de la chaîne d'ADN : "ACAGCAGTCGAGCTCCAGCTCCAGCTCGA".

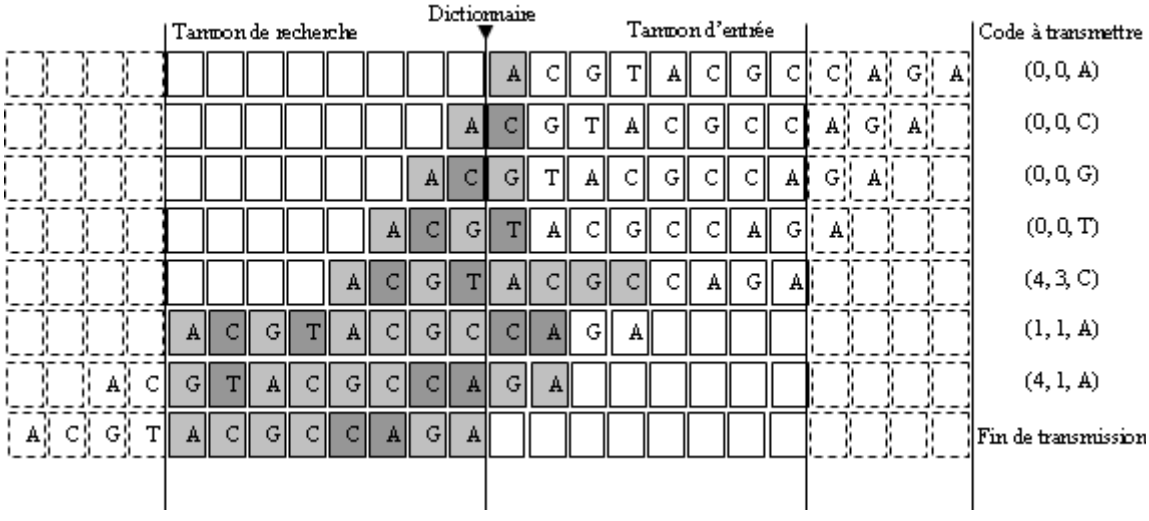


Figure 2.9 Exemple de codage d'une séquence d'ADN avec LZ77

Séquence à coder		Dictionnaire		Code à transmettre
Symbole ou bloque reconnu	Symbole suivant	Index	Symbole ou bloque	
nom	A	1	A	(0,A)
nom	C	2	C	(0,C)
A	G	3	AG	(1,G)
C	A	4	CA	(2,A)
nom	G	5	G	(0,G)
nom	T	6	T	(0,T)
C	G	7	CG	(2,G)
AG	C	8	AGC	(3,C)
T	C	9	TC	(6,C)
CA	G	10	CAG	(4,G)
C	T	11	CT	(2,T)
C	C	12	CC	(2,C)
AGC	T	13	AGCT	(8,T)
CG	A	14	CGA	(7,A)

Table 2.4 Codage LZ78

	Identification		Dictionnaire	
codes reçu	symbole ou bloque reconnu	Symbole suivant	Index	Symbole ou bloque reçu
(0,A)	nom	A	1	A
(0,C)	nom	C	2	C
(1,G	A	G	3	AG
(2,A)	C	A	4	CA
(0,G)	nom	G	5	G
(0,T)	nom	T	6	T
(2,G)	C	G	7	CG
(3,C)	AG	C	8	AGC
(6,C)	T	C	9	TC
(4,G)	CA	G	10	CAG
(2,T)	C	T	11	CT
(2,C)	C	C	12	CC
(8,T)	AGC	T	13	AGCT
(7,A)	CG	A	14	CGA

Table 2.5 Décodage LZ78

2.4.4.3 Extension de LZ78 : LZW

Terry Welch [9] a proposé une amélioration de l'algorithme LZ78 en codant un symbole ou bloc de symboles par un simple index sans transmettre le symbole suivant. Initialement, un dictionnaire contenant tous les symboles du message est construit. A la réception de chaque symbole on concatène celui-ci avec les précédents pour former un bloc qui va être recherché dans le dictionnaire. Si le bloc est trouvé, le codeur transmet son index,

sinon le bloc est ajouté à la première position disponible dans le dictionnaire. Le décodage se fait de manière symétrique.

Le tableau 2.6 montre les étapes de codage de la chaîne d'ADN : "ACAGCAGTCGAGCTC-CAGCTCGA".

Message reçu		Dictionnaire		
Symbole ou bloque courant	Symbole reçu	Symbole ou bloque	index	Code à transmettre
vide	vide	A	1	
vide	vide	C	2	
vide	vide	G	3	
vide	vide	T	4	
vide	A			
A	C	AC	5	1
C	A	CA	6	2
A	G	AG	7	1
G	C	GC	8	3
C	A			
CA	G	CAG	9	6
G	T	GT	10	3
T	C	TC	11	4
C	G	CG	12	2
G	A	GA	13	3
A	G			
AG	C	AGC	14	7
C	T	CT	15	2
T	C			
TC	C	TCC	16	11
C	A			
CA	G			
CAG	C	CAGC	17	9
C	T			
CT	C	CTC	18	15
C	C	CC	19	2
C	A			
CA	G			
CAG	C			
CAGC	T	CAGCT	20	17
T	C			
TC	C			
TCC	A	TCCA	21	16
A	G			
AG	C			
AGC	T	AGCT	22	14
T	C			
TC	G	YCG	23	11
G	vide			3

Table 2.6 Codage LZW

2.4.5 Compression sans pertes d'image

2.4.5.1 Application du RLE à la décomposition en plans de bits

L'image numérique peut être décomposée en plans de bits. Sur chaque plan p on aura une image binaire contenant les valeurs 0 et 1 avec une forte probabilité d'avoir des séquences uniformes assez longues. Une solution très intéressante consiste donc à appliquer un codage RLE sur une image numérique décomposée en plans de bits :

Plan 7	1	1	1	1	1	1	1	0	0	2 plages <1, 7><0, 2>
Plan 6	0	0	0	0	0	0	0	0	0	1 plage <0, 9>
Plan 5	0	0	0	0	0	1	1	1	1	2 plages <0, 5><1, 4>
Plan 4	1	1	1	1	1	1	1	1	1	1 plage <1, 9>
Plan 3	0	1	1	1	0	0	0	1	1	4 plages <0, 1><1, 3><0, 3><1, 2>
Plan 2	0	0	0	0	1	1	1	1	1	2 plages <0, 4><1, 5>
Plan 1	1	1	1	0	0	0	0	0	0	2 plages<1, 3><0, 6>
Plan 0	1	0	1	0	1	0	1	0	1	9 plages

Table 2.7 Décomposition en plan de bits

2.4.5.2 Codage prédictif sans pertes

Le codage prédictif exploite l'existence des corrélations entre les échantillons voisins d'une source. La prédiction d'un échantillon n_i se fait à partir de la connaissance des échantillons précédents : n_{i-1}, n_{i-2},... Dans le cas de la prédiction linéaire, l'échantillon n_i est prédit à l'aide de la relation : $\hat{n}(i) = a_1.n\ (i\text{-}1) + a_2.n\ (i\text{-}2) + n(i)\ a_3.n(i - 3) + ::: + a_m.n(i - m)$. En général, la valeur de n est différente de $\hat{n}$ le codeur transmet la différence entre n et $\hat{n}$ qui est très faible et par conséquent va prendre un nombre réduit de bits par rapport à n.

Le prédicteur DPCM (Differential Pulse Code Modulation) utilise un contexte d'ordre 4 au maximum (figure 2.10) et les prédicteurs standard suivants [1] :

$$\hat{n}(i,j) = A\ ;\ \hat{n}(i,j) = \frac{A+B}{2}\ ;\ \hat{n}(i,j) = \frac{A+C}{2}\ ;\ \hat{n}(i,j) = \frac{2A+C+D}{4}\ ;$$

$$\hat{n}(i,j) = A - B + C\ ;\ \hat{n}(i,j) = \frac{2A+D-B}{2}\ ;\ \hat{n}(i,j) = \frac{3(A+C)-2B}{4}$$

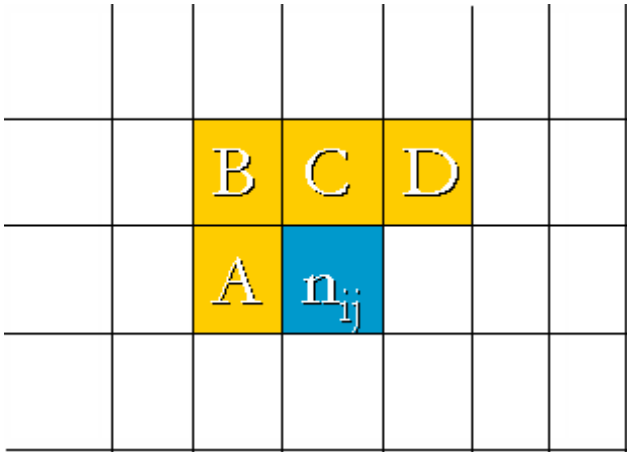


Figure 2.10 Voisinage de DPCM

2.5 Compression avec pertes d'image

2.5.1 Quantification d'image

2.5.1.1 Quantification scalaire uniforme

La quantification est une technique de distorsion qui consiste à réduire l'alphabet d'une source. Cette réduction est effectuée en substituant les valeurs du signal s par des seuils r_k si s appartient à l'intervalle $I_k = [a_k, a_{k+1}[$ associé au seuil r_k (figure 2.11). Une telle transformation vise à minimiser la distorsion entre l'entrée et la sortie [6]. Dans le cas d'image en couleurs, on cherche à utiliser un ensemble présélectionné de couleurs représentatives [10, 2]. Les s_i représentent dans ce cas les valeurs d'intensités des pixels.

2.5.1.2 Quantification de Lloyd-Max

Lloyd [11] et Max [12] ont proposé une méthode de quantification scalaire non uniforme qui permet de minimiser l'erreur quadratique moyenne entre l'entrée et la sortie pour un espace des valeurs quantifiées de dimension n. L'expression qui minimise cette erreur est la suivante :

$$E\left[(s-r)^2\right] = \int_{a1}^{an+1} (s-r)^2 p(s)\,ds \qquad (2.3)$$

Ce qui conduit aux valeurs suivantes :

$$a_k = \frac{r_k + r_{k-1}}{2} \qquad (2.4)$$

$$r_k = \frac{\int_{ak}^{ak+1} s.P_s(s)ds}{\int_{ak}^{ak+1} P_s(s)ds} \qquad (2.5)$$

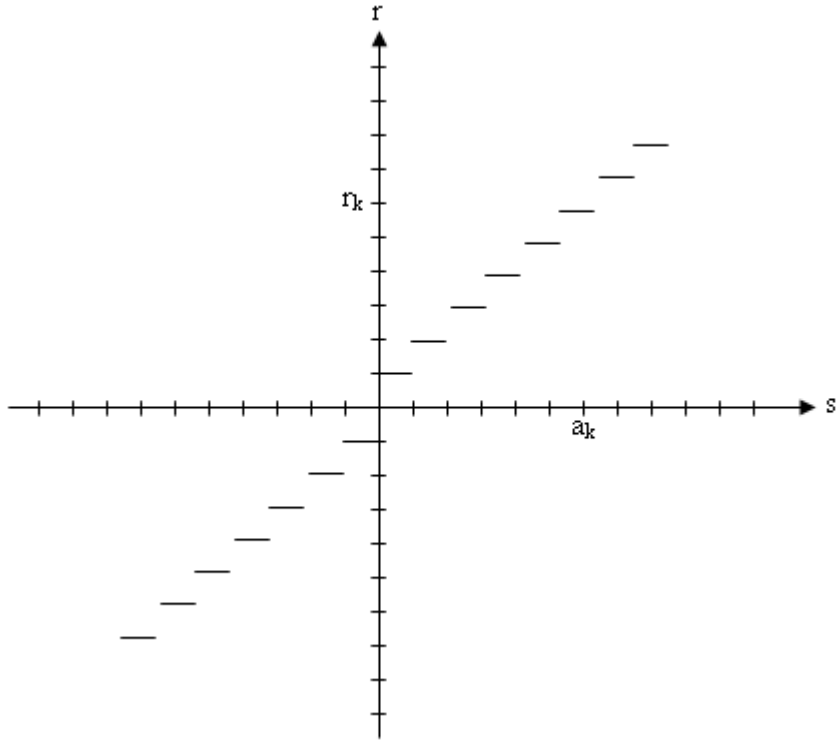


Figure 2.11 Quantification scalaire uniforme

où $p_s(s)$ est la densité de probabilité continue de la variable s.

2.5.1.3 Quantification vectorielle

La quantification vectorielle est une transformation qui permet de réduire les symboles d'une source en associant à chaque vecteur (pixel à trois composantes (RGB) par exemple) $X_i = (xj,\ j = 1...k)$ un vecteur $Y_i = (y_j,\ j = 1\ldots k)$ choisi d'un dictionnaire (ou Codebook) contenant des index ou codes $C = (\hat{X}_n, n = 1\ldots N_c)$. La décision d'associer un tel vecteur est prise par calcul d'un critère de ressemblance (exemple : distance euclidienne). La construction du dictionnaire peut être réalisée en appliquant une classification des vecteurs d'entrée en se basant sur un critère de ressemblance. Il existe un ensemble de méthodes parmi lesquelles on trouve l'algorithme de Lunde-Buzo-Gray [13] qui est en quelque sorte une généralisation de l'algorithme de Lloyd-Max. Malgré l'efficacité de la quantification vectorielle, celle-ci n'est pas encore intégrée par les normes de compression. Elle est utilisée surtout pour extraire une palette de teintes dans le cas d'une image couleur indexées. La quantification scalaire est très

utilisée par les normes de compression car elle est facile à implémenter et ne demande pas beaucoup de temps de calcul.

2.5.2 Transformée de Fourier discrète : DFT

La transformée de Fourier discrète permet de transformer une image *I(i; j)* du domaine spatial au domaine fréquentiel *DFTI*(*u*, *v*) selon la relation suivante :

$$DFTI(u,v) = \frac{1}{\sqrt{M.N}} \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} I(i,j) e^{-2p\left(\frac{ui}{N} + \frac{vi}{M}\right)} \qquad (2.6)$$

La DFT présente l'avantage d'être séparable : elle peut être calculée sur chaque ligne de l'image de façon indépendante. En plus, la version rapide de la DFT : FFT permet de baisser considérablement le temps de calcul. Cependant, les coefficients générés par la DFT sont complexes ce qui entraîne l'obtention d'un nombre considérable de coefficients redondants (presque la moitié).

2.5.3 Transformée en cosinus discrète : DCT

La DCT est une des transformations les plus utilisées dans la compression d'image. Elle ressemble à la DFT sauf qu'elle utilise seulement les valeurs réelles. Elle a été intégrée dans plusieurs normes de codage d'image fixe et animée de référence telles que JPEG et MPEG. La transformation DCT d'un bloc carré *I(i; j)* de taille N d'une image est obtenue par la formule suivante :

$$DCTI(u,v) = \frac{2}{N} . C(u) C(v) \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} I(i,j) \cos\left(\frac{(2i+1)pu}{2N}\right) \cos\left(\frac{(2j+1)pv}{2N}\right) \qquad (2.7)$$

$$C(x) = \begin{cases} \frac{1}{\sqrt{2}} si\ x = 0 \\ 1 si\ x \neq 0 \end{cases} \qquad (2.8)$$

La DCT regroupe les basses fréquences dans le triangle supérieur gauche et les hautes fréquences dans le triangle inférieur droit. Le système visuel humain est moins sensible vis-à-vis des hautes fréquences. Cette propriété est exploitée pour réduire les coefficients correspondant aux hautes fréquences en les éliminant par quantification. Ceci va augmenter de façon significative le gain en nombre de bits.

2.5.4 Transformée en ondelettes discrète : DWT

La transformée en ondelettes est une représentation du signal dans le domaine fréquentiel et temporel. Elle a été développée pour résoudre les limitations de la transformée de Fourier. Au moment que la transformée de Fourier donne une résolution constante pour toutes les fréquences, la transformée en ondelettes utilise une technique multi-résolution qui permet d'analyser plusieurs fréquences avec plusieurs résolutions. L'analyse par ondelettes est effectuée de la même façon que celle de Fourier. Le signal est multiplié par une fonction d'ondelettes comme s'il est multiplié par une fenêtre de transformation de Fourier. Cependant, la largeur de la fonction d'ondelettes peut changer pour chaque composante spectrale.

Les bases de la DWT ont été introduites dés 1976 avec l'apparition des techniques de décomposition en sous-bandes des signaux discrets dans un contexte de compression du signal de parole [14]. Ces techniques ont connu plusieurs améliorations jusqu'à l'établissement du schéma d'analyse multirésolution par Mallat [15]. La DWT d'une image est obtenue par une série de filtrages successifs passe-haut H et passe-bas G appliqués sur les lignes puis sur les colonnes comme le montre la figure 2.12.

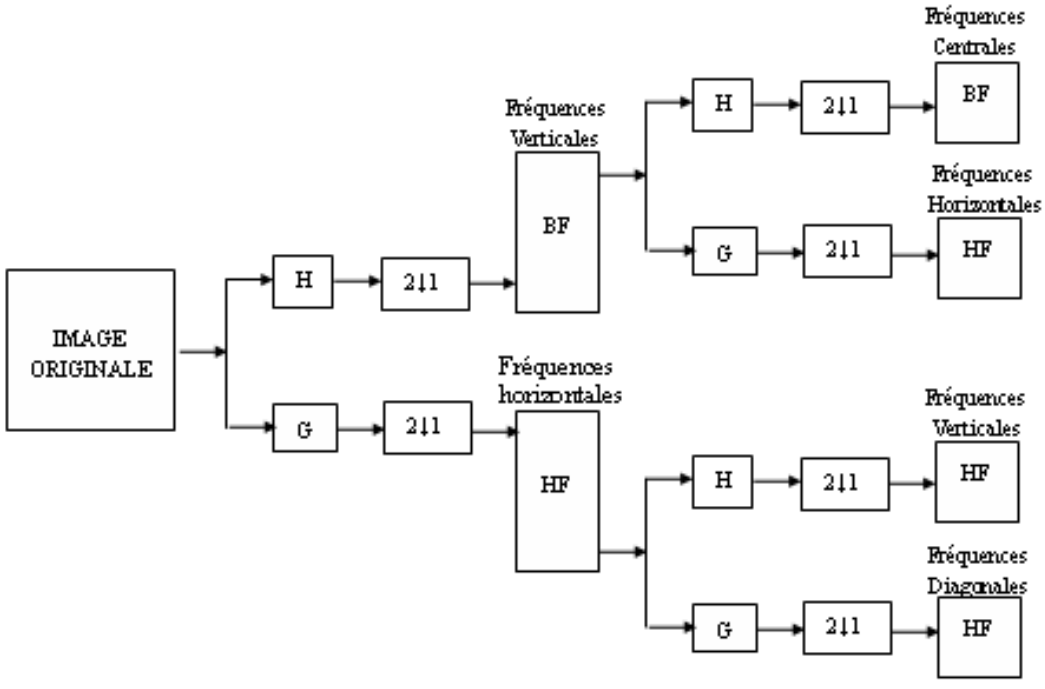


Figure 2.12 Schéma de décomposition d'une image en sous-bandes

Cette décomposition en sous-bandes fournit à chaque échelle 4 sous-images :

- Image de faible résolution.
- Image de détails horizontaux.
- Image de détails verticaux.
- Image de détails diagonaux.

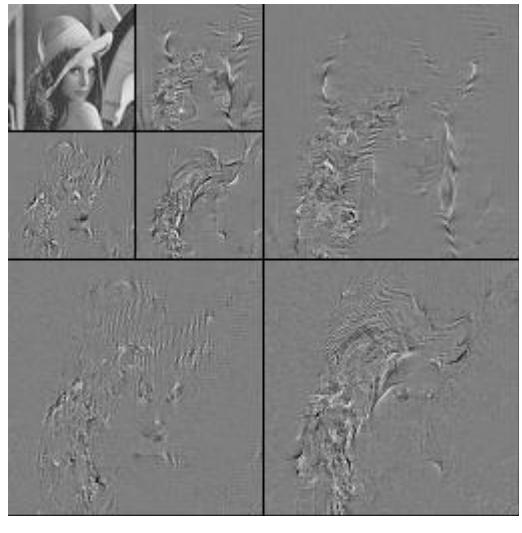


Figure 2.13 DWT de l'image "Lena"

L'application de la DWT à la compression a montré son efficacité. En effet, Antonini et al. [6, 17] ont proposé une technique de compression qui prend en considération les caractéristiques psychovisuelles dans les domaines spatial et fréquentiel. L'image est décomposée en ondelettes. Les coefficients qui en résultent subissent une quantification vectorielle. L'avantage de la compression par DWT est la possibilité de décodage progressif grâce à la représentation multirésolution de l'image.

2.6 Normes de compression d'image fixe

2.6.1 Norme JPEG

Depuis les années 80, les membres de l'ITU et l'ISO ont travaillé ensemble pour établir une norme internationale pour la compression des images fixes à niveau de gris et en couleurs. Ce travail est connu sous le nom de JPEG "Joint Photographic Experts Group", le mot "Joint" fait référence à la collaboration entre l'ITU et l'ISO. Officiellement JPEG correspond à la norme internationale ISO/IEC 10928-1, codage et compression numérique d'images fixes ou à la recommandation ITU-TT.81. Le contenu des deux documents est identique. Le processus de l'établissement de la norme a été réalisé de la façon suivante : après l'évaluation de plusieurs schémas de codage, les membres de JPEG ont choisi la méthode basée sur la DCT en 1988. Du 1988 au 1990, le groupe JPEG a continué son travail de simulation, test et documentation de l'algorithme. JPEG devient ainsi une norme internationale "Draft International Standard" DIS en 1991 et "International Standard" IS en 1992 [18, 19, 20].

Le schéma bloc du codeur/décodeur JPEG est illustré dans la figure 2.14. L'image est tout d'abord découpée en blocs de 8x8 pixels. Chaque bloc va subir le processus de codage de façon indépendante. Les blocs passent par une transformation en cosinus discrète. Les coefficients issus de la DCT subissent une quantification suivie d'un codage entropique. Le processus de décodage est symétrique à celui du codage.

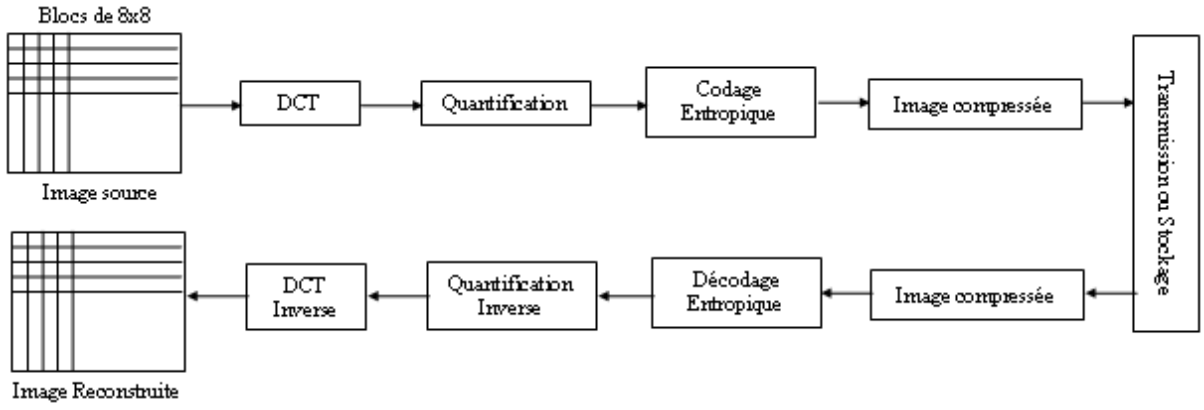


Figure 2.14 Diagramme bloc du codeur JPEG

2.6.1.1 Préparation de l'image

Les images couleurs passent tout d'abord par une transformation de composantes de l'espace RGB vers l'espace de luminance et chrominance. Cette transformation vise à réduire le nombre de bits qui codent un pixel de 24 bpp à 16 bpp. Sachant que le système visuel humain est plus sensible à la luminance qu'à la chrominance, la norme JPEG attribue à chaque bloc de quatre pixels quatre luminance et une seule chrominance partagée par les quatre. Les composantes sont découpées ensuite en blocs de 8X8 pixels (figure 2.15). Les échantillons sont ensuite décalés des entiers non-signés de l'intervalle $[0, 2^p -1[$ vers les entiers signés de l'intervalle $[-2^{P-1}, 2^{P-1} - 1[$.

2.6.1.2 Transformée en cosinus discrète

Chaque bloc des composantes passe par une transformation en cosinus discrète DCT. Le bloc transformé contient 64 coefficients. Le coefficient de fréquence nulle est appelé la composante continue DC. Les 63 coefficients restants sont appelés composantes AC. Puisque la variation d'intensité entre pixels voisins est très lente, les informations sont concentrées en basses fréquences. La majorité des coefficients du bloc sont nuls ou presque nuls après transformation DCT et ne nécessitent pas d'être codés.

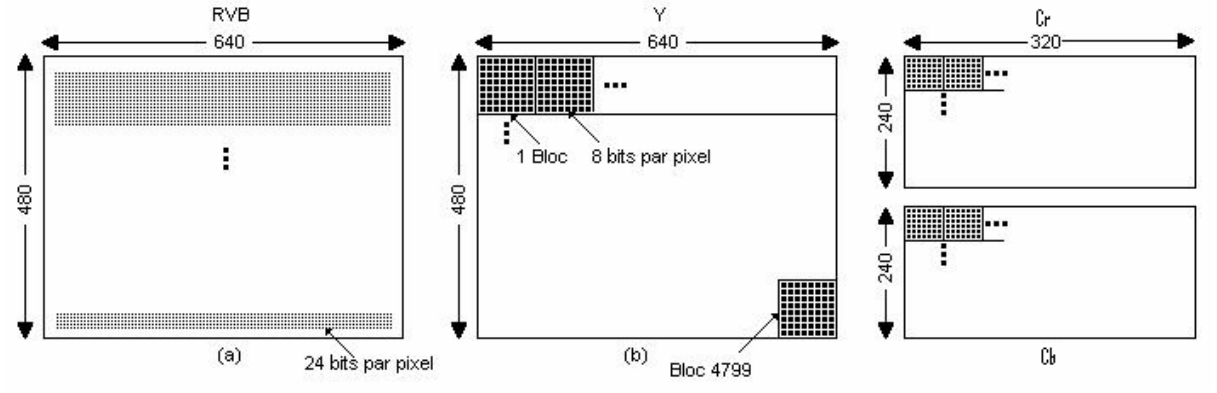


Figure 2.15 Découpage en blocs de 8X8

2.6.1.3 Quantification

Après la DCT, les 64 coefficients sont quantifiés selon une table de quantification spécifiée par l'application ou fournie comme entrée par l'utilisateur. La table de quantification contient 64 coefficients compris entre 1 et 255. La quantification consiste à diviser chaque coefficient par le facteur de quantification correspondant (qui se trouve dans la même position), le résultat de la division est arrondi à l'entier proche. Les facteurs de la table de quantification sont choisis de façon à permettre de conserver les coefficients correspondants aux basses fréquences et réduire ceux correspondants aux hautes fréquences. En tenant compte du fait que le système visuel humain est moins sensible aux hautes fréquences.

2.6.1.4 Codage des DC et parcours en ZigZag

Après la quantification, les coefficients DC sont traités séparément à ceux des AC. Le coefficient DC mesure la valeur moyenne des 63 coefficients AC restants. Puisqu'il y a une forte corrélation entre les DC des blocs adjacents, les DC sont codés par leurs différences avec les DC des blocs précédents dans l'ordre de codage. Ensuite les coefficients sont balayés en Zigzag pour augmenter la chance d'avoir des zéros successifs. Les séquences obtenues sont codés en RLE.

2.6.1.5 Codage entropique

La dernière étape du codeur JPEG est le codage entropique. Cette étape applique une compression additionnelle sans pertes sur les coefficients issus de l'étape précédente. JPEG a spécifié deux codeurs entropiques : le codeur de Huffman et le codeur arithmétique.

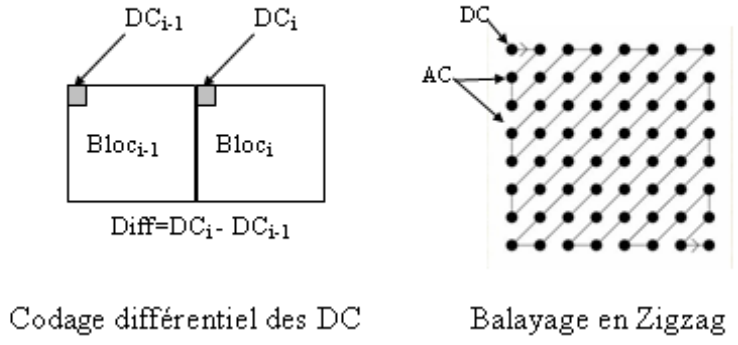


Figure 2.16 Quantification des coefficients DCT

2.6.2 Norme JPEG 2000

En mars 1997, un nouvel appel à contribution a été lancé pour le développement d'une nouvelle norme de compression d'images fixes baptisée JPEG 2000 [21]. Ce projet, référencié sous JTC 1.29.14 (15444), a été destiné pour créer un nouveau système de codage pour différents types d'images fixes (couleur, niveaux de gris, multi-composantes), dans différents domaines (images naturelles, scientifiques, médicales, texte, etc...) permettant différentes applications (client/serveur, transmission temps réel, archivage de librairie d'image, largeur de bande et espace de stockage limité, etc...) de préférence dans un système unifié. Ce système de codage doit fournir des performances en termes de taux de distorsion et de qualité d'image supérieures à celles des normes existantes à bas débit en intégrant en même temps d'autres caractéristiques intéressantes. La norme est destinée à complémenter JPEG et pas à la remplacer.

Le diagramme bloc du codeur JPEG 2000 est illustré dans la figure 2.17. La transformation en ondelettes discrète est appliquée en premier sur les composantes de l'image. Les coefficients transformés subissent une quantification suivie d'un codage entropique. Ainsi, le train binaire de sortie est formé. Le processus de décodage est symétrique à celui du codag. Le train binaire passe tout d'abord par le décodage entropique, ensuite la quantification inverse suivie de la transformation en ondelettes discrète inverse. Ainsi, l'image est reconstruite.

2.6.2.1 Prétraitement de l'image

Avant de commencer le processus de codage, l'image passe par un prétraitement qui consiste à transformer les composantes de l'image de l'espace RGB à l'espace YCbCr irréversible ou YrUrVr réversible recommandés par JPEG2000, ensuite les composantes de l'image sont découpées en un ensemble de blocs rectangulaires nonsuperposés (tiles). Le processus de codage est appliqué à ces blocs indépendamment comme des images totalement distinctes. Juste avant la DWT, les échantillons sont décalés en niveau continu : DC level shifting [21]. Ce décalage est appliqué sur les valeurs non-signées seulement pour les représenter en complément à 2.

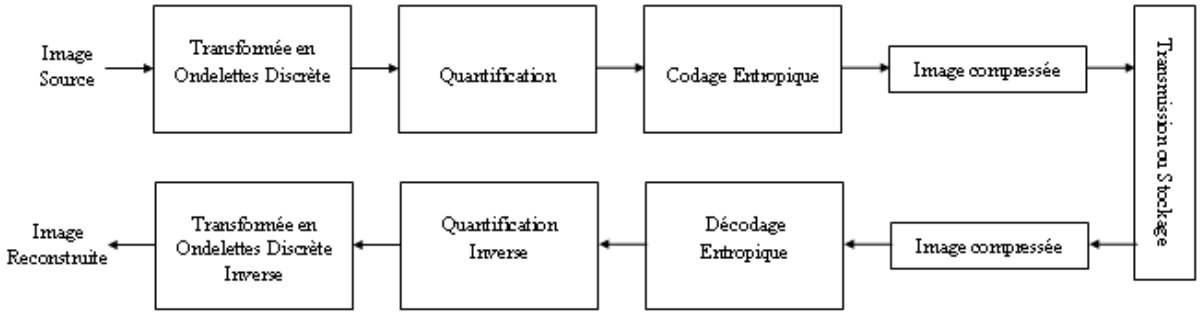


Figure 2.17 Diagramme bloc du codeur JPEG2000

2.6.2.2 La transformée en ondelettes discrète

Les tiles de composantes sont décomposées en plusieurs niveaux en utilisant la DWT. Ces niveaux de décomposition contiennent un certain nombre de sous-bandes. La DWT peut être réversible ou irréversible. La DWT irréversible est implémentée par défaut à l'aide du filtre de Daubechies 9/7 [16]. La DWT réversible est implémentée à l'aide d'un filtre de 5/3.

2.6.2.3 La quantification

La norme JPEG2000 utilise une quantification scalaire qui réduit la précision des coefficients. Cette opération entraîne des pertes sur les données, sauf si le pas de quantification est égal à 1 et les coefficients sont des entiers issus d'un filtre de DWT réversible 5/3.

2.6.2.4 Le codage entropique

Les plans de bits des coefficients des "code-blocks" passent par un codage entropique. Le codeur entropique adopté par JPEG 2000 est le codeur arithmétique binaire dépendant du contexte (QM-Coder). Ce codage est similaire à celui adopté dans la norme JPEG originale [20].

2.6.2.5 Formation du train binaire

Le train binaire contient des marqueurs qui ont pour objectif la protection contre les erreurs de transmission. L'en-tête principal est situé au début du train binaire, il décrit l'image originale, les différentes décompositions et les styles de codage qui sont utilisés pour localiser, extraire, décoder et reconstruire l'image avec la résolution et la fidélité désirées. Le format de fichier optionnel décrit la signification de l'image et ses composantes dans le contexte de l'application.

2.7 Conclusion

Nous avons présenté dans ce chapitre les différentes techniques de compression d'image qui existent dans la littérature. Nous avons commencé par introduire des notions de la théorie de l'information sur laquelle se base l'ensemble des méthodes de compression des données. Nous avons défini les notions de source, de quantité d'information, d'entropie et du codage source - codage canal. Nous avons ensuite donné un aperçu sur les formats bruts d'image qui existent. Un parcourt des méthodes réversibles de compression de données a été présenté introduisant le codage par longueur de séquence, le codage à longueur variable, le codage arithmétique, le codage à base de dictionnaire et la compression d'image sans pertes. Nous avons ensuite abordé en particulier les techniques de compression avec pertes d'image, à savoir, les techniques de quantification, les transformations : DCT, DWT et quelques normes de compression d'image fixe : JPEG, JPEG2000.

Chapitre 3

Compression par régions d'intérêt des images fixes

Dans ce chapitre, nous allons nous intéresser au principe de l'approche de compression adaptative (ou sélective) d'image fixe. Vue que dans la plupart des domaines de traitement d'image, l'utilisateur ne s'intéresse qu'à une partie de l'information transmise à travers l'image. Il n'est pas nécessaire de représenter l'image entière en pleine résolution. Ceci suggère d'appliquer une compression qui consiste à coder l'image de manière à conserver les régions qualifiées d'intérêt par l'utilisateur, et dégrader de façon contrôlable l'arrière plan (le reste de l'image), c'est ce que nous appelons la compression sélective. L'objectif de la compression sélective est de réduire la taille du fichier en conservant l'information intéressante transmise à travers l'image. La méthode proposée consiste à séparer le traitement entre les régions d'intérêt (ROI pour Region Of Interest) et l'arrière plan (BG pour Background). Les ROIs, extraites de manière manuelle, sont compressées à l'aide d'une technique réversible (SPIHT), alors que le BG subit une forte compression avec des techniques irréversibles (JPEG2000 par exemple).

Dans ce qui suit, nous présentons une revue de littérature sur les méthodes de compression sélective existantes, puis nous expliquons le principe de base de l'algorithme SPIHT, et enfin nous présentons la méthode proposée ainsi que les résultats obtenus.

2.1 Aperçu des méthodes existantes

La compression sélective consiste à coder à l'aide de différentes méthodes les zones de la scène caractérisées par degré d'importance. Durant la dernière décennie, la compression adaptative a fait l'objet d'un certain nombre de travaux. En particulier, Guisto et al. en 1990 [22] ont proposé une compression qui permet de localiser d'abord les zones d'intérêt, sur lesquelles, ils ont appliqué des techniques comme la quantification vectorielle ou l'approximation polynomiale sans se soucier des pertes dans ces zones. Nguyen en 1995 [23], proposa une méthode originale de compression sélective de séquences d'images pour la transmission à très bas débit. Cette méthode repose sur la notion de niveau d'intérêt affecté à chaque région de la scène dans laquelle le codeur ayant une structure hybride utilise une représentation sous-bande globale. L'auteur porte une attention particulière au codage mais sa méthode ne permet pas de contrôler la résolution du contexte filtré. Pour pallier ce problème Benharrosh proposa, en 1998 [24], une approche adaptative basée sur des techniques classiques de compression réversibles associées à des techniques d'analyse multi-résolution afin de pouvoir transmettre de façon optimale sur un réseau à débit limité une image contenant des thèmes préservés à haute résolution dans un contexte à basse résolution, en évitant toute perte sur les zones d'intérêt. Dans ce travail, les zones d'intérêt sont détectées manuellement. La détection automatique des zones d'intérêt a fait l'objet de travaux divers tel que celui de Hodschneider qui utilisa la Transformation en ondelettes à trous (T.O.T) [24].

3.2 Concept de la compression adaptative

Le concept de la compression adaptative proposé s'inscrit directement dans le processus d'échange de données. Ce processus d'échanges peut être représenté fonctionnellement par la figure ci-dessous. Il consiste en :

– Une analyse préliminaire des données, par l'émetteur, afin de déterminer celles qui sont pertinentes : les zones qualifiées d'intérêt,

– Une sélection manuelle ou semi - automatique des zones d'intérêt,

– Une compression adaptative des donnés à transmettre ou à stocker de manière à ne pas trop dégrader les zones d'intérêt.

On retrouve ainsi l'idée même du concept de généralisation cartographique dans laquelle on sélectionne les objets pertinents avant de les modifier et ce dans le but de les conserver après changement d'échelle [25].

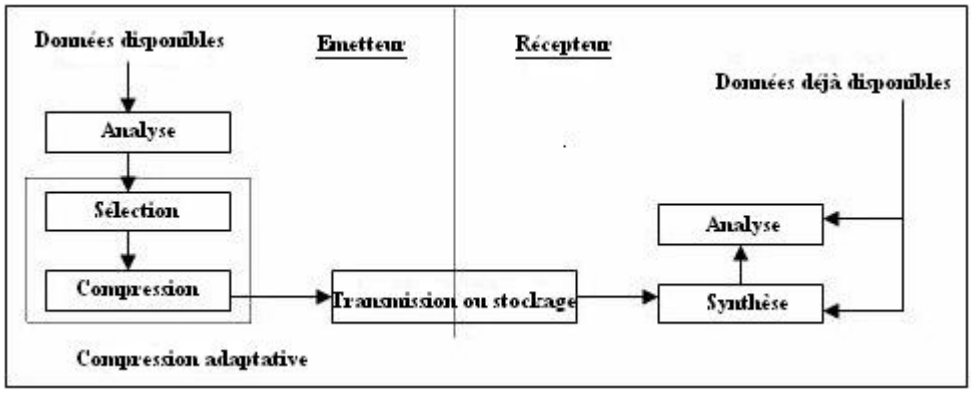


Figure 3.1 Processus d'échange des données entre deux niveaux.

Après la transmission ou le stockage, il y'a synthèse ou reconstruction des données et analyse des données. La synthèse consiste à reconstruire les données transmises ou stockées, qui est une étape correspondant à la décompression des données [26]. L'analyse consiste à examiner les données transmises (zones d'intérêt dans un contexte dégradé).

3.3 Compression des zones d'intérêt

3.3.1 Extraction des zones d'intérêt

Le but, est de présenter une méthode permettant de générer une image basse résolution préservant une zone qualifiée d'intérêt par l'utilisateur. Pour cela, une étape préalable consiste à la détecter puis l'extraire. Les algorithmes d'extraction ne sont pas encore vraiment fiables et nécessitent d'être implémentés et testés. D'autre part les zones d'intérêt retenues dans une image peuvent être variées ce qui nécessite d'étudier et de mettre en œuvre un divers algorithmes d'extraction. On préfère aussi se placer dans un contexte supervisé (interactif). Dans ce contexte, l'opérateur peut définir lui-même les pixels qu'il considère comme appartenant à des zones qualifiées d'intérêt. Le principal avantage offert par cette approche est que certains pixels peuvent appartenir simultanément à plusieurs thèmes d'intérêt. L'opérateur peut également définir simplement des " Zones génériques " qu'il qualifie d'intérêt. Ces zones peuvent être de différentes natures : bandes, carrés, ronds,....Pour ces raisons, nous n'avons pas cherché à étudier des algorithmes d'extraction existants dans la littérature. Des méthodes manuelles sont utilisées. Il s'agit de saisir directement sur l'image, les objets qui intéressent l'utilisateur à l'aide de la souris (l'image est une image binaire (zone d'intérêt et contexte)[27]. Ces objets peuvent être de différents types :

– Les objets linéiques (exemple en médecine : cellules anormales enchaînées,...)
– Les objets surfaciques (exemple en médecine : ensemble de cellules infectées,...)
– Les objets ponctuels (exemple en médecine : une cellule infectée,...)

Dans le cas d'une saisie manuelle d'objets linéiques, à partir de l'image binaire de référence, on peut construire un ensemble de masques de plus en plus larges. Ainsi, si on dispose par exemple d'une référence définie par des linéiques de m pixels d'épaisseur, les masques choisis auront une épaisseur de m+2, m+4,...,m+2n (cela ayant pour effet de fournir un lissage progressif en échelle entre un pixel de contexte et un pixel appartenant au thème retenu)[28]. Dans le cas des ponctuelles et des surfaciques, cela revient à générer des masques autour des zones d'intérêt définis par des carrés (Figure ci-dessous).

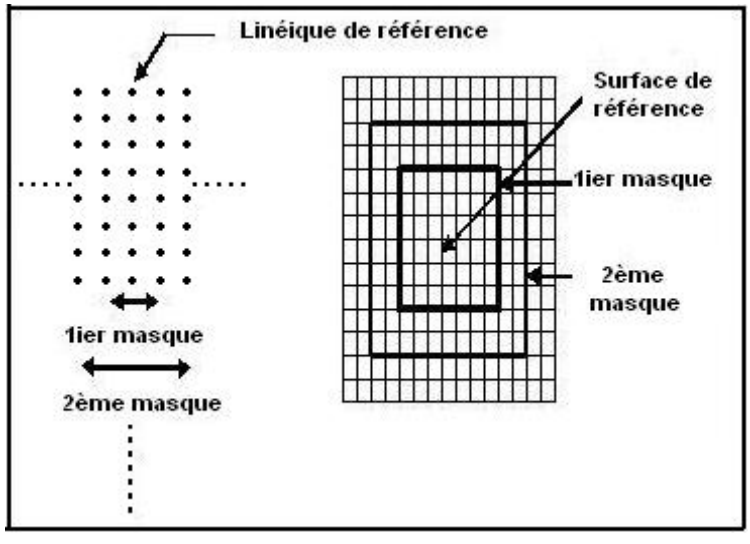
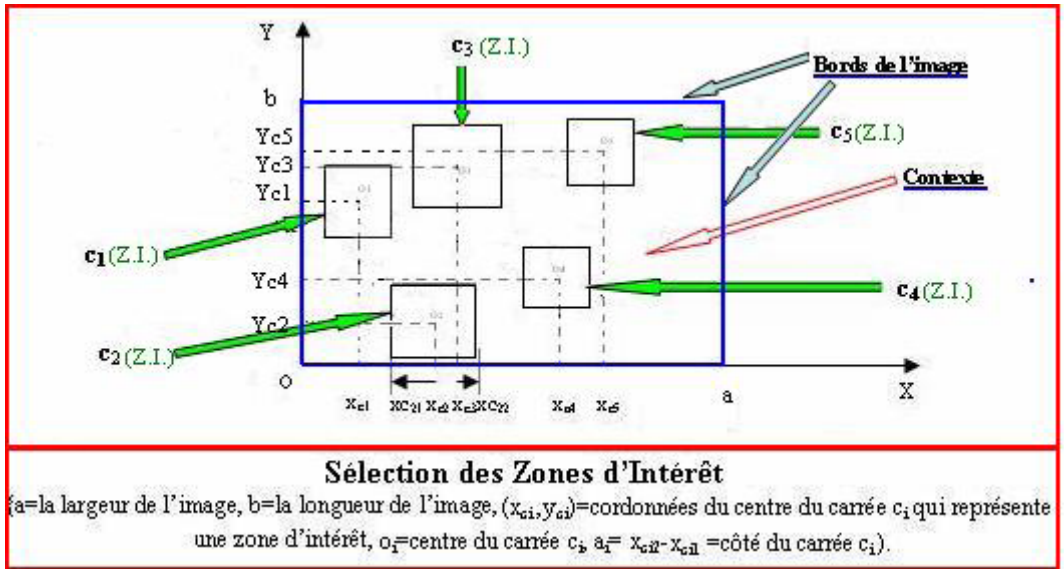


Figure 3.2 Génération des masques.

Dans cette étude, nous avons choisi d'extraire manuellement les zones d'intérêt sur la base de sélection surfacique qui consiste à sélectionner des carrés représenté chacun par trois paramètres : les deux premiers permettent de le localiser dans l'image initiale (cordonnées du centre (xi, yi) et le côté ai), le troisième paramètre correspond à sa valeur intrinsèque dans l'image c'est-à-dire les radiométries des pixels constituant la zone d'intérêt (Figure ci-dessous).



Sélection des Zones d'Intérêt

{a=la largeur de l'image, b=la longueur de l'image, (x_{ci},y_{ci})=cordonnées du centre du carrée c_i qui représente une zone d'intérêt, o_i=centre du carrée c_i, $a_i = x_{ci2}-x_{ci1}$ =côté du carrée c_i).

Figure. 3.3 Sélection des Zones d'Intérêt.

3.3.2 La compression par la méthode SPIHT

3.3.2.1 Introduction

SPIHT (*Set Partitioning in Hierarchical Trees*) est une méthode de compression d'images par plan de bits basée sur la transformé en ondelette. Elle a été développée dans [29]. Elle propose en fait une version améliorée du ZeroTree. Elle est essentiellement basée sur une approche différente pour la transmission des cartes de significativité. Cet algorithme permet de coder et reconstruire une image quasiment identique à l'originale au débit le plus faible.

3.3.2.2 Structures d'arbres

Un des points faibles possibles de JPEG 2000 est qu'il n'utilise pas les relations existantes entre les localisations des coefficients significatifs entre les différentes sous-bandes. Le gain venant du choix du point de troncature compense le fait que les relations entre les coefficients ne sont pas utilisées[30]. Le rapport entre les sous-bandes étant anormalement élevée.

Les arbres de zéros sur les coefficients d'ondelettes ont été développés pour utiliser la relation qui existe entre la position des singularités dans les différentes sous-bandes. Après la transformée en ondelettes, on observe que la localisation des coefficients significatifs est similaire entre les différentes sous-bandes même si leurs amplitudes sont décorrélées (Figure 3.4). La propriété à l'origine des arbres de zéros est que si un coefficient n'est pas significatif

dans une sous-bande alors le coefficient à la même position dans une sous-bande de plus haute fréquence sera lui aussi probablement non significatif.

Cette idée a été utilisée avec succès par Shapiro avec EZW (Embedded Zerotree of Wavelet coefficients) [31]. Une amélioration importante a été apportée quelques années plus tard par Said et Pearlman avec SPIHT. Toutefois, les performances des arbres de zéros sont supérieures à ce qu'on pouvait en attendre [32] et ce succès n'est pas complètement expliqué. L'idée de SPIHT a ensuite été reprise pour donner des solutions progressives en résolution [33], résistantes aux erreurs de transmission [34] ou généralisées pour des structures 3D, principalement pour la compression vidéo [35,36] ou pour les images médicales [37,38]. Sur le modèle de SPIHT, d'autres codeurs ont ensuite vu le jour comme SPECK [39,40] qui tire parti des similarités entre les coefficients adjacents dans une même sous-bande (blocs de zéros plutôt que des arbres de zéros). SPECK a ensuite été adapté en SBHP [41] pour concurrencer EBCOT [42] dans la norme JPEG 2000 ainsi qu'en EZBC [43] utilisant un codage arithmétique contextuel évolué pour augmenter les performances. LTW [44], amélioré dans [45] propose une approche différente pour réduire la complexité. Plus récemment, SPECK a été adapté pour fournir un codage progressif en résolution [46].

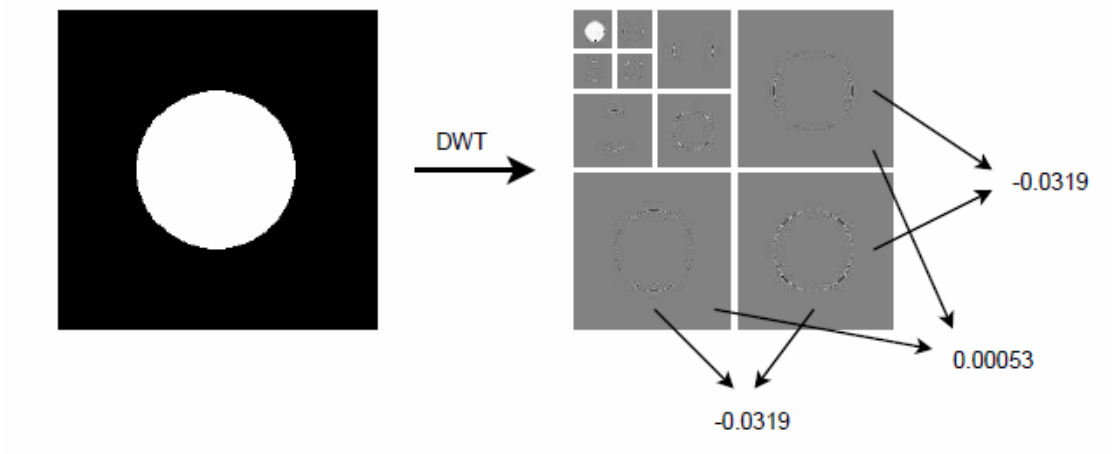


Figure 3.4 Illustration du principe des arbres de zéros

3.3.2.3 Principes généraux de EZW et SPIHT

Au moment de sa publication, l'algorithme EZW de Shapiro, utilisant les arbres de zéros, produisait les meilleures performances en terme de débit-distorsion tout en ayant une complexité faible. Cette structure d'arbres de zéros a été généralisée quelques années après par Said et Pearlman pour donner l'algorithme SPIHT.

Ces deux algorithmes possèdent des propriétés qui les rendent particulièrement attractifs dans le domaine de la compression spatiale embarquée. Ils produisent tous deux un train binaire emboîté : le préfixe d'un train binaire produit par EZW (ou SPIHT) est lui-même un train binaire EZW (ou SPIHT) valide conduisant à une image décompressée avec une qualité plus faible. Ces deux algorithmes atteignent ce résultat avec un niveau modeste de complexité.

Le fait d'avoir un train binaire emboîté est intéressant pour éviter les dépassements des mémoires et garantir la meilleure qualité d'image étant données les ressources disponibles à bord (mémoire et calcul).

3.3.2.4 Adaptation de EZW

La partie délicate dans le codage par plan de bits est le codage de la carte des coefficients significatifs, i.e. l'ensemble des décisions binaires pour savoir si un coefficient est significatif pour un seuil donné T_k. Les algorithmes EZW et SPIHT fournissent des moyens efficaces pour coder cet ensemble. Pour la plupart des coefficients (`a l'exception des sous-bandes de plus basse et de plus haute fréquences), avec la structure d'arbre optimale précédente, un coefficient $c_{x,y,\lambda}$ a deux descendants spectraux $c_{x,y,2\lambda}$ et $c_{x,y,2\lambda+1}$ et quatre descendants spatiaux $c_{2x2,2y,\lambda}$, $c_{2x2+1,y,\lambda}$, $c_{2x2,2y+1,\lambda}$ et $c_{2x2+1,2y+1,\lambda}$.

Chaque plan de bits est codé en deux passes. La première passe, appelée significance pass, code la carte des coefficients significatifs pour le plan de bits courant. La structure d'arbre de zéros permet de réduire le coût pour le codage de la carte en utilisant les relations entre les sous-bandes. La deuxième passe, refinement pass, code un bit pour chaque coefficient ayant été déclaré comme significatif à un autre plan de bits.

Comme il a été montré par Shapiro, il est utile de coder le signe des coefficients significatifs en même temps que la carte. En pratique 4 symboles différents sont utilisés : racine d'un arbre de zéros (Zero Tree Root ou ZTR), zéro isolé (Isolated Zero ou IZ), coefficient positif (POS) ou coefficient négatif (NEG). Chacun de ces symboles peut être codé en utilisant 2 bits. IZ signifie que le coefficient courant est en dessous du seuil mais qu'au moins un de ses descendants est au dessus (ce coefficient ne peut pas être inclus dans un arbre de zéros). Le symbole ZTR indique que le coefficient courant est en dessous du seuil et que tous ses descendants sont également en dessous du seuil (ou alors sont déjà déclarés comme significatifs et seront de toute façon traités durant la refinement pass). Le parcours des coefficients est effectué sousbande par sous-bande (Fig. 3.21 pour le cas 2D) de telle manière

qu'aucun coefficient ne soit vu avant un de ses parents. Au sein de chaque sous-bande, les coefficients sont parcourus en zigzag. Le codage est fait par un alphabet à quatre symboles (POS, NEG, ZTR and IZ).

Algorithme de EZW

Pour chaque plan de bits :

- Significant pass : Pour tous les coefficients non significatifs, coder un symbole ZTR, IZ, POS ou NEG selon le cas. Les coefficients sont traités dans l'ordre indiqué sur la figure 3.21.
- Refinement pass : écrire un bit pour tous les coefficients ayant été déclarés comme significatifs (sauf ceux du dernier plan de bits), ce bit correspond à la valeur du coefficient dans le plan de bits courant.

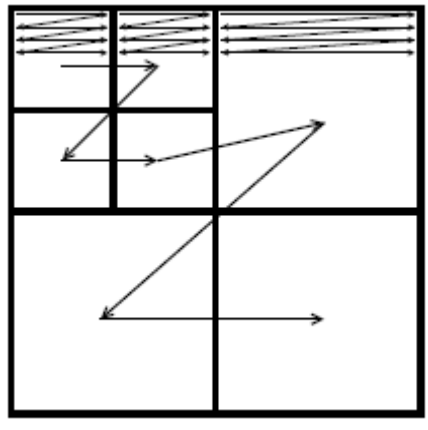


Figure 3.5 Ordre de parcours des coefficients pour EZW.

Il est important de noter que les coefficients ayant déjà été notés comme significatifs seront traités durant la refinement pass et peuvent donc être inclus sans dommage dans un arbre de zéros.

3.3.2.5 Adaptation de SPIHT

L'algorithme SPIHT est présenté par Said et Pearlman dans [29]. Les principales propriétés de EZW sont préservées : codage progressif et faible complexité. Cependant, quelques différences conduisent à une amélioration pour les images 2D classiques. L'algorithme SPIHT maintient trois listes de coefficients : la liste des coefficients significatifs (List of Significant Pixels ou LSP), la liste des coefficients non significatifs (List of

Insignificant Pixels ou LIP) et la liste des ensembles non significatifs. O(x, y, λ) est l'ensemble des enfants de (x, y, λ) (un seul niveau de descendance), D(x, y, λ) est l'ensemble de tous les descendants et L(x, y, λ) = D(x, y, λ) −O(x, y, λ) est l'ensemble des descendants à l'exception des enfants (Fig. 3.23). La fonction St(x, y, λ) est égale à 0 si tous les descendants de (x, y, λ) sont en dessous du seuil Tt (arbre de zéros) et 1 dans le cas contraire.

La première différence à noter par rapport à EZW est que toute sortie est binaire. La seconde différence est que l'ordre de traitement des coefficients est dépendant des données. Alors que les coefficients sont traités en zigzag dans chaque sous-bande pour EZW (Fig. 3.21), le système de liste de SPIHT laisse l'ordre entièrement dépendant des données. Les coefficients sont traités selon leur position dans les listes. Une autre différence à noter est la relation parent-enfant pour la sous-bande de plus basse fréquence (Fig. 3.24). Dans le cas 2D pour EZW, chaque coefficient pour la sous-bande basses fréquences (LL) possède 3 descendants (dans LH, HL et HH). Pour SPIHT, un descendant sur quatre n'a pas de descendant tandis que les autres en ont 4. La définition des arbres est aussi sensiblement différente car SPIHT considère deux types d'arbres de zéros : le type A où tous les descendants ne sont pas significatifs (arbre de degré 1) et le type B où tous les descendants, à l'exception d'au moins un des enfants, ne sont pas significatifs (arbre de degré 2). Il est à noter que dans les deux cas, rien n'est dit sur la valeur du coefficient à la racine qui peut être significatif.

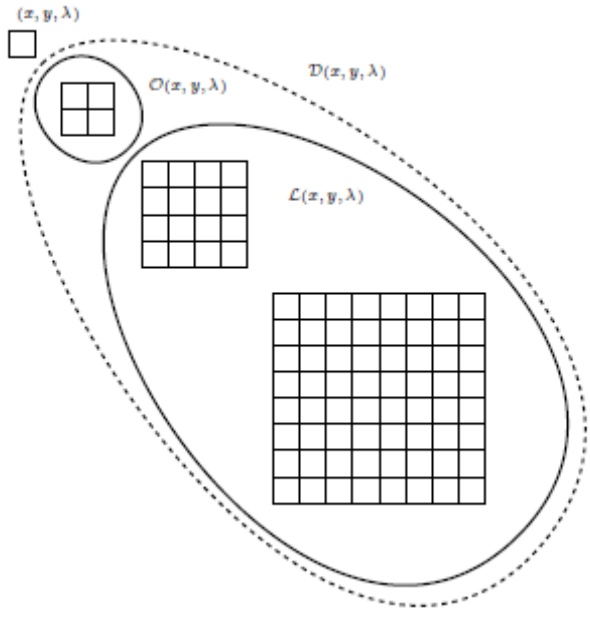


Figure 3.6 Terminologie SPIHT pour les descendants.

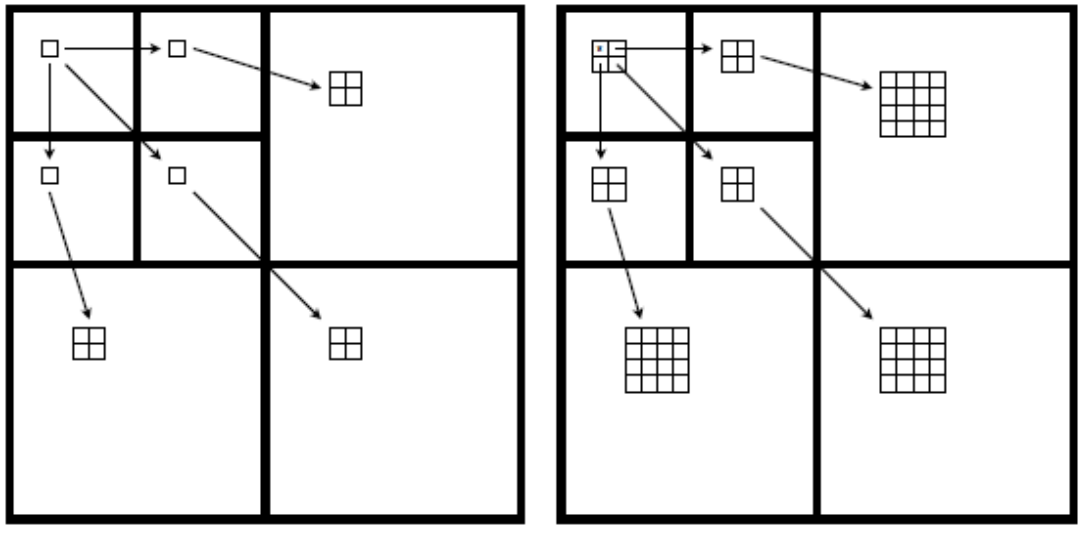


Figure. 3.7 Différences pour la descendance des coefficients basses fréquences entre EZW (a) et SPIHT (b).

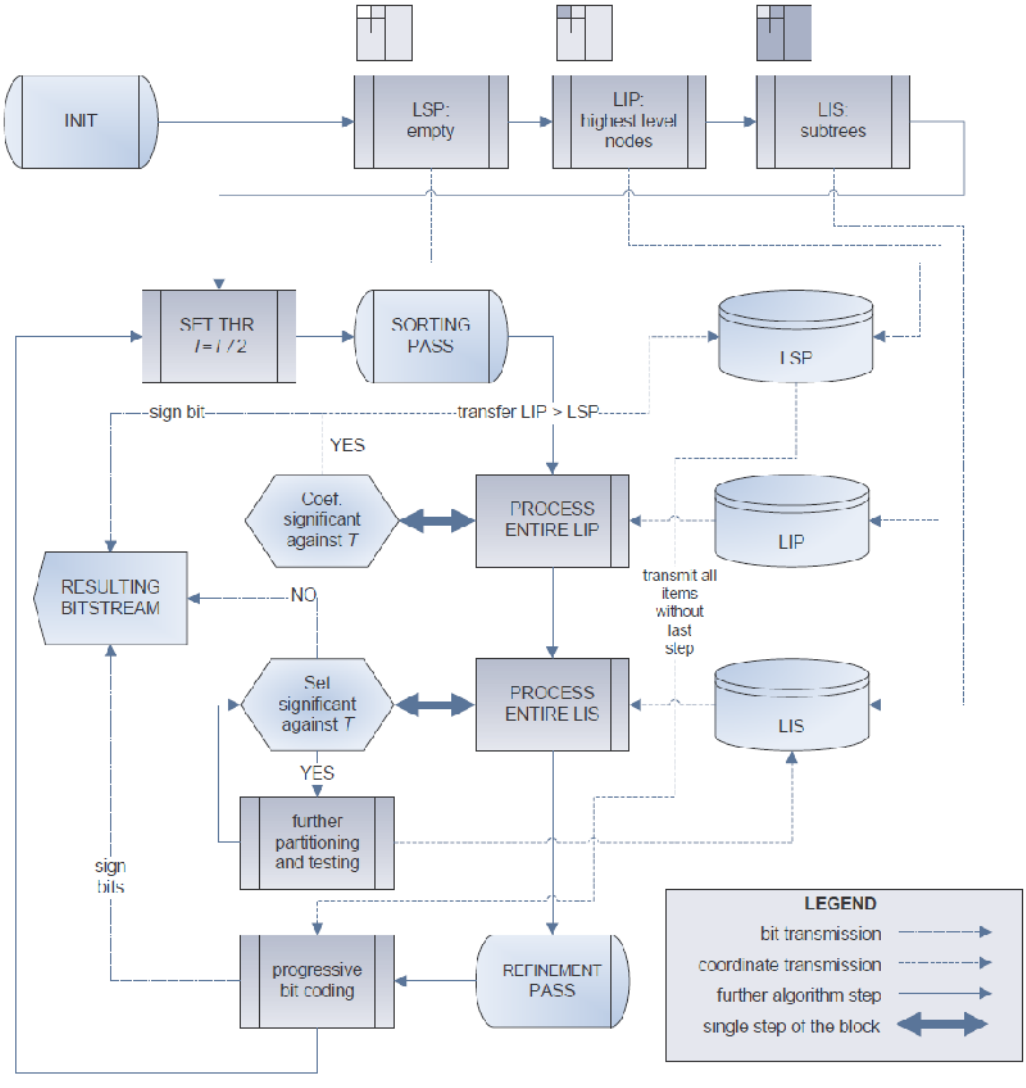


Figure 3.8 Schéma de compression de la méthode SPIHT.

Algorithme de SPIHT

Initialisation :

– Pour chaque plan de bits t :

– LSP = Φ

– LIP : tous les coefficients sans parent (coefficients de la LLL)

– LIS : tous les coefficients de la LIP qui ont des descendants (marqués comme type A, par défaut)

Sorting pass :

Pour chaque coefficient (x, y, λ) de la LIP

– Ecrire St(x, y, λ)

– Si St(x, y, λ) = 1, déplacer (x, y, λ) dans la LSP et écrire le signe de $c_{x,y,\lambda}$

Pour chaque coefficient (x, y, λ) de la LIS

– Si le coefficient est de type A

_ Ecrire St(D(x, y, λ))

_ Si St(D(x, y, λ)) = 1 alors

_ Pour tout (x',y', λ')

_ Pour tout (x′, y′,λ ′) $\in$ O(x, y, λ) : écrire St(x′, y′,λ ′) ; si St(x′, y′,λ ′) = 1, ajouter (x′,

_ y′,λ ′) à la LSP et écrire le signe de $c_{x',y',\lambda'}$ sinon, ajouter (x′, y′,λ ′) `a la fin de la LIP.

(Etape critique pour le problème du tree-crossing).

_ Si L(x, y, λ) ≠ Φ, déplacer (x, y, λ) à la fin de la LIS comme une entrée de type B

_ Sinon, retirer (x, y, λ) de la LIS

– Si l'entrée est de type B

– Ecrire St(L(x, y, λ))

– Si St(L(x, y, λ)) = 1

_ Ajouter tous les (x′, y′,λ ′) $\in$ O(x, y, λ) à la fin de la LIS comme entrée de type A

_ Retirer (x, y, λ) de la LIS

Refinement pass :

_ Pour tous les coefficients (x, y, λ) de la LSP à l'exception de ceux ajoutés au cours de la dernière sorting pass : Ecrire le t^{eme} bit le plus significatif de $c_{x,y,\lambda}$

_ Décrémenter t et retourner à la sorting pass.

Le décodeur est obtenu en remplaçant écrire par lire dans l'algorithme précédent.

Le phénomène de tree-crossing a un impact plus grand dans le cas de SPIHT que dans le cas de EZW. Durant l'algorithme, un coefficient peut être traité avant qu'un de ses parents ne le soit, un soin particulier doit être pris pour éviter de traiter le même coefficient plusieurs fois. On garde donc en mémoire le fait qu'un coefficient a déjà été traité. Cette information n'a pas besoin d'apparaître dans le train binaire de sortie, le décodeur partageant les états du codeur, il sera capable de faire la même opération. Le phénomène du tree-crossing a un impact plus important probablement parce que la destruction d'un arbre (par apparition d'un coefficient significatif) a des conséquences plus importantes pour SPIHT, qui essaie de maintenir des arbres au maximum (par le biais des arbres de degrés 1 et 2), que pour EZW.
Grace aux arbres de degrés 1 et 2, SPIHT tire également plus parti du lien spectral apparaissant uniquement pour les coefficients de basse fréquence spatiale.

3.3.2.6 Exemples de compression par la méthode SPIHT

Figure 3.9 Lena image compression result at 0.3bpp

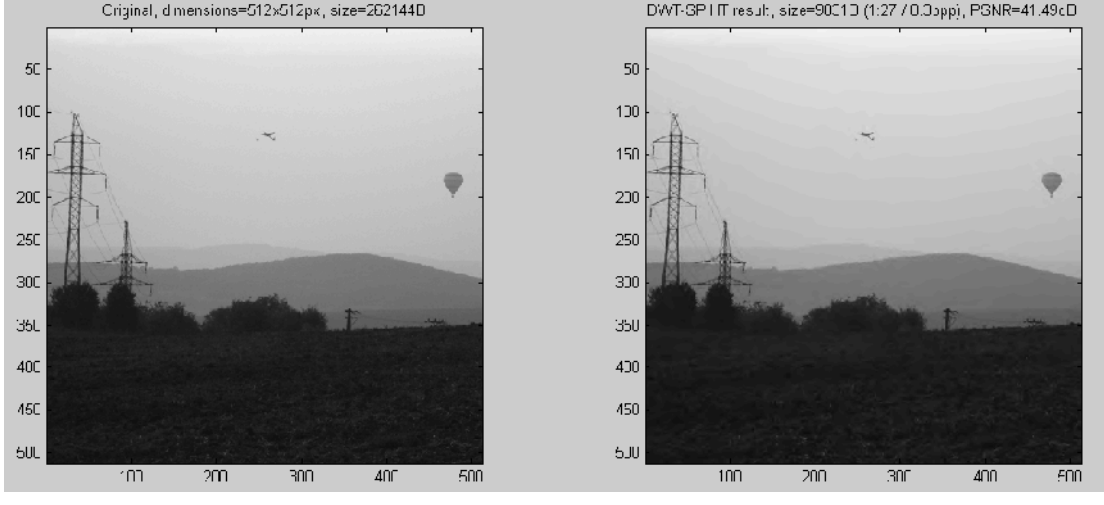


Figure 3.10 Scenery image compression result at 0.3bpp

Figure 3.11 City image compression result at 0.3bpp

3.3.2.7 Eléments pratiques de comparaison des différentes méthodes de compression

On peut comparer les méthodes essentiellement sur deux plans : d'une part l'efficacité en coût de codage qui permet d'atteindre des taux de compression importants et d'autre part la complexité d'implémentation de l'algorithme et sa rapidité d'exécution. On s'intéresse ici essentiellement au premier aspect.

Lena : 0.31bpp, 35,12 dB Codage SPIHT Lena : 0.31bpp, 31,8 dB Codage JPEG

figure 3.12 Comparaison de deux images codées au même débit par ondelettes (SPIHT) et par DCT (JPEG).

On observe que les méthodes de codage SPIHT permettent d'obtenir de meilleurs résultats que les méthodes de codage par DCT. La comparaison des PSNR obtenus ainsi que

les qualités visuelles des images compressées aux mêmes taux sont en effet favorables à ces méthodes (cf figure 3.12 où l'on peut voir des artefacts de type bloc provoqués par la méthode de codage DCT, tandis que les défauts visibles de SPIHT sont très limités). Plus le débit est réduit, plus l'écart entre les deux familles de méthodes est important. Cette différence de qualité provient de la nature des fonctions de base des ondelettes, qui contrairement aux fonctions de la base DCT, ont des supports qui se recouvrent pour des blocs voisins. Cela permet d'éviter les problèmes de discontinuité locale qui apparaissent en particulier dans le cas de la DCT aux frontières des blocs lorsque le débit est insuffisant. Par ailleurs les modèles sous-bandes permettent de mieux visualiser les corrélations entre coefficients et d'élaborer des modèles qui exploitent les redondances de l'image transformée de manière proche de l'optimalité.

Pour ce qui est des performances respectives des différentes méthodes de compression, on trouve des comparatifs pour quelques images standards. Les meilleurs résultats de compression sont fournis par SPIHT, JPEG2000. En figure 3.13 on montre les qualités de reconstruction pour différents débits pour deux images (Goldhill et Baboon). On observe un avantage net pour les méthodes SPIHT et JPEG2000. On remarque que l'on observe peu de différences entre ces deux méthodes : malgré leurs principes de bases différents elles fournissent à peu près la même qualité de reconstruction pour un débit équivalent.

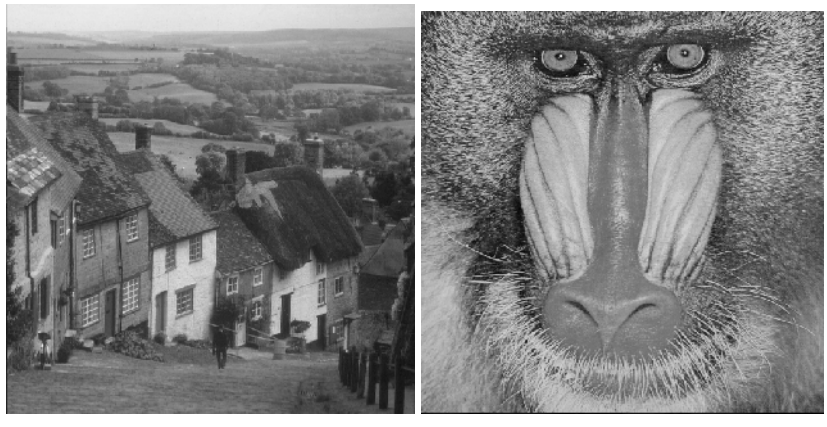
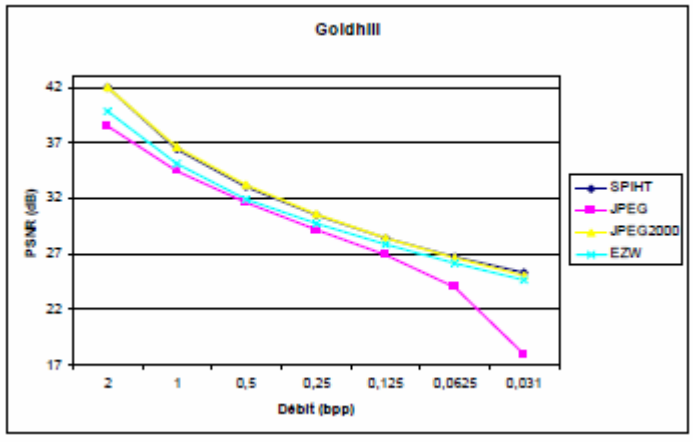
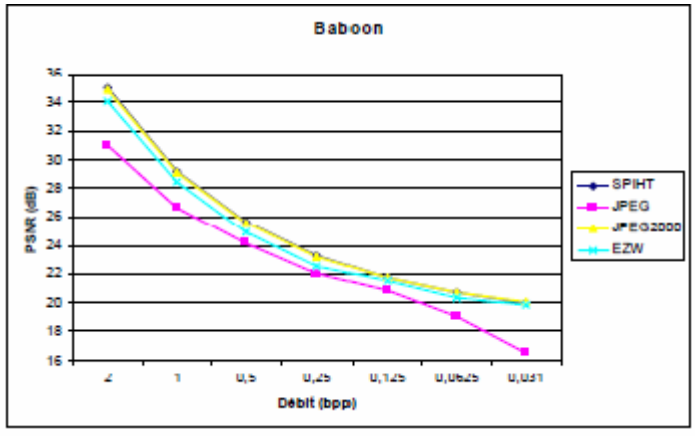


figure 3.13 (a) Goldhill (b) Baboon



A



B

figure 3.14 Comparaison de différentes méthodes de codage sur deux images

3.4 Méthode de compression proposée

Dans plusieurs applications de traitement d'image, telles que l'imagerie médicale, l'imagerie satellitaire, la télésurveillance etc.., les régions d'intérêt nécessitent d'être codées avec une grande précision. Dans des cas pareils, la solution est d'appliquer une compression sans pertes au niveau des régions d'intérêt. Les informations relatives à l'arrière plan sont aussi intéressantes puisqu'elles servent à localiser les régions d'intérêt par rapport à l'image. Cependant, l'arrière plan ne nécessite pas d'être représenté avec une grande précision. Ceci suggère d'appliquer une compression sans pertes sur les régions d'intérêt ou avec une légère dégradation, et avec pertes sur l'arrière plan.

Dans ce travail, nous n'accordons pas d'attention pour la détection automatique des zones d'intérêt, mais plutôt nous nous intéressons à la qualité de l'image et au taux de compression au niveau des zones d'intérêt et de l'image toute entière après compression[47]. Nous proposons d'appliquer deux méthodes de compression, une réversible sur les zones d'intérêt et une autre irréversible sur le reste de l'image. SPIHT offre la possibilité du codage des régions d'intérêt jusqu'à une compression sans pertes. Cependant, le choix du JPEG2000 avec perte permet de dégrader de façon contrôlée les autre zones de l'image .Nous comparons les performances de l'approche proposée, en termes de qualité visuelle et taux de compression, avec ceux obtenus en effectuant une compression de l'image entière par JPEG et par JPEG2000.

la méthode de compression sélective proposée présente un ensemble d'avantage :

- elle permet de compresser les régions d'intérêt et l'arrière plan de façon séparée (nous pouvons employer une variété de techniques de compression dans une seule image)
- le fait de séparer le codage des régions d'intérêt de celui du contexte donne la possibilité à l'utilisateur de choisir entre reconstruire l'image entière ou laisser les régions d'intérêt isolées pour subir un traitement ultérieur.

3.4.1 Application à l'image avec une ROI

Le schéma de codage pour ce type d'image est décrit dans la figure 4.3, la ROI est extraite à l'aide de la méthode manuelle.

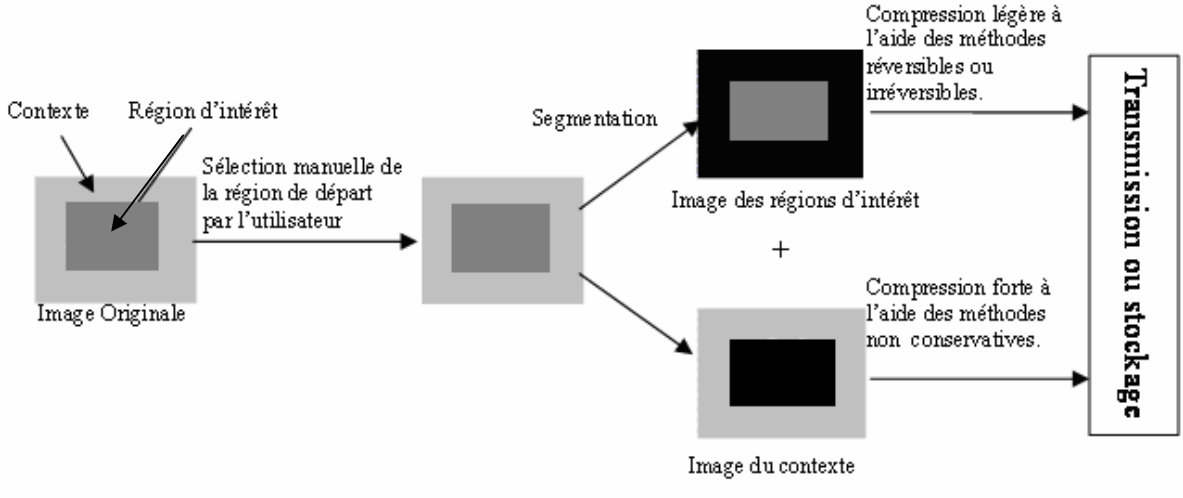
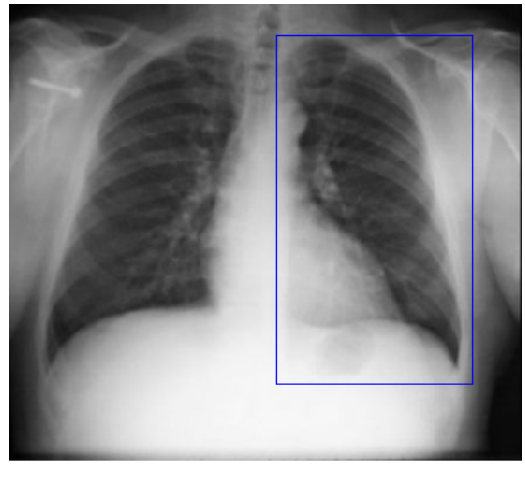


Figure 3.15 Schéma de codage appliqué aux images contenant une ROI

Application aux images biomédicales

La méthode a été appliquée à deux types d'images avec une région d'intérêt.
Les figures qui suivent montre les résultats de l'extraction de la région d'intérêt et de la compression d'une Radiographie normale des poumons de face, ainsi que les résultats correspondant à un scanner présentant un abcès du cerveau (région d'intérêt). Les images de départ sont de type PNG en niveaux de gris 8 bits par pixel. Nous avons compressé les régions d'intérêt à une qualité de 85% tandis que l'arrière plan à 25% de qualité. Nous avons testé la méthode avec différents algorithmes de compression: JPEG, JPEG 2000, la table 3.1 illustre les résultats de la compression uniforme (sans distinction entre les régions d'intérêt et le contexte) aux images " poumons de face " et "cerveau" et la table 3.2 montre les résultats de l'application de la compression sélective à ces images. En comparant les résultats obtenus on constate que la compression sélective permet d'atteindre des taux de compression élevés avec une qualité visuelle appréciée surtout au niveau des régions d'intérêt (voir figures 3.19 et 3.23), mais cela dépend des dimensions et le nombre de régions d'intérêt présentes dans l'image.

Première image : Image poumons de face



.
Figure 3.16 Image **poumons de face (originale)** : Dimensions=438x387 pixels.. Taille du fichier 92.90ko. Type de fichier=PNG

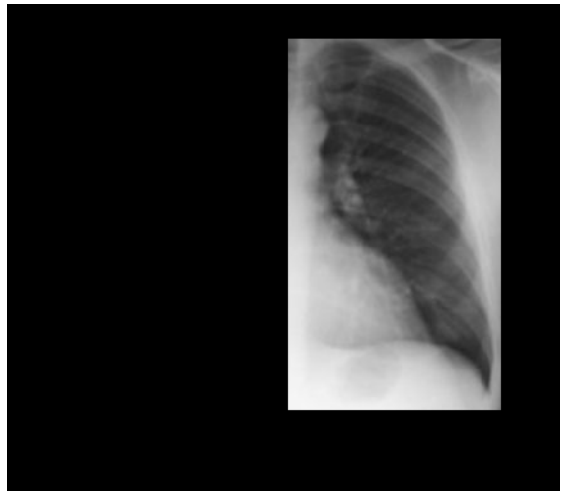


Figure 3.17 Image contenant la région d'intérêt compressée sans contexte: Dimensions=438x387 pixels. Taille de fichier=34.7 Ko

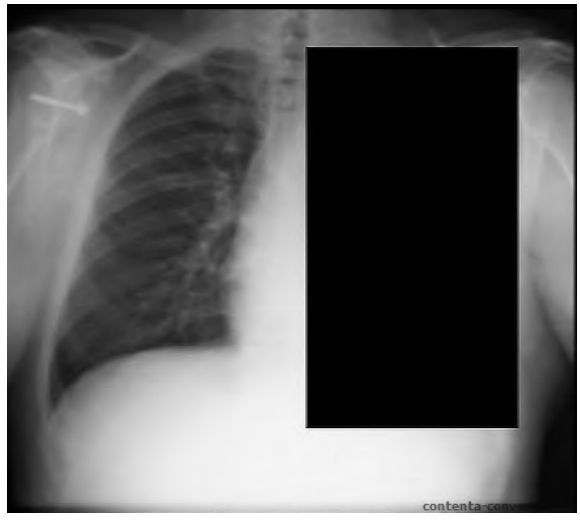


Figure 3.18 Image contexte compressée sans région d'intérêt : Dimensions=438x387 pixels. 66.60 ko.

Figure 3.19 Image **poumons de face** reconstruite

Deuxième image : Image scanner d' un abcès du cerveau

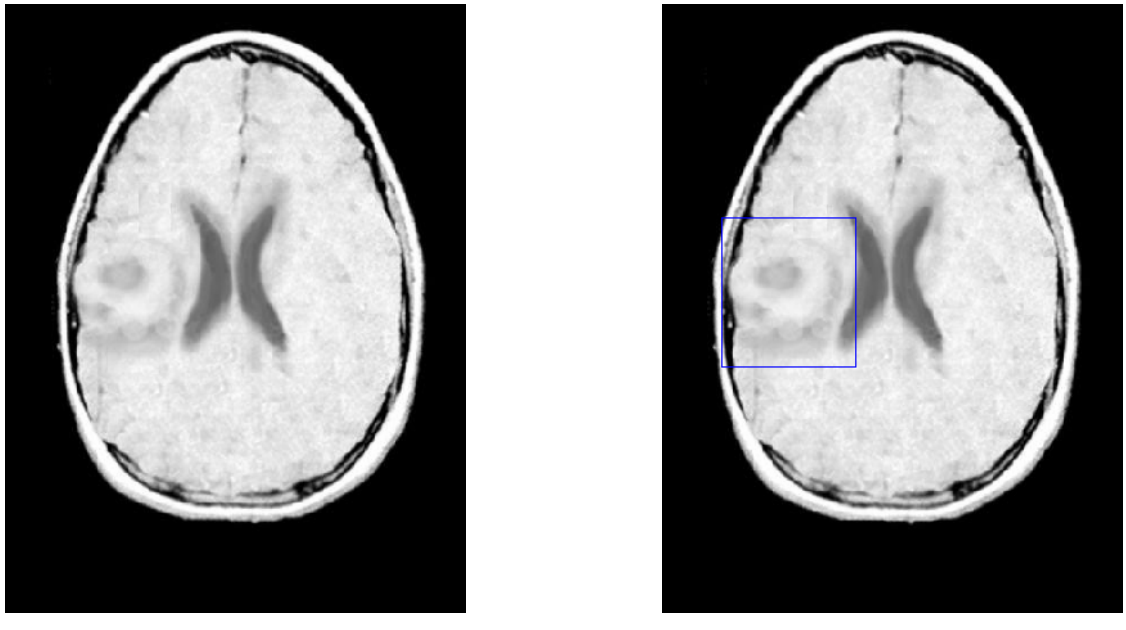


Figure 3.20(a) Image scanner présentant un abcès du cerveau : Dimensions=404x535 pixels. Taille de fichier=90,90Ko. Type de fichier=PNG.

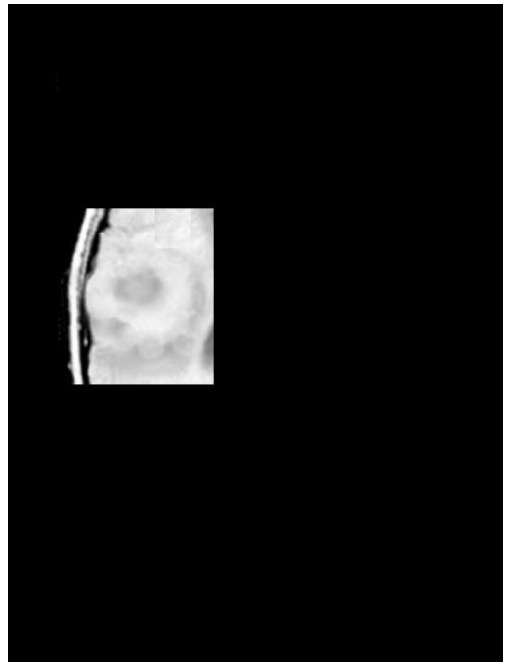


Figure 3.21 Image contenant la région d'intérêt compressée sans contexte : Dimensions=404x535 pixels. Taille de fichier=15.8 Ko.

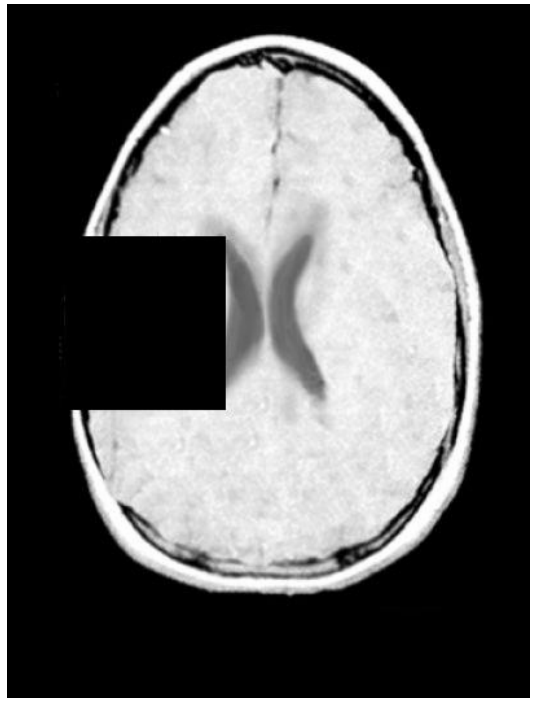


Figure 3.22 Image contexte compressée sans région d'intérêt : Dimensions=404x535 pixels. Taille de 79.70ko.

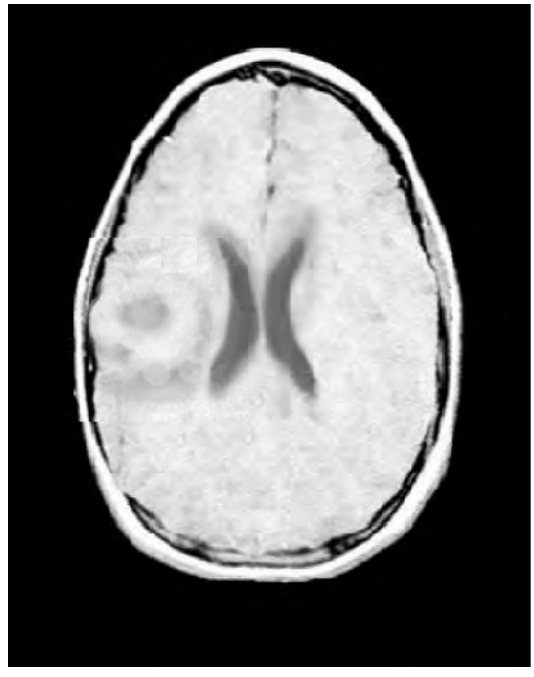


Figure 3.23 Image scanner présentant un abcès du cerveau reconstruite.

	PNG	JPEG	JPEG 2000	SPIHT
Poumons de face	92.90ko*	81.70ko**	21.00ko*	10.35ko*
Cerveau	90.90ko*	31.80ko**	26.40ko*	13.1ko*

Table 3.1 Résultats de la compression uniforme des images " **poumons de face** " et "Cerveau". *Compression sans perte. **Compression avec maximum de qualité. Ko :Kilo Octet.

		PNG	**JPEG**	**JPEG 2000**	**SPIHT**
Poumons de face	Arrière plan	66.60ko	25.70ko qualité 80 %	16.6ko	
	Région d'intérêt	34.7ko	15.00ko qualité 25%	5.20ko	5.23ko Cr=0.05bpp
Cerveau	Arrière plan	79.70ko	42.60ko	6.37KO	
	Région d'intérêt	15.80ko	8.64ko	6.67ko	4.15 Cr=0.05bpp

Table 3.2 Résultats de la compression sélective des images " **poumons de face** "et "**Cerveau**".*Compression sans perte. **Compression avec maximum de qualité. Ko :Kilo Octet. R : Taux de compression.

Applications aux images originales (images de personnes)

Première image : Lena

Figure 3.23 Image de Lena Dimensions=512x512pixels. Taille du fichier 218ko. Type de fichier=PNG.

Figure 3.24 Image contenant la région d'intérêt compressée sans contexte : Dimensions=512x512 pixels. Taille de fichier=30.8 Ko.

Figure 3 .25. Image contexte compressée sans région d'intérêt : Dimensions=512x512 pixels. Taille de 155ko.

Figure 3.26 Image Lena reconstruite

Deuxième image : Stefan

Figure 3.27 Image de Stefan Dimensions=351x286pixels. Taille du fichier 94.8ko. Type de fichier=PNG.

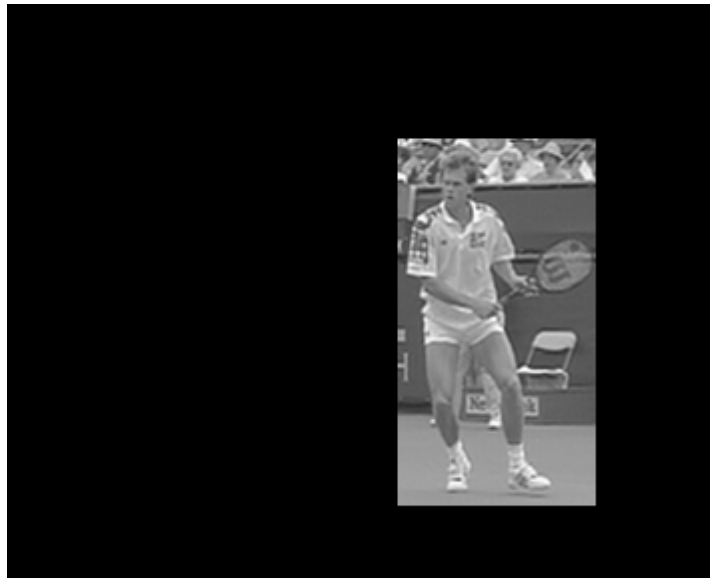


Figure 3.28 Image contenant la région d'intérêt compressée sans contexte : Dimensions=351x286 pixels. Taille de fichier=23.5 Ko.

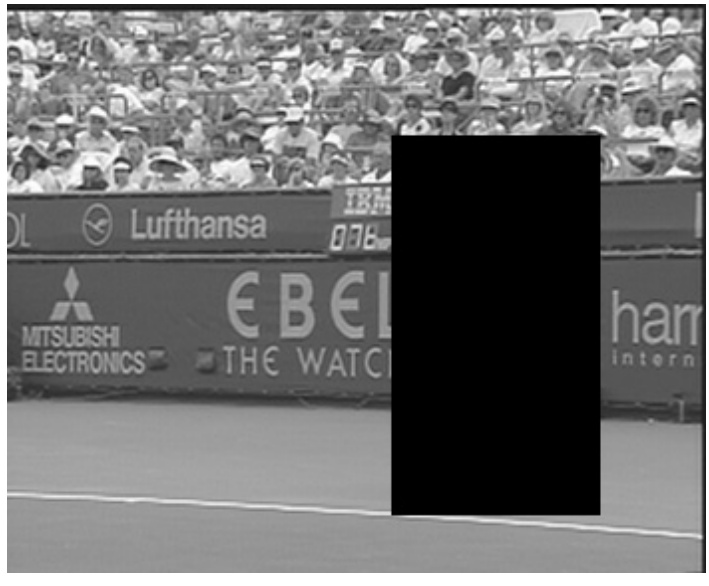


Figure 3.29 Image contexte compressée sans région d'intérêt : Dimensions=351x286 pixels. Taille de 79.7ko.

Figure 3.30 Image de Stefan reconstruite

	PNG	JPEG	JPEG 2000	SPIHT
Lena	218ko*	28.70ko**	23.90ko*	16.00ko*
Stefan	94.80ko*	73.20ko**	9.14ko*	6.13ko*

Table 3.3 Résultats de la compression uniforme des images "**Lena**"et "**Stefan**". *Compression sans perte. **Compression avec maximum de qualité. Ko :Kilo Octet.

		PNG	JPEG	JPEG 2000	SPIHT
Lena	Arrière plan	155 ko	39.50 ko qualité 25 %	8.04 ko	
	Région d'intérêt	30.8 ko	31.60ko qualité 85%	27.20ko	6.20 ko Cr=0.05bpp
Stefan	Arrière plan	79.7 ko	15.20 ko	3.03 ko	
	Région d'intérêt	23.5 ko	10.90 ko	10.40 ko	6.13 ko Cr=0.05bpp

Table 3.4 Résultats de la compression sélective des images "**Lena**" et "Stefan".*Compression sans perte. **Compression avec maximum de qualité. Ko :Kilo Octet. R : Taux de compression.

3.5 Conclusion

Dans ce chapitre, nous avons présenté notre approche de compression sélective d'image basée sur une extraction manuelle des régions d'intérêt.

Dans un premier temps, nous avons donné un aperçu des méthodes de compression par région d'intérêt existantes. Ensuite, nous avons présenté la méthode de compression SPIHT avec laquelle nous avons comprimer les parties intéressantes de nos images, Après, nous avons expliquer notre approche et les résultats de l'application de celle-ci sur un ensemble d'images. Ces résultats ont révélé que la compression sélective séparée permet d'obtenir un taux de compression supérieur à ceux des méthodes existantes tout en conservant une qualité maximale au niveau des régions d'intérêt (sans pertes). En plus, La méthode de compression proposée peut intégrer de manière intelligente des techniques d'analyse multirésolution et des techniques de compression existantes (avec ou sans perte). Elle offre à l'utilisateur la possibilité de traiter de façon optimale chaque donnée en fonction de son type et d'obtenir, par conséquent, des taux de compression relativement élevés et une meilleure qualité.

Chapitre 4

Conclusion générale

La compression des données est appelée à prendre un rôle encore plus important en raison du développement des réseaux et du multimédia. Son importance est surtout due au décalage qui existe entre les possibilités matérielles des dispositifs que nous utilisons (débits sur Internet, sur Numéris et sur les divers câbles, capacité des mémoires de masse,...) et les besoins qu'expriment les utilisateurs (visiophonie, vidéo plein écran, transfert de quantités d'information toujours plus importantes dans des délais toujours plus brefs). Quand ce décalage n'existe pas, ce qui est rare, la compression permet de toutes façons des économies.

La plus part des méthodes de compression ne sont pas figées et définitives. Même si la plus grande part du chemin dans la compression des données semble avoir été accomplie, les recherches se poursuivent pour améliorer encore les performances, au prix d'une complexité toujours accrue, mais compatible avec les progrès parallèles des outils informatiques.

Nous avons présenté dans ce travail une méthode de compression sélective d'image basée sur une extraction manuelle et assistée des régions d'intérêt. La méthode consiste à séparer le processus de codage des régions qualifiées d'intérêt par l'utilisateur de celui de l'arrière plan. L'objectif de la compression sélective est d'améliorer le taux de compression tout en conservant l'information transmise à travers les régions d'intérêt. Ceci est réalisé en appliquant deux méthodes de compression une réversibles et l'autre irréversibles . Nous avons choisi SPIHT comme méthode réversible avec la quelle nous avons codé les ROIs avec une légère dégradation, JPEG2000 est utilisée pour coder le reste de l'image avec une dégradation de façon contrôlable. La méthode d'extraction manuelle est une méthode supervisée. C'est-à-dire elle est basée sur l'intervention de l'utilisateur qui guide l'algorithme d'extraction en sélectionnant une partie des régions d'intérêt. La méthode d'extraction assistée

consiste à appliquer une décomposition de l'image en régions pour faciliter la sélection des régions d'intérêt par l'utilisateur. L'utilisateur sélectionne facilement les régions qui l'intéressent. Après extraction des régions d'intérêt, celles-ci sont compressées en appliquant SPIHT sans pertes. L'arrière plan est compressé en utilisant JPEG200 avec pertes. Les résultats obtenus en appliquant cette méthode ont montré une supériorité vis-à-vis des méthodes classiques en terme du taux de compression et de la qualité visuelle au niveau des régions d'intérêt.

Le travail présenté dans ce mémoire a plusieurs perspectives. La compression sélective peut notamment avoir une extension pour qu'elle soit appliquée aux images animées. La technique d'extraction des régions d'intérêt doit être améliorée pour diminuer l'intervention de l'utilisateur surtout dans le cas d'une séquence vidéo qui contient plusieurs images. La compression sélective d'image peut être utilisée aussi dans le domaine d'archivage d'image ou la recherche d'image par composante visuelle.

Bibliographie

[1] E. Incerti. Compression d'image - Algorithmes et standards. Vuibert, 2003.

[2] A. Trémeau, C. Fernandez-Maloigne, and P. Bonton. Image numérique couleur - De l'acquisition au traitement. Dunod, 2004.

[3] M. Délibéré. "La couleur". Presses Universitaires de France, 220, 1989.

[4] G. A. Agoston. Color Theory and its Application in Art and Design. Optical Sciences. Springer-Verlag, 1987.

[5] J. P. Guillois. Compression de données : Compression des images. Techniques de l'ingénieur, Paris, FRANCE (Revue), TEB1(E5340) :1-32, 1998.

[6] F. Davoine. Compression d'images par fractales basée sur la triangulation de Delaunay. PhD thesis, Institut National Polytechnique de Grenoble, 1995.

[7] J. Ziv and A. Lempel. A universal algorithm for sequential data compression. IEEE Transactions on Information Theory, pages 337-343, May 1977.

[8] J. Ziv and A. Lempel. Compression of individual sequences via variable rate coding. IEEE Transactions on Information Theory, pages 530-536, 1978.

[9] T.A. Welch. A technique for high performance data compression. IEEE Computer, 17(6), 1984.

[10] R. Gentille, J Allebach, and E. Wallowitt. Quantization of color images based on uniform color spaces. Journal of Imaging Technology, 16(1) :11-21, Feb 1990.

[11] S. P. Lloyd. Least squares quantization in pcm. IEEE on Information Theory, March 1982.

[12] J. Max. Quantizing for minimum distorsion. IRE on Information Theory, March 1960.

[13] Y. Linde, A. Buzo, and R. M. Gray. An algorithm for vector quantization design. IEEE Transactions on Communications, 16 :84-95, 1980.

[14] R. E. Crochiere, S. A. Webber, and J. L. Flanagan. Digital coding of speech in subbands. Bell. Syst. Tech. Journal, pages 1069-1085, 1976.

[15] S. G. Mallat. A theory for multiresolution signal decomposition : The wavelet representation. IEEE Transactions on Pattern Analysis and Machine Intelligence, pages 674-693, 1989.

[16] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies. Image coding using wavelet transform. IEEE Transactions on Image Processing, pages 205-220,1992.

[17] M. Barlaud, P.Solé, T. Gaidon, M. Antonini, and P. Mathieu. Pyramidal lattice vector quantization for multiscale image coding. IEEE Transactions on Image Processing, pages 367-381, 1994.

[18] V. Bhaskaran and K. Konstantinides. Image and Video Compression Standards : Algorithms and Applications. Kluwer Academic Publishers, 1997.

[19] W. B. Pennebaker and J. L. Mitcell. JPEG : Still Image Data Compression Standard. Van Nostrand Reinhold, 1993.

[20] G. K. Wallace. The jpeg still picture compression standard. IEEE Transactions on Consumer Electronics, 38(1), Feb 1992.

[21] C. Christopoulos, A. Skodras, and T. Ebrahimi. The JPEG2000 still image coding system : An overview. IEEE Transactions on Consumer Electronics, 46(4) :1103-1127, November 2000.

[22] DD.Guisto, C.S. Regazzoni, S.B.Serpico, G. Vernazza, " A New Adaptative Approach to
Picture Coding ", Ann. Télécommunications, 45, N_9-10, pp.503-518, 1990.

[23] E.Nguyen, " Compression sélective et focalisation visuelle : Application au codage hybride de séquences d'images ", thèse de l'université de Rennes I, décembre 1995.

[24] Jean- Michel Benharrosh, " Extraction de thèmes cartographiques dans les images satellitales ou aériennes : Application à la génération de Quick-Looks adaptatifs et à la compression des images ", thèse de doctorat, Université de Nice-SOPHIA ANTIPOLIS , Avril 1998.

[25] F.Falzon, " Analyse Multi- échelle, Détection des singularités et Caractérisation de la Régularité des Images ", thèse de doctorat, Nice - Sophia- Antipolis, Décembre 1994.

[26] Rapport d'activité sur Projet TEMICS," Traitement, Modélisation d'images et Communications ", Université de Rennes, 2001.

[27] Véronique EGLIN, " Communication, Images et Numérique ", RFV-INSA de Lyon, 2001.

[28] A. Bijaoui, J.L.Strack, F. Murtagh, " Restauration des images MultiEchelles par l'algorithme à trous ", INIST-CNRS, I-Revues, Traitement du signal et des Images, vol.11, pp.229- 243, 1994.

[29] A. Said et W. A. Pearlman. “A new, fast, and efficient image codec based on set partitioning in hierarchical trees”. IEEE Transactions on Circuits and Systems for Video Technology, vol. 6, no 3, p. 243–250, juin 1996.

[30] D. S. Taubman et M. W. Marcellin. JPEG2000 Image Compression Fundamentals, Standards and Practice. Kluwer Academic Publishers, Boston, MA, 2002.

[31] J. M. Shapiro. “Embedded image coding using zerotrees of wavelet coefficients”. IEEE Transactions on Signal Processing, vol. 41, no 12, p. 3445–3462, d´ec. 1993.

[32] M. W. Marcellin et A. Bilgin. “Quantifying the parent-child coding gain in zero-tree-based coders”. IEEE Signal Processing Letters, vol. 8, no 3, p. 67–69, mars 2001.

[33] H. Danyali et A. Mertins. “Fully spatial and SNR scalable, SPIHT-based image coding for transmission over heterogenous networks”. Journal of Telecommunications and Information Technology, , no 2, p. 92–98, 2003.

[34] E. Christophe, D. L´eger et C. Mailhes. “Comparison and evaluation of quality criteria for hyperspectral imagery”. Dans Image Quality and System Performance II, vol. 5668, p. 204–213. SPIE, jan. 2005.

[35] B.-J. Kim, Z. Xiong et W. A. Pearlman. “Low bit-rate scalable video coding with 3-D set partitioning in hierarchical tress (3-D SPIHT)”. IEEE Transactions on Circuits and Systems for Video Technology, vol. 10, no 8, p. 1374–1387, d´ec. 2000.

[36] C. He, J. Dong et Y. F. Zheng. “Optimal 3-D coefficient tree structure for 3-D wavelet video coding”. IEEE Transactions on Circuits and Systems for Video Technology, vol. 13, no 10, p. 961–972, 2003.

[37] A. Bilgin, G. Zweig et M. W. Marcellin. “Three-dimensional image compression with integer wavelet transforms”. Applied optics, vol. 39, no 11, p. 1799–1814, avr. 2000.

[38] Z. Xiong, X. Wu, S. Cheng et J. Hua. “Lossy-to-lossless compression of medical volumetric data using three-dimensional integer wavelet transforms”. IEEE Transactions on Medical Imaging, vol. 22, no 3, p. 459–470, mars 2003.

[39] A. Islam et W. A. Pearlman. “An embedded and efficient low-complexity hierarchical image coder”. Dans Visual Communications and Image Pro- cessing ’99, vol. 3653, p. 294–305. SPIE, jan. 1999.

[40] W. A. Pearlman, A. Islam, N. Nagaraj et A. Said. “Efficient, lowcomplexity image coding with a set-partitionning embedded block coder”. IEEE Transactions on Circuits and Systems for Video Technology, vol. 14, no 11, p. 1219–1235, nov. 2004.

[41] C. Chrysafis, A. Said, A. Drukarev, A. Islam et W. A. Pearlman. “SBHP - a low complexity wavelet coder”. Dans IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 2000, vol. 4, p. 2035–2038. 2000.

[42] D. Taubman. “High performance scalable image compression with EBCOT”. IEEE Transactions on Image Processing, vol. 9, no 7, p. 1158– 1170, juil. 2000.

[43] S.-T. Hsiang. “Embedded image coding using zeroblocks of subband/ wavelet coefficients and context modeling”. Dans Data Compres- sion Conference, p. 83–92. mars 2001.

[44] J. Oliver et M. P. Malumbres. “Fast and efficient spatial scalable image compression using wavelet lower trees”. Dans Data Compression Conference, p. 133–142. mars 2003.

[45] J. Guo, S. Mitra, B. Nutter et T. Karp. “A fast and low complexity image codec based on backward coding of wavelet trees”. Dans Data Compression Conference, p. 293–301. mars 2006.

[46] G. Xie et H. Shen. “Highly scalable, low-complexity image coding using zeroblocks of wavelet coefficients”. IEEE Transactions on Circuits and Systems for Video Technology, vol. 15, no 6, p. 762–770, juin 2005.

[47] S. Saadi, H. Seridi, et A. Benzenache « Codage des régions d'intérêts par l'algorithme SPIHT » Journées Scientifiques sur l'Informatique et ses Applications (JSIA'09)