République **A**lgérienne **D**émocratique et **P**opulaire

Ministère de l'**E**nseignement **S**upérieur et de la **R**echerche **S**cientifique

UNIVERSITE **M**OULOUD **M**AMMERI, **T**IZI-**O**UZOU

FACULTE DE **G**ENIE **E**LECTRIQUE ET D'**I**NFORMATIQUE

DEPARTEMENT D'ELECTRONIQUE

Mémoire de Fin d'Etude de MASTER ACADEMIQUE

Spécialité : **Télécommunication et Réseaux**
Filière : **Génie Electrique**

Présenté par

CHERFA Fahima.

Mémoire dirigé par **Mr LAHDIR Mourad.**

Thème

Compression d'images hyperspectrales en utilisant le toolbox Qccpack.

Mémoire soutenu publiquement le 26 septembre 2017 devant le jury composé de :

Mr ATTAF Youcef	**Maitre de conférences B à l'UMMTO**	**Président**
Mr LAHDIR Mourad	**Maitre de conférences A à l'UMMTO**	**Rapporteur**
Mr IDJERI Boussad	**Maitre assistant A à l'UMMTO**	**Examinateur**

Remerciements

Ce mémoire a été réalisé au sein du Laboratoire d'Analyse et Modélisation des Phénomènes Aléatoires (LAMPA) de la faculté de Génie Electrique et Informatique de l'Université Mouloud Mammeri de Tizi-Ouzou.

Avant tout je tiens à remercier celui qui nous a créé, protégé, aidé, et nous donné la patience et le courage pour pouvoir accomplir entre autre notre mémoire de fin d'études dans les meilleures conditions en disant « Dieu merci ».

Je tiens à exprimer ma gratitude et mes remerciements les plus sincères à Mr. LAHDIR pour avoir accepter d'être rapporteur de mon travail.

Je remercie particulièrement, Nos enseignants pour leur contribution et le temps qu'ils ont bien voulus consacrer pour nous instruire. Qu'ils veuillent bien trouver ici l'expression de ma profonde reconnaissance.

Je remercie également les membres du jury qui me feront l'honneur de juger mon travail.

Je tiens à remercier tous ceux qui m'ont encouragé et ont été la pour moi toujours que ce soit ma famille, mon cher mari et mes amies.

Dédicace

J'ai tout le plaisir de dédier ce modeste travail à :

Ma chère Mère.

Mon cher Père.

Mes adorables frères Ahmed et Khaled.

Ma sœur Zakia.

Mon cher mari.

Toute la famille.

Tous ceux qui je tiens.

Toute la promotion 2017.

Résumé:

L'imagerie hyperspectrale est caractérisée par une forte résolution spectrale qui permet d'acquérir des informations présentes dans l'ensemble du spectre, là où les systèmes classiques de télédétection ne proposent que quelques portions de ce spectre. Ce mode d'acquisition apporte une quantité considérable de données, qui peut rapidement saturer les systèmes conventionnels de transmission et de stockage.

La question fondamentale abordée dans ce mémoire est celle de l'élaboration d'une méthode de compression pour faciliter l'archivage et la transmission des images hyperspectrales avec de forts taux de compression et le minimum de distorsions. Nos travaux se sont déployés autour d'un schéma de compression, utilisant le toolbox Qccpack associée à un codage SPIHT 3D. L'originalité de notre méthode réside dans l'extension de ces algorithmes à la troisième dimension, afin d'exploiter la nature 3D de ces images hyperspectrales caractérisées par une signature spectrale et améliorer ainsi la qualité de compression obtenue pour ce type d'images. Cette nouvelle méthode de compression est basée sur une transformée en ondelettes 3D (TOD3D) qui permet en effet d'exploiter de manière efficace à la fois les corrélations existantes au sein d'un canal (scène spatiale), mais également entre les canaux (scène spectrale) (décorrélelation des trois dimensions du cube hyperspectrale) et d'un codeur à arbres de zéros 3D (SPIHT3D) qui exploite les redondances inter-échelles dans les différentes dimensions des coefficients ondelettes.

L'algorithme ainsi développé a été appliqué à une séquence d'images tests AVIRIS. Les résultats montrent que la qualité du PSNR varie entre 30 et 62.83 dB pour des rapports de compression allant de 0.1 à 1 dB en fonction de l'ondelette utilisée.

Ces résultats montrent que l'exploitation de la troisième dimension (dimension spectrale) améliore sensiblement la qualité de compression.

Mots-clés:

Compression ; Image hyperspectrale; Signature spectrale; Codage; SPIHT3D; JPEG2000; PSNR.

Acronymes et abréviations :

NASA : National Aeronautics and Space Administration.

AVIRIS : Airborne Visible-Infrared Imaging Spectrometer .

AIS : Airborne Imaging Spectrometer.

JPL : Jet Propulsion Laboratory.

CASI : Compact Airborne Spectrometer Imager.

HYDICE : Hyperspectral Digital Collection Experiment.

AISA : Airborne Imaging Spectrometer for applications.

MODIS : Moderate Rsolution Imaging Spectrometer.

NEMO: Naval Earth Map Observer.

SSTI HSI : Small Satellite Technology Initiative Hyperspectral Imager.

QS : Quantification Scalaire.

QV : Quantification Vectorielle.

JPEG : Joint Photographic Experts Group.

ISO : International Standards Organization.

IEC : International Electro-Technical Commission.

TCD : transformée en cosinus discrète.

TOD : transformée en ondelette discrète 2D.

CDF: Cohen Daubechies Fauvaue.

SPIHT: Set Partitionning in Hierarehical Tree.

TO : transformée en ondelettes.

Qccpack: Quantization, Compression, and Coding Library.

GPL: GNU General Public License.

LGPL: GNU Lesser General Public License.

SPECK: Set Partitioning Embedded Block.

IDCT : Transformée en cosinus discrète inverse

FIR : Réponse Impulsionnelle Finie.

MSE : Erreur Quadratique Moyenne.

ASCII: American Standard Code for Information Interchange.

EZW: Embedded Zerotree Wavelet coding.

PSNR: Peak Signal to Noise Ratio.

DWT : discrete wavelet transform.

Table des matières

Tables des figures et des tableaux :

Tables des figures :

Tables des tableaux :

Introduction

L'imagerie hyperspectrale est un domaine en plein essor du fait du développement des technologies numériques. L'élément nouveau apporté par les images hyperspectrales est la signature spectrale. Nous pouvons dire qu'avec les images hyperspectrales, nous sommes effectivement capables d'acquérir des informations présentes dans l'ensemble du spectre, là où les systèmes classiques de télédétection comme ceux d'imagerie multispectrale ne proposent que quelques portions de ce spectre. L'imagerie hyperspectrale permet donc une investigation de plus en plus fine. En revanche, elle apporte des difficultés et de nouveaux défis pour les techniques d'analyse, de traitement et de compression d'image.

La problématique principale de ce type d'image, réside dans la quantité considérable de données générées, qui peut rapidement saturer les systèmes conventionnels de transmission et de stockage. En effet, observer la même scène dans environ 200 longueurs d'ondes, multiplie logiquement la taille des données par 200. Du fait de la taille importante de ces images hyperspectrales, on comprend vite l'intérêt d'une compression efficace pour ce type d'images. Pour ce faire, nous allons exploiter leurs différentes propriétés, notamment la forte corrélation spectrale et la possibilité de les disposer sous forme d'un cube de données et proposer une méthode de compression adaptée.

Nous proposons dans ce mémoire, un algorithme de compression des images hyperspectrales en utilisant le toolbox Qccpack, qui est destiné à être utilisé dans le développement de prototypes des systèmes de codage et de compression.

Pour réaliser ce travail, nous avons divisé notre mémoire en trois chapitres :

Le premier chapitre est un état de l'art, rappelant les propriétés des images hyperspectrales et quelques standards de compression. Après avoir fait le tour de la théorie sur les images hyperspectrale, nous allons proposer et justifier notre méthode et nous allons décrire brièvement les concepts de base des deux algorithmes qui la compose.

Le deuxième chapitre de ce mémoire porte sur les ondelettes qui sont utilisées pour la compression des images hyperspectrales et sur la description du toolbox Qccpack et son installation.

Le dernier chapitre est consacré à l'application de la méthode élaborée sur des images hyperspectrales tout en donnant et évaluant les résultats obtenus. Nous terminons notre mémoire par une conclusion ainsi que par des perspectives ouvertes par ce travail.

I.1. Préambule :

Hyperspectral, compression et contraintes de l'espace, dans ce chapitre nous allons présenter le contexte de cette étude qui porte sur la compression des images hyperspectrales. Dans un premier temps, nous allons donner les notions de base ou bien nous verrons ce que sont les images hyperspectrales, la façon dont elles sont acquises, les problèmes posés par leur nature ainsi que leurs particularités. Par la suite, nous rappellerons les notions de base nécessaires pour comprendre les principes de la compression des images.

I.2. Introduction à l'imagerie hyperspectrale :

I.2.1. La télédétection :

La télédétection se définit comme étant l'ensemble des techniques et des propriétés physiques en particulier leurs propriétés optiques qui permettent l'acquisition d'images, d'obtenir des informations sur des objets et des phénomènes observés par l'intermédiaire d'instruments de mesure (capteurs), sans contact direct avec les objets détectés. La physique de la télédétection précise que chaque matériau présente une réflectivité spectrale unique. Cette réflectivité spectrale aussi appelée signature spectrale. Elle utilise les propriétés d'émission ou de réflexion des ondes électromagnétiques par les objets et les phénomènes.

Les capteurs multispectraux fournissent une résolution spatiale suffisamment intéressante. Cependant leur résolution spectrale ne contient pas assez de détails spectraux des objets. Les capteurs hyperspectraux représentent une révolution importante pour identifier les objets dans une scène.

La télédétection mesure le rayonnement réfléchi par les objets observés et permet de déterminer certaines propriétés de ceux-ci. Elle utilise le même principe que la vision humaine : exploiter la couleur et parfois la texture pour identifier ces objets, mais son champs d'analyse est élargi à des parties du spectre électromagnétique allant au-delà de la lumière visible.
On donne comme exemple, les capteurs utilisés par le satellite SPOT 5, peuvent mesurer le rayonnement réfléchi dans le domaine visible (de 0.50 à 0.68 μm) et dans le domaine proche et moyen infrarouge (de 0,78 à 0,89 μm et de 1.58 à 1.75 μm).

Le satellite d'observation de la Terre SPOT 5, lancé en mai 2002, permet l'acquisition d'images multispectrales avec quatre bandes spectrales. Les bandes B1 à B3 dans le vert, le rouge et le proche infrarouge ont une résolution de 10 m par pixel. La bande B4 dans le moyen infrarouge a une résolution de 20 m par pixel [1].

- Bande 1 : Vert (0,50-0,59 μm).
- Bande 2 : Rouge (0,61-0,68 μm).
- Bande 3 : Proche infrarouge (0,78-0,89 μm).
- Bande 4 : Moyen infrarouge (MIR) (1,58-1,75 μm) à 20 m.

Pour le satellite LANDSAT 7, ses capteurs peuvent mesurer le rayonnement dans le domaine visible, du proche et moyen infrarouge et le rayonnement dans linfrarouge thermique [2].

- Bande 1 : Bleu (0,45-0,52 μm).
- Bande 2 : Vert (0,52-0,60 μm).
- Bande 3 : Rouge (0,63-0,69 μm).
- Bande 4 : Proche infrarouge (0,76-0,90 μm).
- Bande 5 : Moyen infrarouge (1,55-1,75 μm).
- Bande 6 : Infrarouge thermique (10,4-12,5 μm).
- Bande 7 : Moyen infrarouge (2,08-2,35 μm).

Ces capteurs électroniques permettent d'observer la même scène dans plusieurs longueurs d'ondes. Ils sont appelés capteurs multispectraux et les images qu'ils fournissent sont appelées des images multispectrales. Un nombre important de ces capteurs multispectraux se sont développés, le premier étant Landsat au début des années 70 [2].

I.2.2. Principe de l'imagerie hyperspectrale :

Les principes de l'imagerie hyperspectrale dans le cadre de la télédétection satellitaire sont illustrés dans la figure suivante :

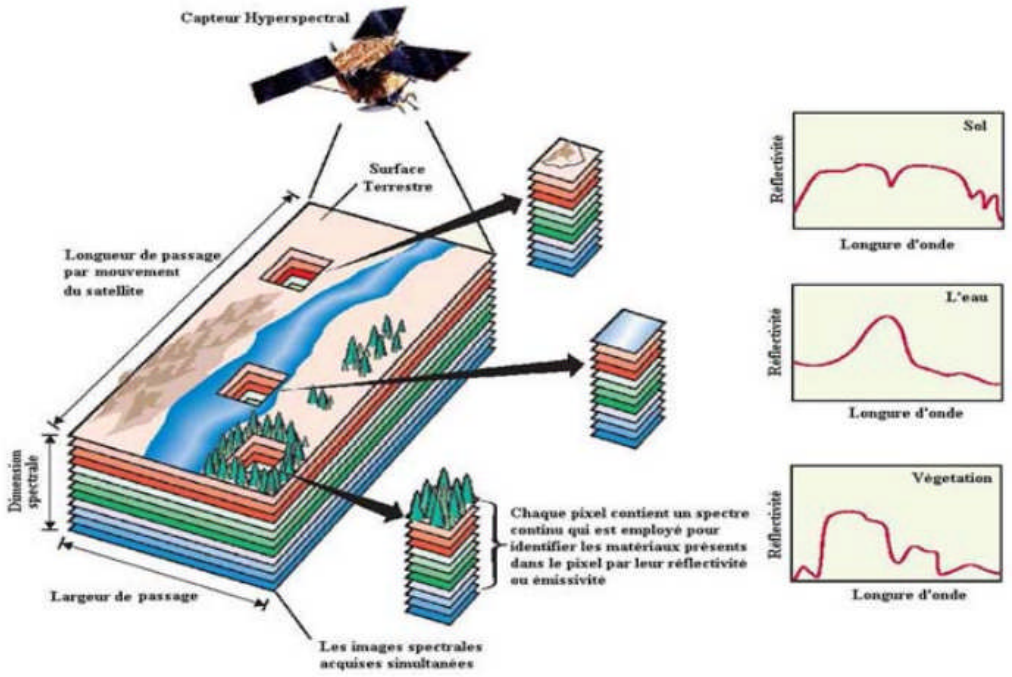

Figure I.1 : Concept d'imagerie hyperspectrale.

L'évolution naturelle des capteurs d'images a conduit à l'acquisition non pas d'une, trois ou quatre bandes spectrales mais plutôt de plusieurs centaines [2] [3].

Le recouvrement spectral va du visible jusqu'à l'infrarouge. L'acquisition de telles données est faite via des capteurs hyperspectraux et cette acquisition aboutit à la formation d'un cube de données comme illustré à la figure suivante, qu'on appelle le cube hyperspectral. En effet, les données acquises ont trois dimensions : deux dimensions spatiales qui représentent la scène et une troisième dimension qui représente le spectre de la scène dans différentes longueurs d'ondes [2] [4].

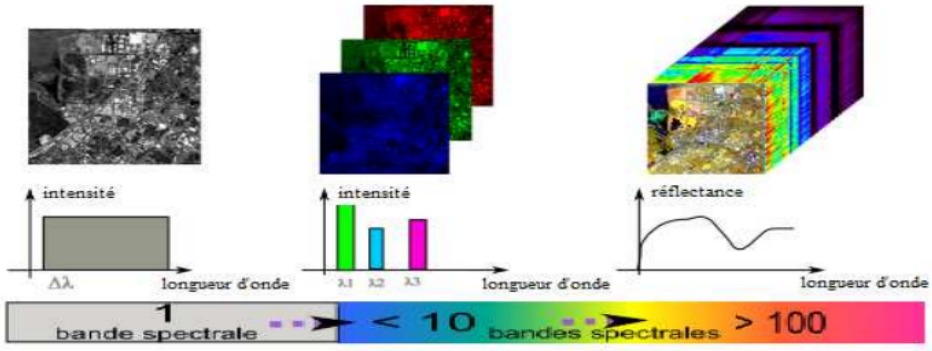

Figure I.2: Du monochrome à l'hyperspectral.

Il existe une grande ressemblance entre les images présentées dans la figure (I.3), mais les différences contiennent beaucoup d'information [3].

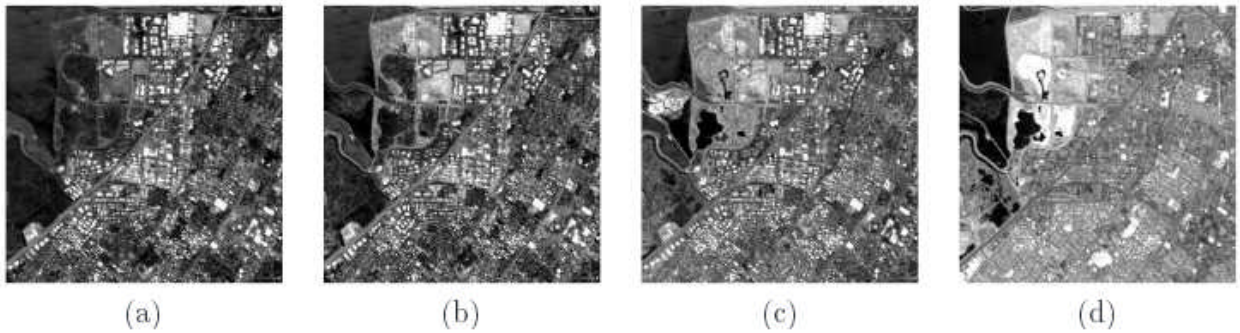

Figure I.3 : Même scène dans diverses bandes spectrales (Moffett Field) à 458 nm (a), 664 nm(b), 712 nm (c) et 1211 nm (d).

L'ajout de bandes spectrales permet d'augmenter le pouvoir discriminant des données acquises (Figure I.3). On peut ainsi arriver à différencier deux matériaux possédant une couleur identique à l'œil. Par exemple, une peinture verte et une feuille qui ont la même couleur, i.e. la même réponse spectrale dans le rouge, le vert et le bleu, ne pourront pas être différenciées à l'œil, l'ajout d'autres bandes spectrales permettront de faire la différence.

Les images hyperspectrales tirent parti de ces propriétés. Sur la figure (I.4), rien ne retient l'attention dans l'observation des données en couleurs naturelles. En revanche, en regardant la bande infrarouge, on remarque immédiatement un point chaud qui est confirmé sur le terrain par la présence de cheminées en activité.

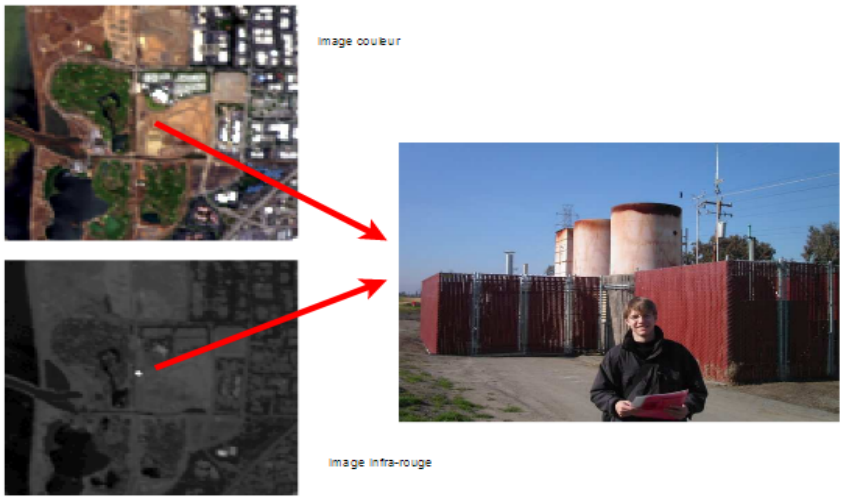

Figure I.4 : L'observation des données dans une bande infrarouge.

L'observation des données dans une bande infrarouge permet de détecter la présence de points chauds qui étaient invisibles dans le domaine visible.

- **La signature spectrale :**

L'élément nouveau apporté par les images hyperspectrales est la signature spectrale. La signature spectrale de chaque objet est une caractéristique parmi l'ensemble des caractéristiques avec lesquelles nous sommes capables d'identifier un objet sur une image. Autrement dit, cette signature correspond à la réaction des objets par rapport au rayonnement solaire (en fonction des longueurs d'onde de ce rayonnement solaire) [2] [5]. Ainsi, nous pouvons discriminer certaines cibles ayant des signatures spectrales très similaires, alors que ces différences spectrales peu perceptibles ne sauraient être observées dans les signatures spectrales acquises par les capteurs à bandes spectrales larges comme celles de l'imagerie multispectrale.

Par exemple, la figure (I.5) comporte deux spectres acquis par deux capteurs multi et hyper spectraux. Evidemment, les informations spectrales acquises par le capteur hyperspectral sont considérables par rapport à celles fournies par de l'imagerie multispectrale. En principe, ces informations spectrales obtenues peuvent être utilisées pour caractériser et identifier un matériau [5] [6]. Donc, cette nouvelle forme de télédétection nous permet de reconnaitre la plupart des objets des images.

Cette technique d'acquisition simultanée des données d'imagerie en quelques centaines de canaux spectraux contigus, nous permet d'enregistrer la signature spectrale pour chaque pixel de l'image comme dans le laboratoire. La différence par rapport aux images multispectrales tient donc au nombre important de bandes (100 à 200), à leur largeur fine (10 à 20 nm) et au fait qu'elles soient contiguës et régulièrement espacées permettant d'obtenir un spectre quasi-continu pour chaque pixel.

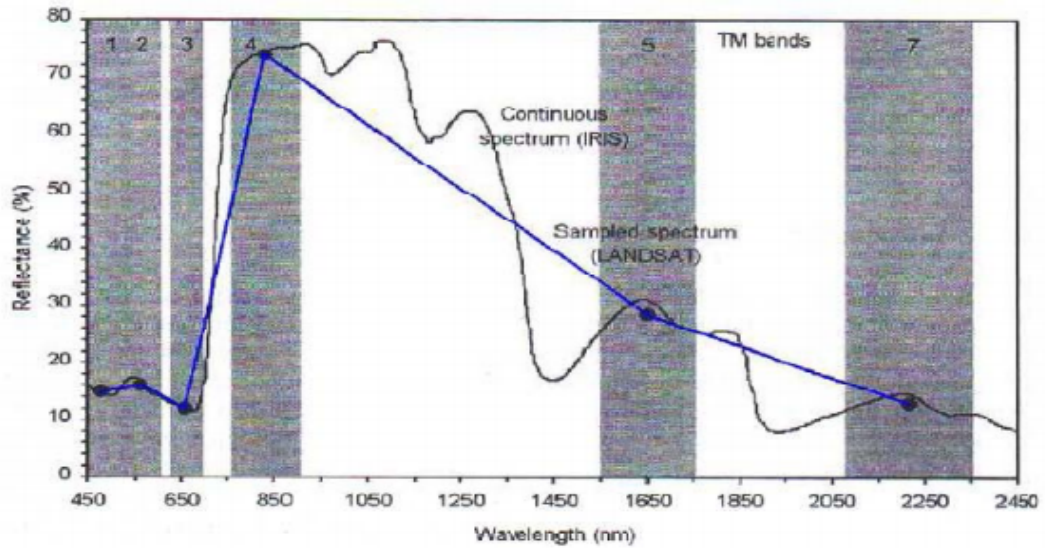

Figure I.5 : Deux spectres fournis par deux capteurs multi et hyperspectraux.

La réflectivité spectrale est le pourcentage de lumière réfléchie par la surface d'un matériau. Elle est unique pour les objets naturels ou artificiels contenant un seul matériau. La réflectivité spectrale est donc un moyen idéal pour l'identification des matériaux, et c'est aussi l'objectif principal de l'application et du développement de la télédétection.

Puisque la réflectivité de chaque matériau est unique, certains chercheurs préfèrent l'appeler la signature spectrale. Les signatures spectrales comportent généralement les détails et les informations nécessaires pour quantifier et qualifier les matériaux existant dans le milieu. La signature spectrale peut être considérée comme une fonction continue de longueur d'onde qui est mesurable dans les laboratoires ainsi que dans les milieux naturels (voir Figure I.6) [5].

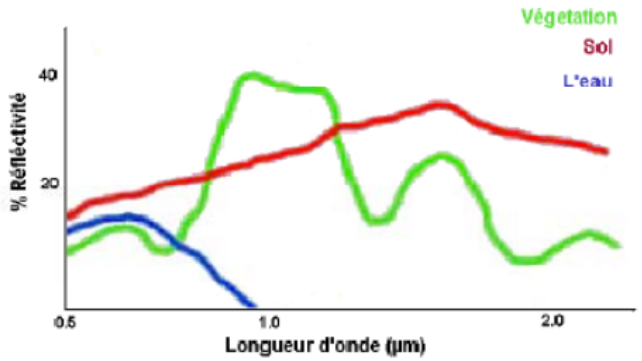

Figure I.6 : La réflectivité spectrale pour trois matériaux distincts.

La mesure et l'analyse des signatures spectrales pour différents types et catégories de matériaux nous permettent de constituer des bibliothèques spectrales. Une bibliothèque spectrale est une collection de réflectivités spectrales de matériaux courants.

I.2.3. Domaines applicatifs :

Les images HS fournissent donc une information plus détaillée des propriétés spectrales d'une scène et permettent d'accéder à une identification et une discrimination plus précises des objets imagés que les capteurs multispectraux.

De fait, l'émergence depuis le milieu des années 1990 de cette technologie a permis d'envisager un large spectre d'applications. La miniaturisation très récente des dispositifs d'acquisition d'imagerie HS permet d'accroître encore le champ des applications potentielles. Les principales applications de l'imagerie HS sont :

– La médecine (ex. : extraction du réseau veineux, détection de mélanome).

– Le contrôle de production (ex. : détection de défauts).

– La surveillance de sites industriels (ex. : détection d'émissions polluantes).

– L'agro-alimentaire (ex. : analyse de la qualité).

– L'agriculture (ex. : traitement ciblé des parcelles).

– La défense (ex. : détection de cibles).

Les chercheurs dans ces différents domaines ont ainsi un accès à des informations spatiales et spectrales permettant de caractériser un échantillon de manière rapide et avec une résolution spatiale et spectrale très fine pour analyser une étendue d'échantillons hétérogènes complexes. Toutefois, cette multiplicité des attributs spectraux n'est pas sans poser de problèmes pour le traitement postérieur à l'acquisition [7].

Une image hyperspectrale est obtenue grâce à un spectro-imageur. L'acquisition d'une même scène est réalisée dans plusieurs bandes spectrales. Le développement de la technologie hyperspectrale a permis en 1989, à la NASA de réalise rune avancée majeure en mettant en œuvre le système AVIRIS. L'AVIRIS était un capteur hyperspectral avancé. Il a acquis des données d'imagerie dans 224 canaux spectraux sur une gamme spectrale de 400 à 2500 nm.

Suite à ce programme, d'autres capteurs hyperspectraux ont été développés et rendus opérationnels [2]. Actuellement, deux technologies sont opérationnelles : aéroportées et spatioportées.

I.2.4. Les capteurs d'imagerie hyperspectrale :

I.2.4.1. Les capteurs aéroportés :

Les capteurs aéroportés sont historiquement les premiers capteurs grâce auxquels l'IHS a été appliquée et développée. Les capteurs hyperspectraux aéroportés trouveront toujours des applications, notamment lorsque de très hautes résolutions spatiales (de 1 à 4 m) sont nécessaires ou en cas d'événements à court terme et à durée critique pouvant ne pas coïncider avec les paramètres orbitaux des satellites. Le premier système de capteurs spectrométriques a été développé en 1983 par JPL (Jet Propulsion Laboratory). Le système s'est appelé AIS (Airborne Imaging Spectrometer). Il faisait des acquisitions de données spectrales en 128 canaux de 1.2 à 2.4 μm. Chaque image acquise avait seulement 32 pixels sur une ligne.

A la suite de cette première expérience, JPL a démarré le programme AVIRIS (Airborne Visible-Infrared Imaging Spectrometer) en 1987. Ce capteur était capable d'acquérir des images spectrales en 224 bandes de 0.40 à 2.45 μm avec 512 pixels sur chaque ligne. Différents sites relevant des thèmes géologiques, agricoles, forestiers ont été observés par ce capteur.

Les capteurs hyperspectraux les plus connus sont cités dans le tableau (I.1) :

Capteur	Nom complet	Opérateur/Pays	Bandes	Gamme spectrale (µm)
AVIRIS	Airborne Visible InfraRed Imaging Spectrometer	NASA/JPL USA	224	0.4-2.5
HYDICE	Hyperspectral Digital ImageryCollection Experiment	USA	210	0.4-2.5
CASI	Compact Airborne Spectrometer Imager	Itres Eesearch canada	228	0.1-1.0
AISA	Airborne Imaging Spectrometer	Specim Ltd Finland	228	0.43-1.0
MODIS	Moderate Rsolution Imaging Spectrometer	NASA USA	36	0.41-14.24

Tableau I.1 : Capteurs hyperspectraux aéroportés.

I.2.4.2. Capteurs spatioportés :

Suite à la réussite des capteurs hyperspectraux aéroportés et grâce à l'expérience de la télédétection spatiale, l'idée de monter un capteur hyperspectral sur satellite a suscité l'attention des agences spatiales. Aujourd'hui, plusieurs capteurs hyperspectraux spatiaux sont actifs : Hyperion est à bord du satellite Earth Observing-1 (EO-1). Plusieurs programmes gouvernementaux ont été également développés, parmi lesquels le capteur CASI (Compact Airborne Spectrometer Imager) qui l'un des plus employés [2].

Le tableau présente les capteurs hyperspectraux portés par satellites.

capteur	Nom	Opérateur/Pays	Bandes	Gamme spectrale (µm)
Hyperion	EO-1	NASA/USA	220	0.4-2.5
FTHSI	MightySatII	Air force Research Lab USA	256	0.475-1.05
Orbview-4	Airforce warfighter	Air force Research Lab USA	200	0.4 -2.5
SSTI HSI	Small Satellite Technology Initiative Hyperspectral Imager	TRW Inc. NASA USA	384	0.4 -2.5
NEMO	Naval Earth Map Observer	Office of Naval Research USA	220	0.4 -2.5

Tableau I.2 : Capteurs hyperspectraux spatioportés.

Les capteurs spatioportés permettent d'offrir parfois certains avantages par rapport aux capteurs aéroportés et aussi par rapport aux satellites multispectraux conventionnels :

1. Ils permettent l'acquisition des données hyperspectrales partout dans la planète à un cout peu de valeur pour l'utilisateur.
2. Les données des capteurs spatioportés sont capables de discerner les propriétés physiques et chimiques des objets à la surface de la terre.
3. Les capteurs hyperspectraux spatioportés offrent des données temporelles durant toute l'année [8].

I.2.5. Caractéristiques des images hyperspectrales :

Avant de s'intéresser à la compression des images dans un système d'imagerie, il faut bien comprendre leurs propriétés statistiques. Il existe certaines caractéristiques principales basées sur les différentes capacités du système d'imagerie qui définissent les propriétés de ses images. Il s'agit des caractéristiques spatiales, spectrales et radiométriques des images [3].

I.2.5.1. La résolution spectrale :

La résolution spectrale est définie comme la largeur $\Delta\lambda$ minimum d'un canal spectral. Dans l'IHS, on insiste plutôt sur cette caractéristique de système et d'image, ce qui est légèrement différent de la définition donnée par l'imagerie multispectrale où la résolution spectrale peut être considérée comme le nombre de canaux. Pour l'IHS, la résolution est donc le nombre de canaux spectraux étroits et contigus. Dans ce cas, la largeur de chaque bande est normalement entre 10 et 14 nm [2] [8].

C'est pourquoi, dans le cas ou la gamme spectrale couverte par le capteur est continue, le nombre total de canaux spectraux peut s'obtenir par l'équation (1) :

$$N\ bandes\ = \frac{[\text{Gamme Spectrale}]}{\Delta\lambda}\ \text{.....(1)}$$

Par cette mesure, les capteurs sont classés comme panchromatiques avec une seule bande, multispectraux avec un nombre de bande compris entre 2-20, hyperspectraux pour un nombre de bandes compris entre 20-250 et enfin ultraspectraux avec plus de 250 bandes [2].

I.2.5.2. La résolution spatiale :

La caractéristique spatiale d'une image se décrit par la résolution spatiale. Elle a été définie comme le pouvoir de discrimination de deux objets. En d'autres termes, elle correspond à la taille du plus petit objet identifiable dans l'image. Elle dépend de la taille du détecteur. Cette taille du détecteur, dans le système électro-optique du capteur, détermine un paramètre qui s'appelle le champ de vue instantané. La résolution spatiale est très variable en fonction de l'application de l'imagerie, elle peut aller de quelques dizaines de centimètres à quelques centaines de mètres [2] [8].

La résolution spatiale permet de classer les capteurs en divers classes telles qu'ultra haut (moins d'un mètre), très haut (entre 1 et 4 m), moyen (de 10 à 50 m), bas (de 50 à 250m) [2].

I.2.5.3. La résolution radiométrique :

Le flux de la radiance qui arrive sur chaque détecteur, pour une longueur d'onde spécifique est une valeur analogique. Maintenant une question se pose : comment cette valeur peut-elle être convertie en valeur d'intensité utilisable par les ordinateurs numériques.

La question peut être considérée comme un problème de quantification dans laquelle nous convertissons les valeurs de flux de radiance entre le minimum et le maximum sur une gamme de valeurs discrètes. Ces valeurs discrètes déterminent les niveaux gris de chaque pixel. La résolution radiométrique se mesure normalement en nombre de bits. L'équation (2) définit le nombre N de niveaux de gris par rapport au nombre de bits n :

$$N = 2^n \quad \text{.....(2)}$$

Par exemple, pour un capteur qui a une résolution radiométrique de 10 bits, nous avons des pixels avec des valeurs de gris compris entre 0 et 1023. Avec cette caractéristique qui est normalement élevée dans le cas de l'IHS, il est probable que deux matériaux très similaires apparaissent avec des valeurs légèrement différentes. Bien que cette probabilité soit petite, elle est importante à prendre en considération.

Par ce critère, nous pouvons classer les capteurs en très haut (plus de 12 bits), haut (entre 8 et 12 bits), moyen (entre 6 et 8 bits) et bas (moins de 6 bits) [2] [5] [8].

I.2.5.4. La couverture spectrale :

Il s'agit de l'étendue du spectre électromagnétique qui est couverte par les capteurs, tels qu'ultra violet, visible, infrarouge réfléchissant, infrarouge thermique et les micro-ondes. Pour l'utilisation de l'imagerie hyperspectrale, nous insistons sur les critères les plus importants tels que la gamme spectrale couverte par les capteurs, la résolution spectrale ou le nombre de bandes spectrales [2].

I.2.5.5. Taille des images hyperspectrales :

Les données hyperspectrales sont volumineuses. Observer la même scène dans environ 200 longueurs d'onde multiplie logiquement la taille des données par 200. Le capteur spatial AVIRIS

acquiert une scène de 512*614 pixels dans chacune des 224 bandes spectrales. Comme ces données sont quantifiées sur 12 bits, cela représente environ 134.4 Mb pour une résolution de 20 mètres seulement [2].

Le capteur spatial Hyperion présente en 3 secondes une scène de 7.5 × 19.8 km, ce qui représente 256 × 660 pixels dans chacune des 242 bandes spectrales [3]. Le tableau présente les spécifications typiques pour un capteur spatial.

Spectre	**400-2500 nm**
Résolution spatiale	**20 m**
Résolution spectrale	**10 nm**
Nombre de bandes	**200**
Quantification	**12 bits**

Tableau I.3 : Spécification typique pour un capteur hyperspectral spatial.

I.2.5.6. Des dimensions aux propriétés différentes :

Le cube hyperspectral est un cube avec des dimensions aux propriétés différentes (axe spectral ou spatial), ce qui va introduire des spécificités sur les données. Chaque valeur d'un pixel correspondant à la luminance pour une longueur d'onde donnée, les propriétés de ces valeurs sont différentes selon les directions spectrales et spatiales. Dans les directions spatiales, la corrélation est forte à faible distance et décroit rapidement quand le décalage augmente. Au contraire, la corrélation spectrale est présente pour tout le spectre. Les propriétés statistiques sont donc différentes selon la direction considérée.

I.2.5.7. Concept du cube de donnée :

Les signaux enregistrés par le capteur sont retransmis à la station terrestre par télémétrie dans le cas de l'imagerie satellitaire ou stockés dans le cas de l'imagerie aéroportée.

Au sol, les données acquises sont transformées en image. Considérant les caractéristiques citées ci-dessous, une image hyperspectrale peut être décrite comme un cube de données à trois dimensions, la face supérieure du cube correspond à la scène spatiale, toutes les scènes pour les différentes longueurs d'ondes sont ensuite empilées pour donner le cube hyperspectral [2] [5] de la manière suivante :

- La largeur de ce cube, mesurée en pixels, est liée à la résolution spatiale.
- La longueur du cube mesuré également en pixels, est liée à la résolution spatiale.
- Enfin, la profondeur est le nombre de canaux spectraux et représente la résolution spectrale de l'image.

La figure (I.7) montre un cube de données hyperspectrales du capteur HYPERION sur la ville de Wichita (U.S.A). Pour préciser la position des pixels dans une image hyperspectrale, on notera I(x, y, λ) : la valeur sur la colonne x, la ligne y et dans la bande spectrale λ. Dans chaque cube, les images spectrales peuvent être considérées comme des couches différentes. De cette manière, pour chaque Δλ (Longueur d'ondes) nous avons une image [5].

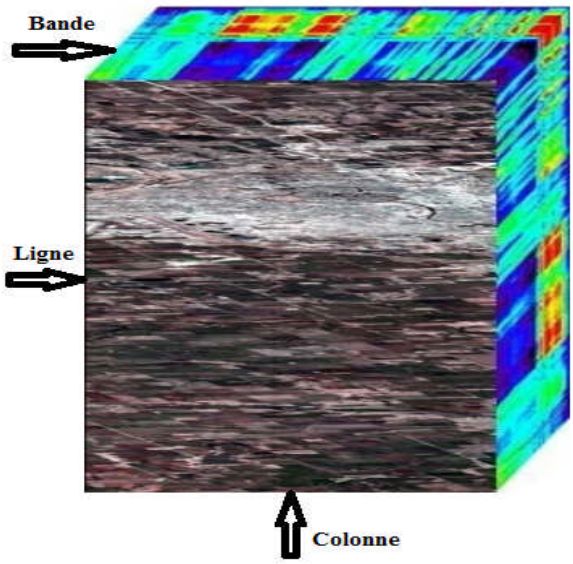

Figure I.7: Un cube de données hyperspectrales d'HYPERION

(256 par 2905 pixels en 220 canaux spectraux).

Vu la taille importante des images hyperspectrales, on comprend vite l'intérêt et la nécessité d'une compression efficace de ces données, pour ce faire nous allons exploiter les différentes propriétés de ces images, notamment la forte corrélation spectrale et la possibilité de disposer ces données sous forme d'un cube de données et proposer une méthode de compression adaptée et exploitant ces propriétés. Dans ce qui suit, nous allons d'abord revoir les notions de base de la compression, en suite nous proposerons notre méthode et la mettrons en œuvre dans les chapitres II et III.

I.3. Notions de base sur la compression d'images :

I.3.1. Notions de base :

La compression d'images est au cœur d'avancement des technologies de l'information. Elle est utilisée dans la majorité des standards de communications pour réduire le nombre de bits nécessaire pour la représentation des données volumineuses telles que les images ou les séquences vidéo.

La compression de données est une opération informatique consistant à transformer une suite de bits à une autre suite de bits plus courte pouvant restituer les mêmes informations en utilisant un algorithme particuliers [9].

Dans cette partie de ce chapitre, nous détaillons les idées essentielles des méthodes de compression. Les méthodes de compression varient suivant les types d'images (naturelles, médicales, satellitaires, etc.) et les applications visées (internet, stockage, etc.) [5]. De plus, les exigences en termes de qualité sont différentes. Il existe deux principales chaînes de compression :

I.3.1.1. La chaîne de compression sans perte (réversible) :

Les valeurs de l'image comprimée ne sont tributaires d'aucune modification par rapport aux valeurs de l'image originale (par exemple, pour les applications médicales : aucune erreur de diagnostic ne peut être tolérée). L'avantage de ce type de chaîne est d'avoir une image reconstruite identique, mais l'inconvénient réside dans le faible taux de compression que l'on peut atteindre. En effet, celui-ci est limité par l'entropie de la source.

Les méthodes sans pertes peuvent être employées directement dans une chaîne de compression. Cependant, certaines d'entre elles sont bien souvent utilisées après la phase de

quantification d'une chaîne de compression avec perte, lors de la transmission ou du stockage des index. Nous pouvons distinguer :

- **Les méthodes prédictives** : celles-ci exploitent la redondance spatiale qui existe entre la valeur courante et les valeurs précédentes ou suivantes.
- **Les codeurs entropiques** : ceux-ci tentent de s'approcher le plus possible de l'entropie de la séquence de valeurs à coder, en affectant un nombre de bits le plus faible possible aux valeurs les plus probables et vice versa. Le codage de Huffman et le codage arithmétique sont les principaux codeurs entropiques utilisés dans le domaine de la compression d'images.

I.3.1.2. La chaîne de compression avec perte (non réversible) :

Lors de la phase de quantification, des modifications sont apportées aux valeurs de l'image. L'avantage de ce type d'approche est qu'il est possible d'atteindre des taux de compression importants, mais au détriment de la qualité de l'image reconstruite. Cependant, la majorité des applications grand public s'est orientée vers ce type de compression (appareil photo numérique, images naturelles, transmission d'images sur les différents réseaux, stockage d'images, etc.).

Les principales étapes d'une chaîne de compression sans perte sont présentées dans la figure (I.8).

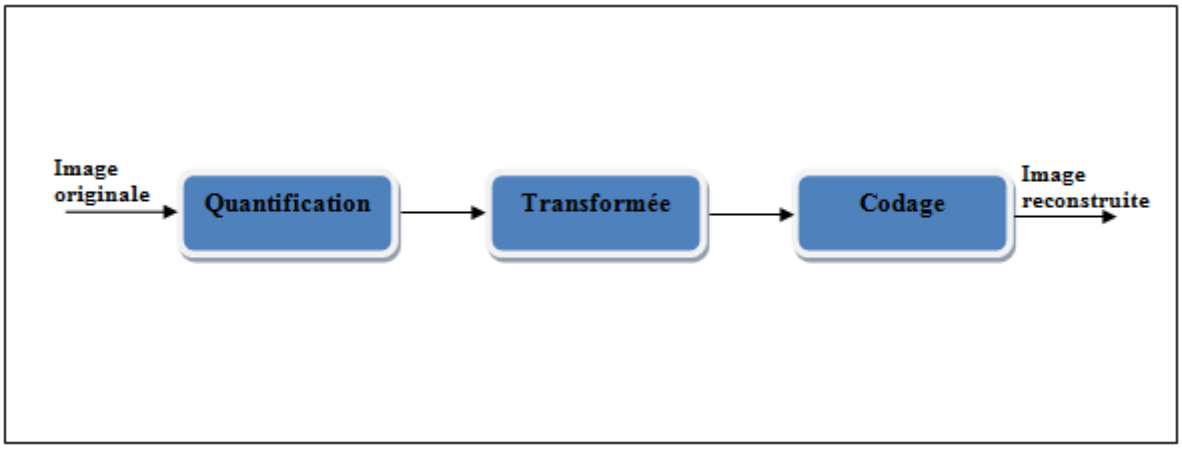

Figure I.8 : Structure générale de compression d'images.

Les méthodes de compression d'images avec pertes constituent la majorité des travaux de recherche, en particulier lors de l'étape de quantification. Nous pouvons distinguer :

- **Les méthodes basées sur la Quantification Scalaire (QS) :** elles consistent à traiter les valeurs (de pixels ou de coefficients) de manière individuelle. Différents types de Quantification Scalaire existent, et sont encore utilisés, par exemple pour le standard JPEG2000.
- **Les méthodes basées sur la Quantification Vectorielle (QV) :** elles traitent en même temps un groupement de pixels ou de coefficients, appelés vecteurs. Elles permettent théoriquement d'être toujours plus performantes que les méthodes basées sur la Quantification Scalaire.

I.3.2. Les trois étapes classiques en compression:

Les méthodes de compression avec pertes d'images actuelles suivent les 3 étapes classiques de compression d'images [9]. Elles débutent pour la plupart par une réorganisation du contenu de l'image, afin de séparer les composantes importantes des composantes contenant peu d'informations. Cette tâche est remplie par une transformation mathématique. Cette étape est suivie par la quantification qui dégrade de manière irréversible le signal puis vient la dernière étape; le codage (sans perte) qui produit un flux binaire.

I.3.2.1. Transformation des données:

Le but de la transformée dans un schéma de compression est double. En effet, en plus de réorganiser l'information, elle doit représenter les composantes importantes d'un signal avec le moins d'éléments possibles : c'est ce qu'on appelle une représentation creuse du signal ou, compacter l'énergie.

Le but d'une transformation est de projeter un signal sur une base de fonctions dont les propriétés sont adaptées à la nature et aux caractéristiques du signal que l'on désire analyser. La projection est généralement orthogonale. Soit X un signal et T_x le signal projeté dans une base de fonction (f_w) w ∈ R, on a :

$$T_x\,(\mathrm{w}) =< \mathrm{X},\, f_w >= \int_{-\infty}^{+\infty} \mathrm{X(t)}\; f_w(t)dt \ldots\ldots\ldots\ldots\ldots (3)$$

La première transformation mathématique employée pour analyser les signaux est la transformée de Fourier (TF). Celle-ci occupe une place centrale, notamment dans le domaine du

traitement du signal, en raison de l'universalité liée au concept de fréquence et de son optimalité pour traiter les signaux stationnaires. En revanche, elle rend difficile l'analyse des signaux dits transitoires car les fonctions sur lesquelles s'effectuent la projection du signal sont définies sur R. Il existe de nombreuses recherches liées à la détermination de différent type de base de projection. Un des points culminants de ces travaux est la théorie des ondelettes qui sera présenté dans le chapitre II.

I.3.2.2. Quantification:

Dans le schéma de compression, l'étape de quantification est celle qui dégrade de manière irréversible le signal. Elle est cependant d'une importance capitale dans la réduction du débit binaire. La quantification est une opération qui transforme un signal continu en un signal discret à l'aide d'un ensemble appelé dictionnaire. Ce passage du continu au discret peut s'effectuer échantillon par échantillon, dans ce cas on parlera de quantification scalaire (QS), ou bloc par bloc : c'est ce qu'on appelle la quantification vectorielle (QV). On définit la fonction de quantification Q appliquée à un échantillon ou un bloc d'échantillons de la source X, le quantificateur Q associe à X le plus proche élément du dictionnaire C. L'objectif, lorsqu'on applique Q à une représentation creuse du signal (c'est-à-dire après transformation du signal) est de diminuer le nombre de bits nécessaires pour coder le signal.

I.3.2.3. Codage:

Il y a deux grandes familles de codeurs sans perte : les codeurs entropiques et les codeurs par plages. Ils sont utilisés dans une chaîne de compression sans perte, directement sur l'image de départ et ils sont également employés à la dernière étape de la chaîne de compression avec pertes (Figure. I.8) afin d'exploiter les redondances présentes à la sortie du quantificateur.

Les codes entropiques sont basés sur la génération de mots dont la longueur dépend de la probabilité d'apparition des symboles de la source qu'il représente : un grand nombre de bits sera utilisé pour coder un symbole peu probable tandis qu'un symbole redondant sera codé sur très peu de bits.

Le codage par plages (Run-length en anglais) consiste à coder la longueur d'une série d'échantillons nuls plutôt que de coder chaque échantillon indépendamment.

Après avoir fait le tour de la théorie de la compression d'images nous, dans ce qui suit, allons voir ce qui se fait en pratique, ensuite nous allons décrire brièvement les concepts de base de deux des algorithmes de compression les plus utilisés dans la compression d'images à savoir : Le standard JPEG et son successeur le JPEG 2000.

L'étude de ces deux standards nous permettra de justifier le choix des méthodes et codes proposés dans notre schéma de compression pour images hyperspectrales.

I.3.3. Du JPEG à JPEG 2000:

JPEG a été établi par l'ISO (International Standards Organization) et l'IEC (International Electro-Technical Commission). JPEG a rencontré un gros succès dans les applications de compression d'images et de transfert d'images. Il utilise une transformée en cosinus discrète 2D (TCD 2D) qui fait passer l'information de l'image du domaine spatial en une représentation identique dans le domaine fréquentiel. Ainsi on parvient à représenter l'intégralité de l'information de l'image sur très peu de coefficients. Récemment, JPEG 2000 a été proposé pour fournir une nouvelle méthode de compression utilisant une transformée en ondelette discrète 2D (TOD 2D). Cette dernière améliore de manière significative la qualité de compression obtenue.

I.3.3.1. JPEG:

En 1992, JPEG établit le premier standard international de compression d'image ou le codage et le décodage sont basés sur la TCD. La TCD transforme un signal d'une représentation spatiale à une représentation fréquentielle. Les basses fréquences sont plus importantes dans une image que les hautes fréquences de ce fait lors d'une TCD, sur une image beaucoup de coefficients hauts fréquences sont éliminées réduisant ainsi la quantité de données nécessaires à la description d'une image sans trop sacrifier la qualité de l'image. Une vue simplifié de la compression TCD consiste :

- Découpage de l'image en block de 8x8 pixels [10].
- Application de la TCD 2D sur chaque block de 8x8 pixels.
- Élimination des coefficients insignifiants pour réduire la taille de l'image.

• Codage des coefficients par un codage entropique comme le codage de Hoffman. Malgré les différents avantages de la compression JPEG qui sont la simplicité, des performances satisfaisantes, il présente de nombreuses limitations. Le besoin d'images compressées de bonnes qualités est en constante augmentation et on a vu que la compression JPEG, à son époque avait fourni des taux records, qui ont permis par exemple le développement d'internet tel que nous le connaissons actuellement. Cependant, la méthode TCD par bloc est désormais un facteur limitant. A partir d'une compression "moyenne", on voit apparaitre des artéfacts de compression, comme les blocs 8x8. Le successeur du standard JPEG, prenant en compte ses limitations.

I.3.3.2. JPEG2000 :

JPEG 2000 va reprendre certains éléments du JPEG comme le prétraitement de l'image, mais ensuite, la compression va se faire sur l'image entière, et non plus sur des blocs réguliers. Cela permettra d'avoir une approche plus globale de la compression par l'usage d'ondelettes. On remplace ainsi la TCD (Transformée en Cosinus Discrète) par la TOD (Transformée en Ondelettes Discrète) [5]. Actuellement, JPEG2000 fournit une compression supérieure d'environ 20% à JPEG pour des faibles taux (peu de détérioration de l'image originale). Pour des taux de compression élevés, JPEG2000 est très largement meilleurs.

Le principe de l'algorithme consiste à diviser en quatre l'image à chaque itération: trois blocs correspondants aux détails de l'image, et le quatrième correspond aux informations les plus importantes pour l'œil (les basses fréquences), qui sert de base pour la prochaine itération. Pour décomposer cette image, on utilise donc deux filtres issus du choix d'ondelette : un filtre passe-haut et un filtre passe-bas. Seules deux types de transformations s'appuyant sur deux types d'ondelettes ont été choisis pour le format JPEG2000 :

• La transformation ``CDF 9/7" pour Cohen-Daubechies-Fauvaue dans le cas d'une transformation irréversible.

• La transformation ``spline 5/3" de Le Gall, beaucoup plus simple pour permettre une transformation réversible.

I.3.3.3. Performance de la TCD et de la TOD:

Les performances de la compression JPEG (utilisant la TCD 2D) et de JPEG 2000 (utilisant la TOD 2D) appliquées à l'image LENA 512x512 codé sur un octet (8 bits) [11]

montrent que pour un rapport de compression inferieur à 25 :1 la compression JPEG est meilleure que celle du codeur à ondelettes et que à partir d'un rapport de compression de 30 :1 la compression JPEG se détériore rapidement pendant que celle avec ondelettes se dégrade linéairement jusqu'à un facteur de 100 :1.

La figure suivante montre la qualité de compression obtenue par JPEG (b) et JPEG 2000 (c) pour l'image LENA (a) pour un rapport de compression de 40:1.

Figure I.9: (a) Image LENA; (b) compression par JPEG; (c) compression par JPEG 2000.

I.3.3.4. Discussion:

Le codeur TCD donne de très bons résultats pour de faibles taux de compression, pour des taux élevés la qualité de l'image se dégrade rapidement alors que le codeur TOD augmente sensiblement la qualité des images pour ces taux élevés de compression.

I.4. Démarche générale du schéma de compression proposé :

Notre schéma de compression s'inscrit dans le cadre d'une nouvelle méthode de compression, à codage SPIHT 3D et à transformée en ondelettes (TO), qui est un outil largement utilisé en traitement du signal et d'image. Sa capacité à compacter l'énergie sur un petit nombre de coefficients permet un codage efficace de l'image. Elle correspond à la première étape de la chaîne de compression classique et elle est couramment utilisée dans le cas des images 2D par application de filtres séparables dans les deux directions (axe x et y). La forte corrélation "fréquentielle" en imagerie hyperspectrale, nous a conduit à étendre cette

transformée à la troisième dimension (axe z) pour décorréler les trois dimensions (x, y et z). Cette transformée qui se nomme transformée en ondelettes 3D (TOD), est maintenant admise comme référence en compression d'images volumiques [2].

L'objectif principal de ce mémoire est de présenter une nouvelle méthode de compression d'images hyperspectrales basée sur l'utilisation du toolbox Qccpack, qui est destiné à être utilisé dans le développement de prototypes des systèmes de codage et de compression. Et présenter aussi les ondelettes, de développer une transformée en ondelettes [8] et de faire l'extension 3D de cette transformée ainsi que son implémentation sous Matlab. Pour la compression d'images, la transformée représente le premier maillon de la chaîne, afin de comprimer l'information, nous allons compléter le cycle par une quantification et un codage.

Le second objectif de ce mémoire est de présenter les différentes façons de coder les coefficients d'ondelettes. Les schémas de codage proposés ici, utilisent un modèle simple pour caractériser les dépendances parmi les coefficients d'ondelettes localisées dans les sous-bandes ayant la même orientation. Les modèles sont basés sur l'hypothèse des arbres de zéros [2] [10], ces méthodes appliquent une quantification par approximation successive pour améliorer la précision de la représentation des coefficients d'ondelettes. Nous choisirons en suite de faire l'extension 3D et l'implémentation de la variante la plus populaire de ce type de codage qui est le SPIHT (Set Partitionning in Hierarehical Tree).

SPIHT produit directement des symboles binaires. Ainsi, un codeur entropique n'est pas obligatoire même s'il est souvent implanté [2]. L'implémentation d'un codeur entropique n'améliore pas les résultats de manière significative, nous avons donc décidé de ne pas l'implémenter afin de ne pas augmenter la complexité de l'algorithme élaboré.

La figure (I.10) présente notre schéma de compression/décompression développé pour les images hyperspectrales 3D.

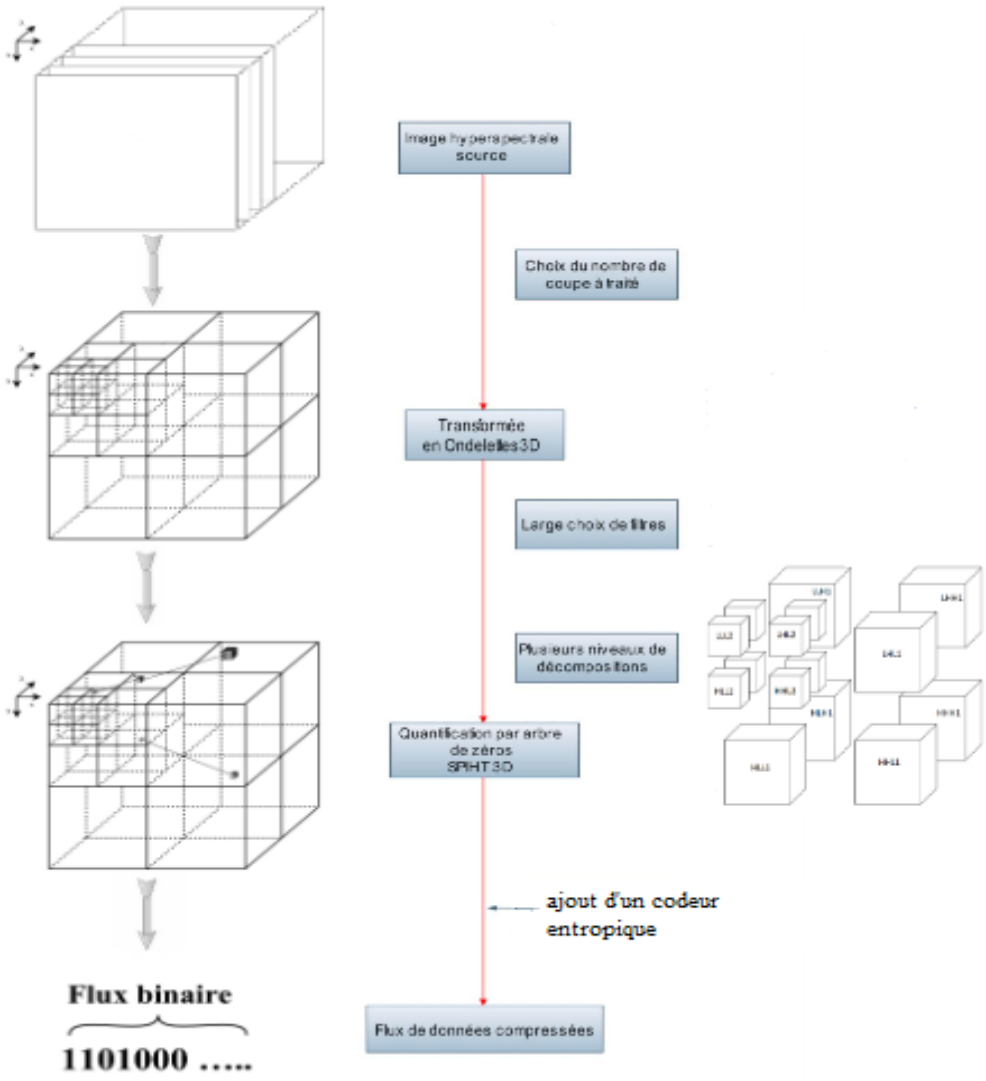

Figure I.10 : Le schéma de compression pour images hyperspectrales.

Notre schéma se compose de deux étapes: une transformation (TOD 3D) et un codage (SPIHT 3D).

I.4.1. Transformation TOD 3D:

Nous proposons de développer une transformée en ondelettes 3D basée sur l'algorithme de Mallat [2], qui consiste en une décomposition pyramidale et réversible d'une image en un jeu de huit sous-images. La décomposition s'appuie sur les filtres passe-haut et passe-bas discrets déduits de l'ondelette et de la fonction échelle associée.

Nous allons aussi définir un large choix de filtres utilisables, tel que les filtres de Daubechies d'autres filtres qui peuvent être utilisés par la transformation développée. Cependant l'utilisation du filtre CDF 9/7 ainsi que CDF pour la transformation en x et y dans le JPEG2000 a montré ses performances. D'autre sauteurs ont testé les performances de compression en appliquant des filtres différents dans la direction z, leur étude montre que le filtre CDF 9/7 est bien adapté quand la distance inter-canal est faible, ce qui est le cas dans les images hyperspectrales (bandes contigües), alors que le filtre de Haar est mieux indiqué pour une distance inter-canal plus importante. Pour ces raisons nous avons donc choisi d'utiliser le filtre CDF 9/7 ainsi que le filtre CDF 5/3 pour tester les performances de notre algorithme.

De plus cette opération peut être appliquée autant de fois que le nombre de décompositions souhaitées. Dans la transformée dyadique, chaque nouvelle décomposition est obtenue à partir d'une nouvelle transformée en ondelettes 3D sur la sous bande basse-fréquence LLL. La figure suivante illustre les blocs de données 3D obtenus par une transformée dyadique sur 2 niveaux.

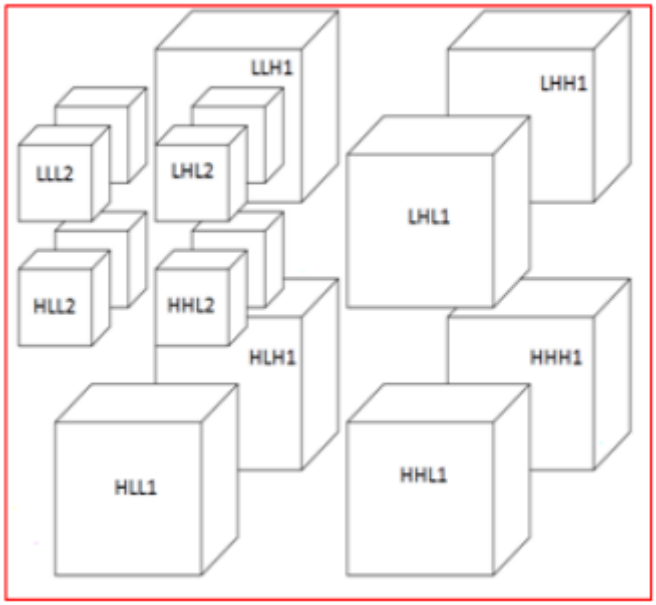

Figure I.11: Blocs de données 3D obtenus par une transformée sur 2 niveaux.

I.4.2. Codage SPIHT 3D :

La deuxième étape consiste à convertir l'image transformée en une série de symboles favorables à un encodage de plus faible volume. Plusieurs techniques de conversion existent. L'un des algorithmes les plus performants dans ce domaine est le SPIHT [2], il exploite la structure pyramidale des coefficients ondelettes obtenus par la TO où il existe d'importantes corrélations entre les différentes échelles. Nous avons donc choisi de développer une version 3D du SPIHT afin d'exploiter les dépendances inter-échelles dans trois dimensions au lieu de deux. Pour cela le SPIHT 3Dutilise des arbres tridimensionnels. On isole ainsi des zones vastes de coefficients non significatifs, c'est ce qui permet d'atteindre de bonnes performances de compression.

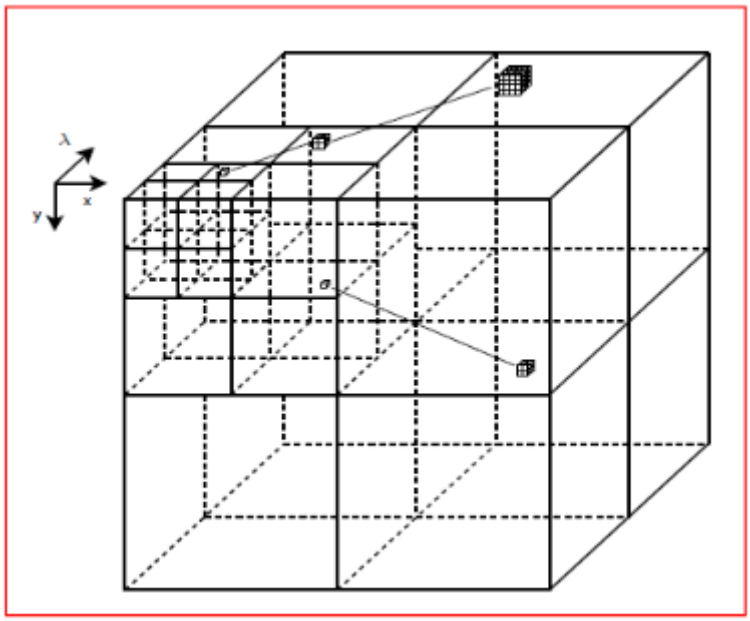

Figure I.12 : Arbres de zéros du SPIHT 3D.

Le codage SPIHT assigne un code binaire à chaque symbole et génère un flux binaire à la sortie de notre chaine de compression.

II.1. Préambule :

Afin d'accélérer l'adoption de la compression des données dans de nouveaux domaines d'application, il est grandement nécessaire de mettre en œuvre des techniques de compression fondamentales à la fois générales, flexibles, robustes et fiables.

Qccpack est destiné à être utilisé dans le développement de prototypes de systèmes de codage et de compression qui étendent la théorie et les algorithmes de compression établis dans de nouvelles zones d'application. Il est également utile dans la recherche universitaire qui emploie ou s'appuie sur des techniques de compression fondamentales.

Sa principale raison d'existence est d'éviter la réinstallation incessante d'algorithmes communs pour des détails spécifiques à l'application.

II.2. Description du toolbox Qccpack :

Qccpack fournit une collection de routines de bibliothèques et des programmes utilitaires pour la quantification, la compression et le codage des données. Qccpack a été écrit pour fournir des implémentations très souples et des procédures générales couramment utilisées dans les applications de codage et de compression.

L'élément essentiel de la collection Qccpack est une bibliothèque de procédures mettant en œuvre une grande variété d'algorithmes de compression et de codage. Les programmes d'application peuvent utiliser des routines de la bibliothèque Qccpack en liant l'application à la bibliothèque lors de la compilation. Chaque fonction de la bibliothèque est très générale dans sa mise en œuvre afin d'être utile dans une grande variété d'applications.

En outre, une grande partie de la fonctionnalité des routines de bibliothèque Qccpack a été fournie sous forme de programmes exécutables autonomes. L'importance primordiale de ces programmes utilitaires est probablement qu'ils fournissent des exemples de la façon d'interagir avec plusieurs routines de la bibliothèque Qccpack. Les programmes de services publics pourraient également être appelés à partir de scripts pour simuler le fonctionnement des systèmes de codage et de compression complexes avant d'implémenter toutes les fonctionnalités du système en un seul programme autonome.

Qccpack est destiné à être utilisé dans le développement de prototypes des systèmes de codage et de compression, et dans la recherche universitaire. Sa principale raison d'être est d'éviter la réimplantation incessante des algorithmes communs pour le bien des détails spécifiques à l'application.

En utilisant Qccpack, nous utilisons la même mise en œuvre du codage arithmétique indépendamment de notre application. L'inconvénient de cette approche de développement basée sur la bibliothèque est le fait que la généralité requise des routines de la bibliothèque entraîne une utilisation fréquente de l'allocation de la mémoire dynamique qui, à son tour, implique que les routines fonctionnent plus lentement qu'une implémentation similaire codée en détails spécifiques à l'application. Pour cette raison, Qccpack est destiné au prototypage et aux besoins de la preuve de concept plutôt que pour une utilisation directe dans les applications nécessitant un fonctionnement à grande vitesse [12].

- **Support multithread :**

Qccpack supporte éventuellement une utilisation dans des applications multithread. À l'heure actuelle, Qccpack ne fait pas usage de fils en interne; actuellement et n'utilise pas les threads en interne ; cependant, si le support multithread est activé lors de la compilation de la bibliothèque, un programme d'application peut invoquer les fonctions Qccpack dans plusieurs threads simultanément.

Pour utiliser Qccpack dans une application multithread, Qccpack doit avoir été compilé avec le support de fil activé et relié à la bibliothèque pthreads [12].

II.2.1.Les modules de Qccpack :

Les routines de la bibliothèque Qccpack et les programmes utilitaires sont répartis en plusieurs modules en fonction de leurs fonctionnalités et ses modules sont répartis en deux types [12] :

II.2.1.1. Les modules standards :

Les modules standards disponibles pour toutes les installations de Qccpack, sont les suivants:

II.2.1.1.1. Le codage entropique :

Le module Qccpack pour le codage entropique est appelé QccPackENT. Il fournit une collection de routines de bibliothèques et de programmes utilitaires pour le codage entropique.
Le codage arithmétique; comprend des modèles adaptatifs de premier et second ordre avec des tailles d'alphabet arbitraires.

II.2.1.1.2. Quantification Scalaire :

Le module Qccpack de quantification scalaire est appelé QccPackSQ. Ce module fournit une collection de routines de bibliothèques et de programmes utilitaires pour la quantification scalaire.

II.2.1.1.3. Quantification vectorielle :

Le Module Qccpack pour la quantification vectorielle est appelé QccPackVQ, ce dernier fournit une collection de routines de bibliothèques et programmes utilitaires pour la quantification vectorielle.

II.2.1.1.4. Adaptive Vector Quantization :

Pour la quantification adaptative de vecteurQccPackAVQ est utilisé, il fournit une collection de routines de bibliothèques et programmes utilitaires pour la quantification adaptative de vecteur.

II.2.1.1.5. Traitement de l'image et de manipulation :

QccPackIMG est le Module de Qccpack de traitement et de manipulation d'image. IL fournit une collection de routines de bibliothèques et programmes utilitaires pour la quantification adaptative de vecteur.

II.2.1.1.6. Vaguelettes :

QccPackWAV vaguelettes est utilisé pour fournir une collection de routines de bibliothèque et des programmes d'utilité pour l'analyse par ondelettes et la synthèse des signaux.

II.2.1.1.7. Le codage de la vidéo :

Le module vidéo QccPackVID fournit une collection de routines de bibliothèques et programmes utilitaires pour le codage vidéo de séquences d'images.

II.2.1.1.8. Imagerie hysperctrale :

Le module QccPackHYP pour l'imagerie hyperspectrale, fournit une collection de routines de bibliothèque et des programmes d'utilité pour l'imagerie hyperspectrale.

II.2.1.2. Les modules optionnels [12] :

En plus des modules standards énumérés ci-dessus, il existe des modules optionnels qui peuvent ne pas avoir été installés sur votre système. Ces modules optionnels peuvent avoir des dispositifs de mise sous licence de GPL / LGPL utilisé pour les modules standards de Qccpack (d'où la raison de les séparer des modules en option)

II.2.1.2.1. SPIHT encodage et décodage d'images :

Le QccPackSPIHT est un module optionnel pour la bibliothèque QccPack qui fournit le codage et le décodage des images à l'aide CLOISONNER Situé dans l'algorithme hiérarchique Arbres (SPIHT).

II.2.1.2.2. SPECK encodage et décodage d'images :

Le QccPackSPECK est un module optionnel pour la bibliothèque Qccpack qui fournit le codage et le décodage d'images en utilisant l'algorithme bloc intégré Set-PARTITIONNEMENT (SPECK).

II.2.2. Module général Qccpack :

Ce module fournit des outils à usage général pour la quantification, la compression et le codage des données :

- **Types de données et format de fichier :**

- **QccFilter** : Structure de données générique **QccFilter** pour le filtrage FIR de signaux.
- **QccBitBuffer** : Structure de données générique **QccBitBuffer** pour l'emballage bitstream / déballage.
- **QccFifo** : Structure de données pour bitstream Fifo.
- **QccDataset**: Structure de données générique vecteur **QccDataset** et correspondant **DAT** format de fichier.

- **QccChannel** : structure pour des symboles du canal correspondant au format de fichier **CHN**.

➢ **Routines de la bibliothèque :**

- **QccInit** : Initialisation de la bibliothèque Qccpack.
- **QccExit** : non-erreur, sortie du programme.
- **QccFree** : Mémoire libre.
- **QccGetProgramName** : Obtient le nom du programme en cours.
- **QccParseParameters** : L'analyse syntaxique de paramètres de ligne de commande.

➢ **Traitement des cordes :**

- **QccStringMakeNull** : Fait une chaîne vide.
- **QccStringNull** : Vérifier une chaîne vide.
- **QccConvertToQccString** : Convertir un tableau de caractères à QccString.
- **QccStringCopy** : Copier QccString.
- **QccStringSprintf** : Imprimer QccString.

➢ **Variables d'environnement :**

- **QccSetEnv** : Définir une variable d'environnement.
- **QccGetEnv**: Récupérer la valeur d'une variable d'environnement.

➢ **Bibliothèque version :**

- **QccSetUserHeader**: message HEADR défini par l'utilisateur.
- **QccGetQccPackVersion** : Récupérer la version et la date de la bibliothèque Qccpack.
- **QccCompareQccPackVersions** : Comparer deux séries de versions Qccpack.
- **QccPrintQccPackVersion** : Imprimer la version et la date de la bibliothèque Qccpack.

➢ **Temps :**

- **QccTimeTic** : Commencer à minuter.
- **QccTimeToc** : Déterminer le temps écoulé.

➢ **La gestion des erreurs :**

- **QccErrorAddMessage** : Ajouter un message à la liste de message d'erreur.
- **QccErrorPrintMessages** : Imprimer tous les messages d'erreur sur stderr.
- **QccErrorClearMessages** : Effacer tous les messages de la file d'attente de messages.
- **QccErrorExit** : Imprimer tous les messages d'erreur à stderr et provoquer la sortie du programme avec l'erreur.
- **QccErrorWarning** : Message d'avertissement d'impression immédiatement à stderr.

➢ **Les valeurs binaires :**

- **QccBinaryCharToInt** : Convertir la valeur char de Qccpack à int.
- **QccBinaryIntToChar** : Convertir la valeur int de Qccpack à char.
- **QccBinaryCharToFloat** : Convertir la valeur char de Qccpack à float.
- **QccBinaryFloatToChar** : Convertir la valeur float de Qccpack à char.

➢ **Accès aux fichiers :**

- **QccFileExists** : Vérifier l'existence de fichiers.
- **QccFileGetExtension** : Obtenir l'extension nom de fichier.
- **QccFileOpen** : Dossier ouvert.
- **QccFileDescriptorOpen** : Pointeur de fichier ouvert du descripteur de fichier.
- **QccFileClose** : Fermer le fichier.
- **QccFileRemove** : Effacer le fichier.
- **QccFileGetSize** : Obtenir la taille du fichier.
- **QccFileGetModTime** : Obtenir la dernière modification du fichier.
- **QccFileGetRealPath** : Obtenir le chemin réel du fichier.
- **QccFileGetCurrentPosition** : Obtenir la position actuelle dans le flux de fichiers.
- **QccFileRewind** : Fichier rewind à son début.
- **QccFileReadChar** : Lire omble.
- **QccFileWriteChar** : Écrire omble.
- **QccFileReadInt** : Lire int.
- **QccFileWriteInt** : Écrire int.
- **QccFileReadDouble** : Lire deux.
- **QccFileWriteDouble** : Écrire à double.
- **QccFileReadString** : Lire la chaîne.
- **QccFileWriteString** : Chaîne d'écriture.
- **QccFileReadLine** : Lire une ligne entière.
- **QccFileSkipWhiteSpace** : Sauter les lignes d'espace et blanc de commentaire.
- **QccFileReadMagicNumber** : Lire les informations de nombre magique et la version d'en-tête de fichier.
- **QccFileWriteMagicNumber** : Ecrire le nombre magique et la version à déposer d'en-tête.
- **QccFileWriteMagicNumberVersion** : Ecrire nombre magique et la version à tête de fichier avec la version explicitement spécifiée.

- **QccFileGetMagicNumber** : Récupérer le numéro magique de fichier désigné.
- **QccFilePrintFileInfo** : Imprimer le nombre magique et la version de stdout.

➢ **Recherche du chemin du fichier :**

- **QccFilePathSearch** : Recherche un nom de fichier spécifié dans un chemin d'accès spécifié
- **QccFilePathSearchOpenRead** : Recherches pour ouvrir et lire un nom de fichier spécifié dans un chemin d'accès spécifié.

➢ **Mathématiques Divers :**

- **QccMathMax** : Maximum de deux nombres.
- **QccMathMin** : Minimum de deux nombres.
- **QccMathPercent** : Pourcentage de deux chiffres.
- **QccMathModulus** : Reste signé de la division de deux nombres.
- **QccMathLog2** : Logarithme en base 2.
- **QccMathMedian** : Médiane des trois valeurs.
- **QccMathRand** : Valeur aléatoire uniformément distribuée.
- **QccMathGaussianDensity** : Fonction gaussienne densité.
- **QccMathLaplacianDensity** : La fonction de densité Laplacien.

➢ **Transparence :**

- **QccAlphaOpaque** : Évaluer l'opacité.
- **QccAlphaTransparent** : Évaluer la transparence.
- **QccAlphaTranslucent** : Évaluer la translucidité.

➢ **Mathématiques vecteur**

- **QccVectorAlloc** : allocation d’un vecteur.
- **QccVectorFree** : Vecteur libre.
- **QccVectorZero** : Vecteur nul.
- **QccVectorResize** : Redimensionner un vecteur.
- **QccVectorMean** : Calculer un vecteur moyen.
- **QccVectorVariance** : Calculer la variance de vecteur.
- **QccVectorAdd** : Ajouter deux vecteurs.
- **QccVectorSubtract** : Soustraire un vecteur d'un autre.
- **QccVectorScalarMult** : Multiplier par vecteur scalaire.
- **QccVectorCopy** : Copie d’un vecteur.
- **QccVectorNorm** : Calculer la norme du vecteur.

- **QccVectorNormalize** : Normaliser la longueur unitaire d'un vecteur.
- **QccVectorDotProduct** : Calculer le produit scalaire de deux vecteurs.
- **QccVectorAngle** : Calculer l'angle entre les deux vecteurs
- **QccVectorSquareDistance** : Calculer la distance euclidienne carrée entre deux vecteurs.
- **QccVectorSumComponents** : Somme de composants vectoriels.
- **QccVectorMaxValue** : Trouver composante maximale d'un vecteur.
- **QccVectorMinValue** : Trouver le composant minimum d'un vecteur.
- **QccVectorPrint** : Vecteur d'impression stdout.
- **QccVectorSortComponents** : Le tri rapide de composantes de vecteur.
- **QccVectorGetSymbolProbs** : Calculer les probabilités d'une liste de symboles.
- **QccVectorMoveComponentToFront** : Déplacer la composant spécifié à l'avant de vecteur.
- **QccVectorSubsample** : Sous-échantillonner un vecteur par un facteur de 2.
- **QccVectorUpsample** : Échantillonner un vecteur par un facteur de 2.
- **QccVectorDCT** : Une dimension de transformée cosinus discrète (DCT) d'un vecteur signal.
- **QccVectorInverseDCT** : Une dimension d'une transformée cosinus discrète inverse (IDCT) d'un vecteur signal.

➢ **Mathématiques Entier-vecteur** :

- **QccVectorIntAlloc** : Allouer vecteur entier.
- **QccVectorIntFree** : Libérer vecteur entier.
- **QccVectorIntZero** : Zéro vecteur entier.
- **QccVectorIntResize** : Redimensionner vecteur entier.
- **QccVectorIntMean** : Calculer du moyen entier d'un vecteur.
- **QccVectorIntVariance** : Calculer la variance du vecteur entier.
- **QccVectorIntAdd** : Ajouter deux vecteurs entiers.
- **QccVectorIntSubtract** : Soustraire un vecteur de nombre entier d'un autre.
- **QccVectorIntScalarMult** : Multiplier vecteur entier par scalaire.
- **QccVectorIntCopy** : Nombre entier de vecteur de copie.
- **QccVectorIntNorm** : Calculer la norme du vecteur entier.
- **QccVectorIntDotProduct** : Calculer le produit scalaire de deux vecteurs d'entiers.

- **QccVectorIntSquareDistance** : Calculer la distance euclidienne au carré entre deux vecteurs de nombres entiers.
- **QccVectorIntSumComponents** : Somme des composants de vecteur entier.
- **QccVectorIntMaxValue** : Trouver composante maximale d'un vecteur d'entiers.
- **QccVectorIntMinValue** : Trouver composant au moins un vecteur de nombre entier.
- **QccVectorIntPrint** : Imprimer un vecteur entier à stdout.
- **QccVectorIntSortComponents** : Le tri rapide des composants de vecteur d'un nombre entier.
- **QccVectorIntMoveComponentToFront** : Déplacer composant spécifié à l'avant du vecteur entier.
- **QccVectorIntSubsample** : Sous-échantillonner un vecteur de nombre entier par un facteur de 2.
- **QccVectorIntUpsample** : Echantillonner un vecteur de nombre entier par un facteur de 2.

➢ **Matrices mathématiques:**

- **QccMatrixAlloc** : Allouer une matrice.
- **QccMatrixFree** : Matrice libre.
- **QccMatrixZero** : Matrice Zéro.
- **QccMatrixResize** : Redimensionner la matrice.
- **QccMatrixCopy** : Copie de matrice.
- **QccMatrixMaxValue** : Trouver le plus grand élément dans la matrice.
- **QccMatrixMinValue** : Trouver plus petit élément dans la matrice.
- **QccMatrixPrint** : Matrice d'impression à stdout.
- **QccMatrixRowExchange** : Lignes de la matrice d'échange.
- **QccMatrixColExchange** : Colonnes de la matrice d'échange
- **QccMatrixIdentity** : Mettre l'identité à une matrice.
- **QccMatrixTranspose** : Transposer la matrice.
- **QccMatrixAdd** : Ajout de deux matrices.
- **QccMatrixSubtract** : Soustraire une matrice d'un autre.
- **QccMatrixMean** : Calculer moyen d'éléments de matrice.
- **QccMatrixVariance** : Calculer la variance des éléments de matrice.
- **QccMatrixMaxSignalPower** : Trouver la plus grande norme carré vecteur-ligne dans la matrice.

- **QccMatrixVectorMultiply** : Multiplier la matrice sur la droite par un vecteur de colonne.
- **QccMatrixMultiply** : Multiplier deux matrices.
- **QccMatrixDCT** : Transformée en cosinus discrète à deux dimensions (DCT) d'une matrice.
- **QccMatrixInverseDCT** : Transformée bidimensionnelle en cosinus discrète inverse (IDCT) d'une matrice.
- **QccMatrixAddNoiseToRegion** : Corrompu région de matrice avec un bruit aléatoire.
- **QccMatrixInverse** : Inverse de la matrice.
- **QccMatrixSVD** : Décomposition en valeurs singulières.
- **QccMatrixOrthogonalize** : Générer une base orthonormée pour la gamme de matrice.
- **QccMatrixNullspace** : Générer la base orthonormale pour un espace nul d'une matrice.

➢ **Mathématiques Entier-matrice :**

- **QccMatrixIntAlloc** : Allouer une matrice entière.
- **QccMatrixIntFree** : Matrice entière libre.
- **QccMatrixIntZero** : matrice entière Zéro.
- **QccMatrixIntCopy** : Copie de matrice entière.
- **QccMatrixIntResize** : Redimensionner une matrice entière.
- **QccMatrixIntMaxValue** : Trouver le plus grand élément entier de la matrice.
- **QccMatrixIntMinValue**: Trouver le plus petit élément entier dans la matrice.
- **QccMatrixIntPrint** : Imprimer la matrice entière.
- **QccMatrixIntTranspose** : Transposer la matrice entière.
- **QccMatrixIntAdd** : Ajoute de deux matrices.
- **QccMatrixIntSubtract**: Soustrait une matrice entière à partir d'un autre.
- **QccMatrixIntMean** : Calculer le moyen entier d'éléments de la matrice.
- **QccMatrixIntVariance**: Calculer la variance des éléments de la matrice entière.
- **QccMatrixIntVectorMultiply**: Multiplier la matrice entière sur la droite par un vecteur de colonne.
- **QccMatrixIntMultiply**: Multiplier deux matrices entières.

➢ **Mathématiques Volume :**

- **QccVolumeAlloc** : Affecter le volume.
- **QccVolumeFree** : Volume libre.
- **QccVolumeZero** : Volume nul.

- **QccVolumeResize** : Redimensionner le volume.
- **QccVolumeCopy** : Copie de volume.
- **QccVolumeMaxValue** : Trouver le plus grand élément de volume.
- **QccVolumeMinValue** : Trouver le plus petit élément de volume.
- **QccVolumePrint** : Volume d'impression à stdout.
- **QccVolumeAdd** : Ajoute de deux volumes.
- **QccVolumeSubtract** : Soustrait d' un volume d'un autre.
- **QccVolumeMean** : Calculer le moyen d'éléments de volume.
- **QccVolumeVariance** : Calculer la variance des éléments de volume.

➢ **Mathématiques-volume entier :**

- **QccVolumeIntAlloc** : Allouer le volume entier.
- **QccVolumeIntFree** : Volume entier libre.
- **QccVolumeIntZero** : volume entier Zéro.
- **QccVolumeIntResize** : Redimensionner le volume entier.
- **QccVolumeIntCopy** : Copie du volume entier.
- **QccVolumeIntMaxValue** : Trouver l'élément le plus grand volume entier.
- **QccVolumeIntMinValue** : Trouver plus petit élément de volume entier.
- **QccVolumeIntPrint** : Imprimer le volume entier à stdout.
- **QccVolumeIntAdd** : ajout de deux volumes.
- **QccVolumeIntSubtract** : soustrait d'un volume entier d'un autre.
- **QccVolumeIntMean** : calculé moyen entier d'éléments de volume.
- **QccVolumeIntMean** : calculer la variance des éléments de volume entier.

➢ **Transformée discrète Cosinus rapide (TCD) :**

- **QccFastDCTInitialize**: Initialisation rapide de la TCD.
- **QccFastDCTCreate** : Créer une TCD rapide d'une certaine longueur.
- **QccFastDCTFree**: TCD rapide et gratuite.
- **QccFastDCTForwardTransform1D**: L'avant de TCD 1D rapide.
- **QccFastDCTInverseTransform1D** : inverse TCD 1D rapide.
- **QccFastDCTForwardTransform2D** : L'avant TCD 2D rapide.
- **QccFastDCTInverseTransform2D** : inverse TCD 2D rapide.

➢ **Maillage régulier 2D :**

- **QccPointPrint** : imprimer le point 2D.

- **QccPointCopy** : Copier Point 2D.
- **QccPointAffineTransform** : affine-transformer le point 2D.
- **QccTrianglePrint**: imprimer triangle 2D.
- **QccTriangleBoundingBox** : trouver la boîte englobante de triangle 2D.
- **QccTrianglePointInside** : déterminer si le point est à l'intérieur triangle 2D.
- **QccTriangleCreateAffineTransform** : transformée crée affine entre deux triangles.
- **QccRegularMeshInitialize** : initialiser maillage régulier 2D.
- **QccRegularMeshAlloc** : allouer maillage régulier 2D.
- **QccRegularMeshFree** : maillage régulier libre 2D.
- **QccRegularMeshGenerate** : générer un maillage régulier 2D uniformément espacées.
- **QccRegularMeshNumTriangles** : calculer le nombre de triangles dans le maillage 2D régulier.
- **QccRegularMeshToTriangles** : extraire des triangles de maillage 2D régulier.

➢ **filtrage des signaux :**

- **QccFilterInitialize** : initialiser le filtre.
- **QccFilterAlloc** : allouer le filtre.
- **QccFilterFree** : filtre libre.
- **QccFilterCopy** : copie du filtre.
- **QccFilterReversal** : retournement temporel de filtre.
- **QccFilterAlternateSignFlip** : changement de signe de remplacement du filtre.
- **QccFilterRead** : lire un filtre de fichier.
- **QccFilterWrite** : écrire un filtre pour déposer.
- **QccFilterPrint** : filtre d'impression.
- **QccFilterVector** : effectuer un filtrage FIR d'un signal à une dimension.
- **QccFilterMultiRateFilterVector** : effectuer un filtrage FIR d'un signal à une dimension en conjonction avec changement de fréquence d'échantillonnage du signal.
- **QccFilterMatrixSeparable** : effectuer un filtrage FIR séparable d'un signal à deux dimensions.

➢ **Liste Bidirectionnelle liés :**

- **QccListInitialize** : initialiser la liste.
- **QccListFreeNode** : nœuds de liste libre.
- **QccListFree** : gratuit tous les nœuds de la liste.

- **QccListCreateNode** : créer un nouveau nœud de liste.
- **QccListCopyNode** : copie du nœud de la liste.
- **QccListCompareNodes** : comparer deux nœuds de liste.
- **QccListFindNode** : trouver un nœud de liste dans une liste.
- **QccListLength** : calculer la longueur d'une liste.
- **QccListAppendNode** : ajouter le nœud à la fin de liste.
- **QccListInsertNode** : insérer un nœud de liste dans la liste.
- **QccListSortedInsertNode** : insérer le nœud de liste dans la liste triée maintien de l'ordre.
- **QccListRemoveNode** : supprimer de nœud de liste de la liste.
- **QccListDeleteNode** : supprimer le nœud de la liste de la liste et le nœud gratuit.
- **QccListMoveNode** : déplacer un nœud de liste à une autre liste.
- **QccListSort** : trier une liste.
- **QccListConcatenate** : concaténer deux listes.
- **QccListPrint** : liste d'impression.

➢ **QccBitBuffer Routines :**

- **QccBitBufferInitialize** : initialiser un tampon de bits.
- **QccBitBufferStart** : commencer à lire ou écrire dans le tampon de bit.
- **QccBitBufferEnd** : mettre fin de lire ou d'écrire dans le tampon de bit.
- **QccBitBufferFlush** : un tampon de bits de rinçage à déposer à la fin d'écriture.
- **QccBitBufferCopy** : copie des bits d'une mémoire tampon dans un autre.
- **QccBitBufferPutBit** : bit de sortie de tampon.
- **QccBitBufferGetBit** : bit d'entrée à partir du tampon.
- **QccBitBufferPutBits** : des bits de sortie au tampon.
- **QccBitBufferGetBits** : les bits d'entrée de tampon.
- **QccBitBufferPutChar** : sortie char buffer.
- **QccBitBufferGetChar** : saisie des caractères du tampon.
- **QccBitBufferPutInt** : int sortie au tampon.
- **QccBitBufferGetInt** : int entrée de tampon.
- **QccBitBufferPutDouble** : double sortie au tampon.
- **QccBitBufferGetDouble** : entrée double du tampon.

➢ **QccFifo Routines :**

- **QccFifoInitialize**: Initialiser FIFO.
- **QccFifoStart**: Début de FIFO.
- **QccFifoEnd**: Fin de FIFO.
- **QccFifoFlush** : Fifo de chasse.
- **QccFifoRestart** : Redémarrage FIFO.

➢ **Routines de QccDataset :**

- **QccDatasetInitialize**: initialiser un ensemble de données.
- **QccDatasetAlloc** : allouer un ensemble de données.
- **QccDatasetFree** : ensemble de données gratuit.
- **QccDatasetGetBlockSize** : obtenir la taille du bloc de jeu de données.
- **QccDatasetPrint** : impression de l'ensemble de données.
- **QccDatasetCopy** : copie de l'ensemble de données.
- **QccDatasetReadWholefile** : lire l'ensemble de ces données.
- **QccDatasetReadHeader** : lire l'en-tête du jeu de données.
- **QccDatasetStartRead** : Lancement de la lecture de jeu de données.
- **QccDatasetEndRead** : fin de lecture du jeu de données.
- **QccDatasetReadBlock** : lire un bloc de jeu de données.
- **QccDatasetReadSlidingBlock** : lire un bloc de jeu coulissant de données.
- **QccDatasetWriteWholefile** : écrire l'ensemble de ces données.
- **QccDatasetWriteHeader** : écrire l'en-tête de jeu de données.
- **QccDatasetStartWrite** : commencer à écrire l'ensemble de données.
- **QccDatasetEndWrite** : écriture d'extrémité du jeu de données.
- **QccDatasetWriteBlock** : écrire un bloc à ensemble de données.
- **QccDatasetSetMaxMinValues** : définir des valeurs maximale et minimale des composantes de vecteur pour l'ensemble de données.
- **QccDatasetMSE** : calculer l'erreur quadratique moyenne (MSE) entre les deux ensembles de données.
- **QccDatasetMeanVector** : calculer la moyenne de jeu de données.
- **QccDatasetCovarianceMatrix** : calculer la matrice de covariance du jeu de données.
- **QccDatasetCalcVectorPowers** : calculer la puissance (norme au carré) des vecteurs d'un ensemble de données.

➢ **QccChannel Routines :**

- **QccChannelInitialize**: initialiser un canal.
- **QccChannelAlloc**: allouer un canal.
- **QccChannelFree**: canal libre.
- **QccChannelGetBlockSize** : obtenir la taille de bloc du canal.
- **QccChannelPrint**: canal d'impression.
- **QccChannelReadHeader** : lire l'en-tête du canal.
- **QccChannelStartRead** : début de lecture du canal.
- **QccChannelEndRead** : fin de lecture du canal.
- **QccChannelReadBlock** : lire un bloc de symboles du canal.
- **QccChannelWriteWholefile** : écrire un canal entier.
- **QccChannelWriteHeader** : écrire en-tête du canal.
- **QccChannelStartWrite** : commencer l'écriture du canal.
- **QccChannelEndWrite** : écriture de fin de chaîne.
- **QccChannelWriteBlock** : écrire un bloc de symboles du canal.
- **QccChannelNormalize** : normaliser canal.
- **QccChannelDenormalize** : dénormaliser un canal.
- **QccChannelGetNumNullSymbols** : compter le nombre de symboles nuls dans le canal.
- **QccChannelRemoveNullSymbols** : supprimer des symboles nuls de canal.
- **QccChannelEntropy** : calculer l'entropie des symboles de canal.
- **QccChannelAddSymbolToChannel** : ajouter un symbole au canal.

II.2.3. Programmes utilitaires :

- **arithmetic_sequence** : séquence arithmétique de nombres de sortie
- **asciitodat** : convertir le fichier d'ASCII au format DAT
- **dattoascii** : convertir des fichiers du format DAT en ASCII.
- **chnentropy** : calculer l'entropie de canal.
- **datcat** : concaténer des fichiers format DAT.
- **datcut** : couper une section d'un format de fichier DAT.
- **datdist** : calculer la distorsion entre les deux fichiers DAT.
- **geometric_sequence** : sortie de séquence de nombres géométrique.
- **printfile** : Imprimer le fichier.
- **engendrer** : Exécute un script shell ou d'un programme en arrière-plan.

II.3.Installation du toolbox Qccpack :

Qccpack peut être installé sur Microsoft Windows en utilisant Cygwin, un logiciel gratuit émulant l'environnement UNIX sous une plate forme Windows. Par conséquent, on doit installer Cygwin avant d'installer le toolbox Qccpack. L'utilisation de l'environnement Cygwin est similaire à tout autre système UNIX.

II.4. Les ondelettes :

La transformation en ondelettes est un outil incontournable dans de nombreuses applications. Elle est bien sûr omniprésente en compression puisqu'elle permet de disposer l'information dans une représentation duale temps- fréquence propice au développement de stratégie de sélection de l'information utile. En outre, son pouvoir de décorrélation et de concentration de l'information a détrôné dès les années 90 la transformée en cosinus (DCT).

II.4.1. L'analyse des ondelettes :

La transformée en ondelettes décompose le signal d'entrée en une série de fonctions d'ondelettes $\Psi_{a,b}(t)$ qui dérivent d'une fonction mère $\Psi(t)$ donnée par des opérations de dilatation et de translation :

$$C_{a,b} = \int_{-\infty}^{+\infty} x(t)\psi_{a,b}(t)dt \quad \text{..... (4)}$$

Grossman et Morlet ont montré que les coefficients d'ondelettes *C* a,b résultant de cette transformation contiennent des informations concernant le signal $x(t)$ étudié à différentes échelles.

$$\psi_{s,\tau} = \frac{1}{\sqrt{s}}\psi\left(\frac{t-\tau}{s}\right) \quad avec\ s \neq 0 \quad \text{..... (5)}$$

Ou :

- Le paramètre $\boldsymbol{\tau}$ est un paramètre de position de l'ondelette.

- Le paramètre s est un paramètre d'échelle qui n'est pas proportionnel à la fréquence mais à son inverse :

- Si s>1 (grand) l'ondelette sera plus étalée et correspondra à une fréquence plus faible, la résolution devient alors bonne en fréquence et mauvaise en temps.
- Quand s<1 (petit), l'ondelette sera plus contractée et correspondra à une fréquence
- Plus élevée que celle de l'ondelette mère. (voir la figure suivante)

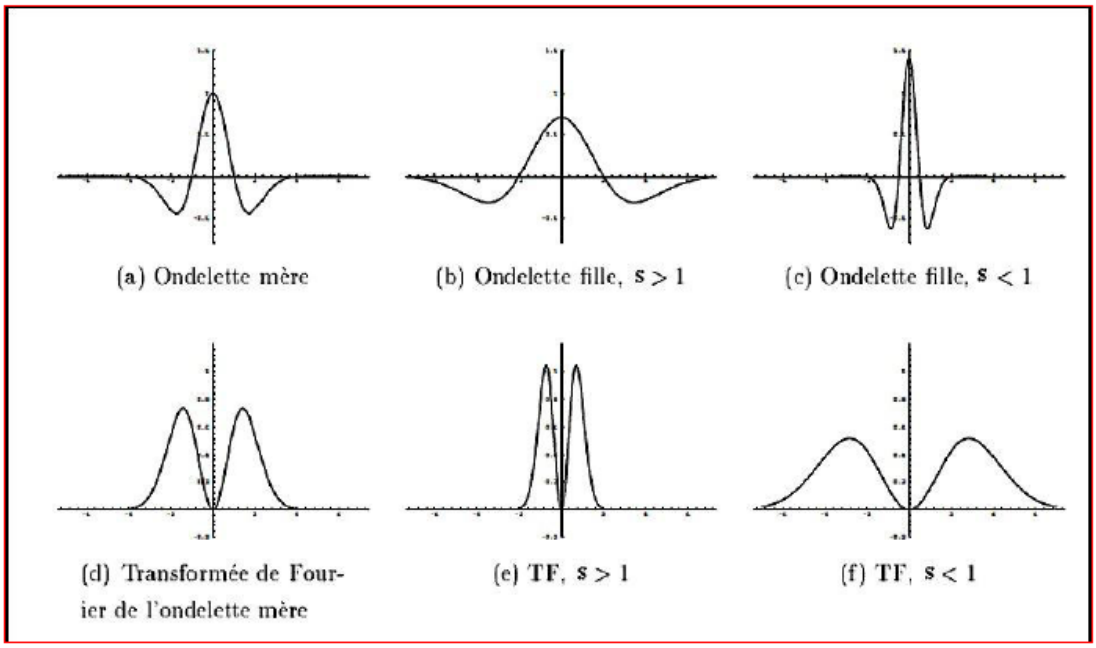

Figure II.1 : Ondelette pour plusieurs facteurs d'échelle.

II.4.2. Transformée en ondelettes continue :

On définit la transformée en ondelettes continue W_f de la fonction f par la formule suivante :

$$Wf(t,s)=\int_R f(\tau)\frac{1}{\sqrt{s}}\overline{\psi\left(\frac{\tau-t}{s}\right)}d\tau \quad \ldots\ldots\ldots\ldots (6)$$

La transformée inverse s'écrit :

$$f(t)=\frac{1}{C_\psi}\iint_{R^2} W(\tau,s)\psi\left(\frac{t-\tau}{s}\right)d\tau\frac{ds}{s^2} \quad \ldots\ldots\ldots\ldots (7)$$

II.4.3. Transformée en ondelette discrète :

Morlet a proposé de construire des bases ou des frames de fonctions construites sur le modèle suivant :

$$g_{t_0,\Delta t}(t) = \frac{1}{\sqrt{\Delta t}}\, g\left(\frac{t-t_0}{\Delta t}\right) \quad \text{..... (8)}$$

Où les valeurs possibles de Δt sont prises sur une échelle géométrique et les paramètres de translation sont proportionnels à Δt.

$$\begin{aligned} \Delta t &= b^j \\ t_0 &= k\Delta t \end{aligned} \quad \text{.....(9)}$$

Une gamme d'échelles Δt couramment utilisée est la gamme des échelles dyadiques 2^j, et on obtient des familles constituées de fonctions de la forme $g_0\left(2^j\left(t-2^{-j}k\right)\right) = g\left(2^j t-k\right)$ ou j et k sont des entiers relatifs. La normalisation la plus couramment utilisée étant une normalisation en norme 2^j on obtient des familles de fonction Où :

$$\psi_{jk}(t) = 2^{j/2}\ \psi_{jk}(2^j\ t\text{-}k) \quad \text{..... (10)}$$

Les différents types des ondelettes :

Il existe de nombreuses méthodes permettant d'effectuer la transformée en ondelettes, chacune se différenciant des autres soit par le type de la fonction de base employée, soit par la méthode d'implantation de la transformée. En fonction de type de traitement que l'on voudra effectuer par la suite on pourra employer l'une des méthodes suivantes :

II.4.3.1. Translation et dilatation dyadiques d'une fonction :

Ce sont les ondelettes classiques .elles sont naturellement reliées à l'analyse multi-résolution et au codage en sous-bandes ;

II.4.3.2. Les paquets d'ondelettes :

Extension des ondelettes classiques, il repose sur des fonctions de base ayant une meilleure localisation fréquentielle au prix d'une transformée légèrement plus lourde.

II.4.4. Les bases trigonométriques locales :

L'idée principale est de travailler aves des sinus et cosinus définis sur des intervalles finis combinées avec une méthode simple mais tés puissante permettant de joindre les fonctions de base à leurs extrémités.

II.4.4.1. Les ondelettes multiples :

L'idée n'est plus d'utiliser une fonction fixe que l'on va translater et dilater mais plutôt un nombre fini de fonctions. Cette méthode permet d'obtenir des combinaisons de propriétés utiles qui seraient impossibles avec les ondelettes classiques ;

II.4.4.2. Le lifting scheme :

On abandonne ici complètement l'idée de translation et de dilatation. Cela fournit une grande flexibilité dans la construction d'ondelettes adaptées à des échantillons irréguliers ou variés

II.3.5.Les familles d'ondelette :

Il existe une infinité de fonctions d'ondelette par ce que oscillante localisée est une ondelette mère possible. Toutefois, elles ne possèdent pas toutes des propriétés intéressantes. Aussi, de nombreux spécialistes des ondelettes ont construit des familles d'ondelettes possédant certaines propriétés remarquables. Parmi ces familles, les ondelettes de Haar sont les plus simples, mais elles ne sont pas bien localisées. Ingrid Daubechies a construit des ondelettes à support compact qui permettent d'utiliser des filtres de taille finie.

Une autre famille d'ondelettes est la famille des splines dont la réponse fréquentielle est bien localisée. Les différentes familles d'ondelettes sont utilisées selon leurs propriétés en fonction du problème à résoudre. La figure II.2 suivante présente quelques familles d'ondelette.

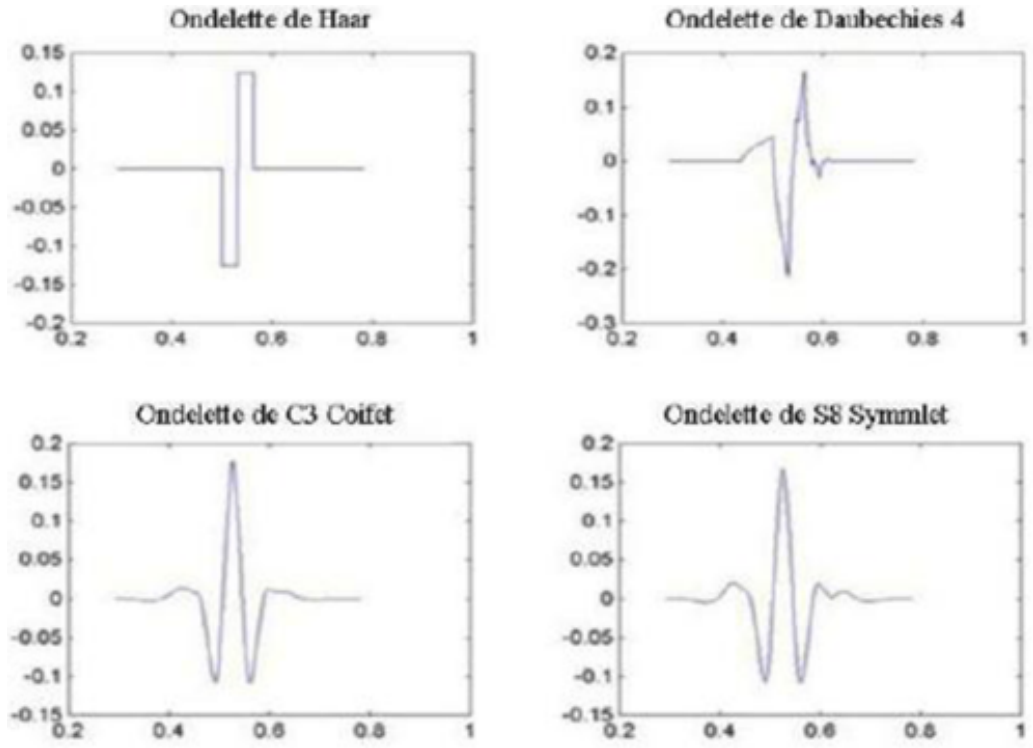

Figure II.2 : Quelques familles d'ondelettes.

II.5. L'algorithme EZW :(Embedded Zerotree Wavelet coding):

Dans une représentation d'image par coefficients d'ondelettes, l'image obtenue est organisée de façon à représenter les principaux traits de l'image dans les bandes de basses fréquences, puis les détails dans les bandes de hautes fréquences. Le principe de l'EZW s'appuie sur cette pour coder les coefficients d'une manière progressive, ainsi, on commence par coder les basses fréquences (approximation), ensuite on code les hautes fréquences (les détails). L'avantage de cet algorithme est que l'on a en tout temps un niveau de compression et que l'on peut arrêter à tout moment le codage.

Son schéma de codage utilise un modèle simple pour caractériser ces dépendances inter-bandes en codant les coefficients d'ondelettes par ordre décroissant, dans plusieurs passages.

Pour chaque passage on choisit un seuil par rapport auquel tous les coefficients d'ondelettes sont comparés. Si un coefficient d'ondelette est supérieur au seuil, il est codé et retiré de l'image, sinon, il est laissé pour le prochain passage. Quand tous les coefficients d'ondelettes ont été examinés, le seuil est abaissé et l'image est remblayée pour ajouter plus de détails à l'image déjà

codée. Ce processus est répété jusqu'à ce que tous les coefficients d'ondelettes soient encodés complètement ou qu'un autre critère soit satisfait.

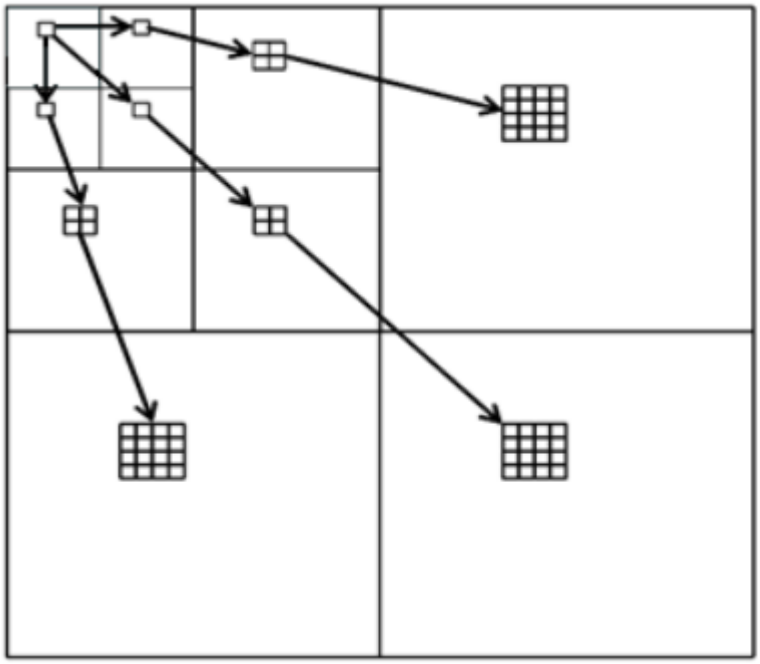

Figure.III.3 : Représentation de l'organisation en arbre des coefficients d'ondelettes.

C’est ici qu'intervient le Zerotree, à travers différentes échelles pour coder efficacement les grandes parties de l'image qui sont au-dessous du seuil actuel.

II.5.1. Définition du Zerotree:

Un Zerotree est un quadruple arbre, dont tous les nœuds enfants sont égaux ou plus petits que les nœuds parents. L'arbre est codé avec un symbole unique et reconstruit par le décodeur comme quadruple-arbre rempli de zéros. Nous devons insister sur le fait que la racine doit être plus petite que le seuil par rapport auquel les coefficients d'ondelettes sont comparés. Sinon, ce coefficient ne serait pas considère comme base de Zerotree.

Le codeur EZW exploite le fait qu'il y a une probabilité très élevée, que tous les coefficients dans un arbre quadruple soient plus petits qu'un certain seuil, si la racine de cet arbre est plus petite que ce seuil. Ceci entraine alors un seul code Zerotree pour tout l'arbre. Ceci dit, en balayant toute la représentation des coefficients d'ondelettes, des basses aux hautes fréquences, on aura automatiquement beaucoup de Zerotree, ce qui constituera un gain considérable au niveau de la compression.

II.5.2. Exemple:

La méthode EZW est illustrée par l'exemple ci-dessous donné par SHAPIRO, le codage a été appliqué sur la matrice de coefficients à trois niveaux de décomposition suivante [9]:

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

Figure II.4 : Exemple de matrice de coefficients.

II.6.L'algorithme SPIHT: (Set Partitioning In Hierarchical Trees):

L'algorithme SPIHT a été proposé par Saïd et Pearlman en 1996 pour la compression d'image 2D avec et sans perte. Cet algorithme repose sur la même idée que celle de Shapiro (EZW) pour caractériser les dépendances entre les coefficients d'ondelettes. Cependant, il utilise les trois principes de base suivant: un rangement partiel par amplitude des coefficients d'ondelettes de la TO2D (résultant de la quantification par approximations successives), un partitionnement dans des arbres hiérarchiques (à chaque seuil appliqué les arbres sont triés sur la base de leur signification en deux catégories d'arbre) et un ordonnancement de la transmission des bits de raffinement (l'amplitude de chaque coefficient significatif est progressivement raffinée).

La différence essentielle entre EZW et SPIHT est la façon dont les coefficients des arbres sont construits, triés et découpés. Ainsi la structure même des arbres de zéros est différente. Dans l'EZW, un arbre de zéros est défini par un coefficient racine et ses descendants ont tous la valeur zéro à l'intérieur d'un plan de bits. SPIHT utilise lui deux types d'arbres de zéros. Le premier

(type A) consiste en une simple racine ayant tous ses descendants à 0 pour un plan de bits donné. Cela diffère un peu des arbres de zéros de l'EZW du fait que la racine elle-même n'a pas besoin d'être non significative.

Le système de liste du SPIHT laisse l'ordre entièrement dépendant des données. Les coefficients sont traités selon leur position dans les listes. La définition des arbres de zéros comme on vient de le voir est sensiblement différente car SPIHT considère deux types d'arbres de zéros : le type A (arbre de degré 1) et le type B (arbre de 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.

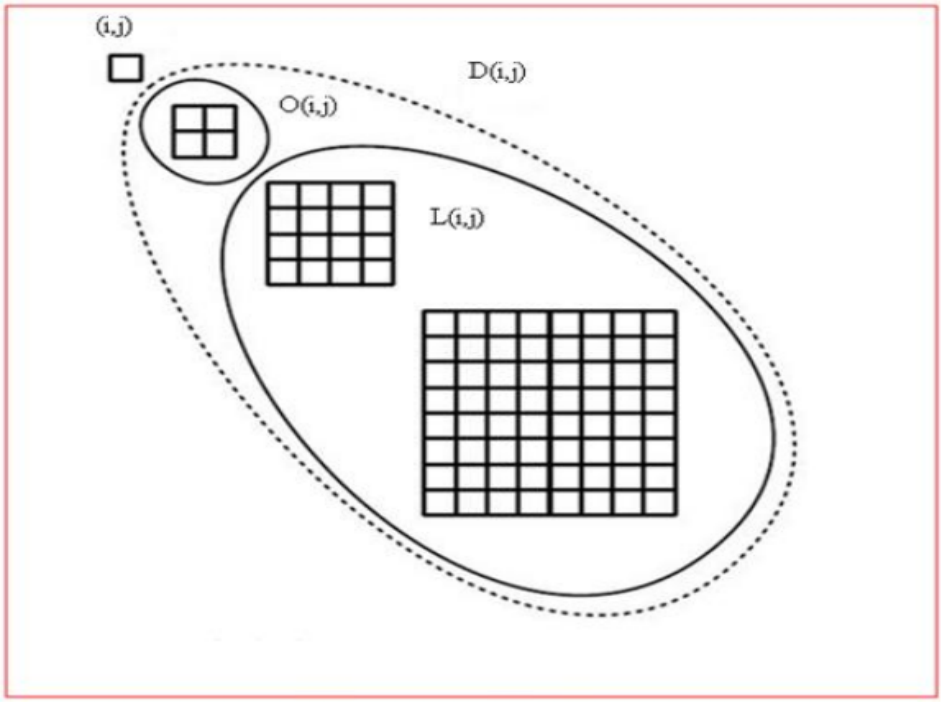

Figure II5 : Terminologie SPIHT pour les descendants.

III.1. Préambule :

Ce chapitre est consacré pour la présentation de schéma de l'algorithme de compression proposé, ainsi que nous allons présenter l'expérimentation et les résultats obtenus. Notre travail consiste à appliquer un algorithme de compression d'images hyperspectrales en utilisant le toolbox Qccpack afin d'atteindre de bons taux de compression pour ce type d'images sans trop altérer leur qualité.

Lorsqu'on utilise des techniques de compression avec perte, l'utilisation de mesure de qualité est indispensable pour l'évaluation des performances. Le problème majeur dans l'évaluation des techniques de compressions avec pertes réside dans la difficulté à décrire la nature et l'importance des dégradations sur l'image reconstruite. Pour évaluer la technique de compression utilisée, le critère d'évaluation PSNR est calculé.

III.2. Algorithme de compression d'images proposé :

Principe : La méthode proposée est basée sur l'utilisation de SPIHT3D Qccpak pour la compression de la séquence initiale. L'algorithme de compression proposée suit un processus de plusieurs étapes, nous l'avons implémenté avec MATLAB (R2013a).

La figure III.1 présente le schéma de l'algorithme de compression proposé :

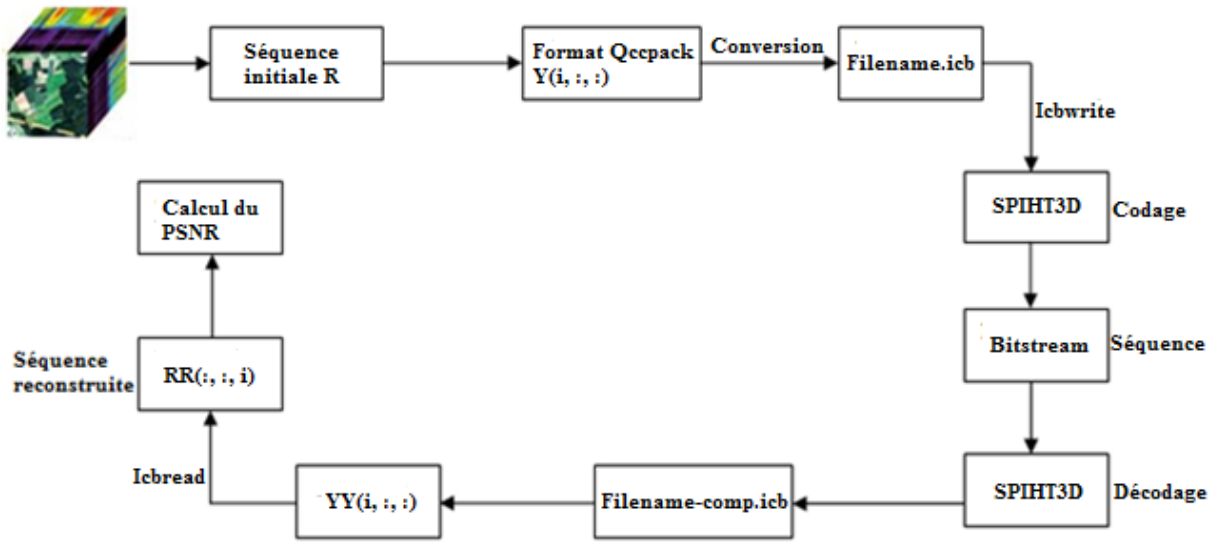

Figure III.1 : Schéma de l'algorithme de compression proposé.

Les étapes de la technique de compression adoptée sont résumées comme suit :

1. Chargement de l'image : cette étape consiste à choisir l'image, sa taille et son affichage pour la comparer avec l'image reconstruite.

2. Ecriture de la séquence initiale « icbwrite » (Conversion au format Qccpack (filename.icb) : dans cette étape le cube d'image (la séquence initiale) est converti au format FILENAME comme un fichier au format ICB Qccpack, elle doit être un tableau de taille num_frames x num_rows x num_cols.

Les valeurs dans la séquence initiale sont exprimées en double avant d'être écrites dans filename en tant que numéros à virgule flottante à 32 bits, car c'est le format requis pour les fichiers ICB.

3. Compression de l'image en appliquant différents filtres :

La compression est réalisée en utilisant l'algorithme Set Partitioning Hierarchical Trees (SPIHT) par Said et Pearlman. L'algorithme SPIHT implique un DWT 2D suivi d'un codage progressif du "bit-plan" des coefficients d'ondelettes en utilisant une structure de quantification zerotree. La transformation est soit la transformée en ondelettes discrète habituelle (DWT), soit une DWT.

Les différents filtres appliqués sont :

- Le filtre cdf9/7.

- Les filtres fbk : Cohen-Daubechies-Feauveau.9-7.fbk', 'Cohen-Daubechies-Feauveau.5-3.fbk', 'coiflet.6.fbk', 'Daubechies.4.fbk','Daubechies.10.fbk','Haar.fbk'};
- Les filtres lft : CohenDaubechiesFeauveau.9-7.lft', 'CohenDaubechiesFeauveau.5-3.lft', 'Daubechies.4.lft'.

Et le format de l'image après compression devient filname_comp.icb.

4. Lecture de format icb « icbread » : dans cette étape le format du fichier QccPack (ICB) spécifié est lu par FILENAME dans la séquence initiale. L'image est renvoyé comme un double tableau de taille num_frames x num_rows x num_cols.

5- Reconstruction de l'image: l'étape de décompression consiste à effectuer les opérations inverses qui ont été accomplis lors de la compression.

III.3. Critères d'évaluation de la compression :

III.3.1 Critères de qualité:

Les méthodes que nous allons utiliser sont basées sur des critères de distances entre l'image originale et l'image après passage dans la chaîne de compression. Dans notre cas, il s'agit de l'image originale sans compression, notée I et de l'image obtenue après compression/décompression dont on veut évaluer les distorsions, notée.

Les images hyperspectrales sont représentées sous la forme d'une matrice tridimensionnelle : I(x, y, λ), x est la position du pixel dans la ligne, y est le numéro de la ligne et λ la bande spectrale considérée. (nx, ny et nλ) sont respectivement le nombre de pixels par ligne, le nombre de lignes et le nombre de bandes spectrales. A l'origine, ces traitements sont adaptés à des données à une seule dimension. Ils ont été ensuite étendus avec succès aux images classiques à deux dimensions.

Pour les images hyperspectrales, des extensions à une troisième dimension ont été réalisées. Dans ce cas, on considère les critères traitant les 3 dimensions (2 spatiales et une spectrale) de la même manière. La spécificité des images hyperspectrales n'est donc pas prise en compte.

Dans le cas hyperspectral, les images peuvent être évaluées avec des critères plus adaptés mais qui sont plus complexes [1], nous utiliserons ici des outils tels que le PSNR, que nous implémenterons sous Matlab.

- **Le PSNR (Peak SNR):**

C'est la mesure de la distorsion entre l'image originale et compressée. Il s'agit de quantifier la performance des codeurs en mesurant la qualité de reconstruction de l'image compressée par rapport à l'image originale. Les valeurs typiques du PSNR pour des images de bonne qualité varient entre 30 et 40 dB.

$$PSNR_{(db)} = 10log_{10}\frac{PeakSignal^2}{MSE} \quad \text{..... (11)}$$

Ou: PeakSignal $=2^q$, q étant le nombre de bits utilisés pour coder les valeurs.

Cette équation du PSNR sera utilisée pour nos calculs, elle traite les 3 dimensions de l'image.

III.3.2. Débit (D) et rapport de compression (RC):

Le débit permet d'estimer le nombre de bits moyens utilisés pour coder un élément. Cet élément peut être un pixel ou un groupement de pixels. Suivant les méthodes de codage, le débit n'est pas toujours un entier. L'objectif est d'obtenir un débit cible Dc, tel que: Dc < Do
(Par exemple, pour une même qualité visuelle de l'image). Pour déterminer le degré de compression obtenu, on définit également : le rapport de compression (R) :

$$RC = = \frac{\text{Nomres de bits de l'image originale}}{\text{Nomres de bits de l'image comprimée}} = \frac{Dc}{D0} \quad \ldots\ldots. (12)$$

III.4.Testes et Résultats de la compression:

Les algorithmes mentionnés ont été mis en œuvre sous Matlab R2013a et les performances ont été évaluées pour différents: niveaux de décomposition, de fonctions d'ondelettes, d'ordre du filtre et de taille de l'image. Le comportement des différentes fonctions ondelettes et leurs caractéristiques ont été étudiées. Trois types de familles d'ondelettes sont utilisés: Daubechies (DbN), et biorthogonales (biorN). Chaque famille d'ondelettes peut être paramétrée par un entier qui détermine l'ordre du filtre (N). La liste de toutes les ondelettes utilisées est :

Famille	**Daubechies.**
Abréviation	**DbN.**
Ordre du filtre utilisé	**N = 4, 6, 10.**
Famille	**Biorthogonales.**
Abréviation	**BiorN**
Ordre du filtre utilisé	**N = 2.2 (LeGaLL5/3) et 4.4 (CDF9/7).**

Tableau.III.1 : Table des filtres utilisés dans les tests.

Dans chaque famille d'ondelettes, nous pouvons trouver une fonction ondelette qui donne la solution optimale associée à l'ordre du filtre, mais cette solution dépend de l'image. Le nombre optimal de décomposition pour chaque ordre de filtre dans chaque famille d'ondelettes sera trouvé. Ce nombre optimal de décomposition donne les PSNR les plus élevées.
Les images 3D utilisées sont produites en combinant plusieurs Canaux 2D. Il est possible de coder ces images 2D de manière indépendante canal par canal (ce que nous pouvons aussi faire

avec l'application développer). Les canaux 2D des images hyperspectrales que nous avons utilisés pour l'algorithme de compression sont de taille 256 × 256 × 1 chacun, chaque image est codée sur 16 bits (après corrections radiométriques); Nous utiliserons pour nos tests une séquence qui contient 224 images, comme nous pouvons faire varier la taille de toute la séquence (dimension de la scène spatiale et dimension de la scène spectrale). Nous rappelons que la taille des images AVRIS est de 512 x 614 x 224 codée sur 16 bit ce qui correspond à 134.32 Mb. La taille de l'ensemble de la séquence que nous avons utilisée est de 512 x 512 x 224.

III.4.1. Influence du type de filtre:

Les tableaux (III.2, III.3 et III.4) montrent les performances de notre algorithme pour le niveau de décomposition J = 3 et pour différentes familles d'ondelettes. Le PSNR change avec les différentes familles d'ondelettes. La façon dont chaque ondelette compacte l'énergie de l'image dépend beaucoup de l'image elle-même (contenu spectral de l'image) de plus chacune d'elles diffère dans la façon de caractérisée ce contenu spectral.

- **Le premier filtre « CDF9/7 » :**

	Débit binaire									
Filtre	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**	**1**
CDF9/7	45.8O5	49.340	52.056	54.098	56.0948	57.5821	58 .9849	60.5516	61.6269	62.8371

Tableau III.2 : PSNR (db) en fonction du débit binaire pour le premier filtre.

- **Le deuxième filtre « FBK » :**

Filtres/Débit binaire	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**
Cdf9/7	43.1695	45.4939	46.1416	46.3749	46.4944				
Cdf5/ 3	37.5060	38.0279	38.1184	38.1444	38.1610				
coiflet	25.6408	25.6242	25.6279	25.6286	25.6278				
Daun4	28.5579	28.4912	28.4842	28.4853	28.4822				
Daub10	26.9478	26.9114	26.9154	26.9173	26.7168				
Haar	29.6118	29.4671	29.4544	29.4472	29.4664				

Tableau III.3 : PSNR (db) en fonction du débit binaire pour le deuxième filtre.

- **Le troisième filtre « LFT » :**

	Débit binaire				
Filtre/Débit binaire	**0.1**	**0.3**	**0.5**	**0.7**	**0.9**
CDF9/7	45.8054	52.0567	56.0948	589849	61.6269
CDF5/3	38.1227	38.7194	38.8244	38.8590	38.8781
Daub4	38.8781	38.8781	38.8781	38.8781	38.8781

Tableau III.4 : PSNR (db) en fonction du débit binaire pour le deuxième filtre.

Interprétation des résultats obtenus :

Nous rappelons que des images de bonnes qualités, le PSNR varie entre 30 et 40 db.

- Pour le premier filtre « CDF9/7 », nous voyons sur le tableau que le PSNR dépasse les 30 db, il varie entre 45,8 et 62,83 pour des débits binaires (db= 0.1 à 1 bpp).
- Pour le deuxième filtre « fbk », nous voyons sur le tableau que les PSNR dépassent les 40 db pour les ondelettes cdf9/7 etcdf5/3 pour des débits binaires (db= 0.1 à 0.9 bpp), et pour les autres ondelettes, le PSNR n'atteint pas 30 db
- Pour le troisième filtre « lft », nous voyons sur le tableau que les PSNR dépassent les 30 db pour les ondelettes cdf9/7 etcdf5/3 et Daub.4 pour des débits binaires (db= 0.1 à 0.9 bpp).

Les figures (III.2, III.3 et III4) montrent l'évolution du PSNR en fonction du débit binaires pour les filtres appliqués et pour différents ordres.

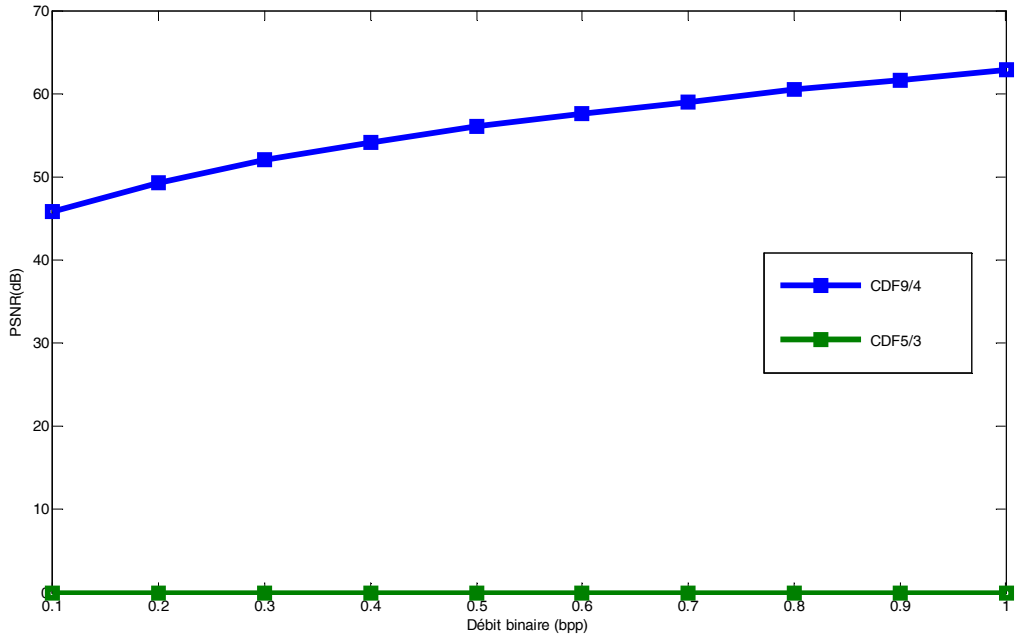

Figure III.2 : PSNR en fonction du débit binaire du filtre.

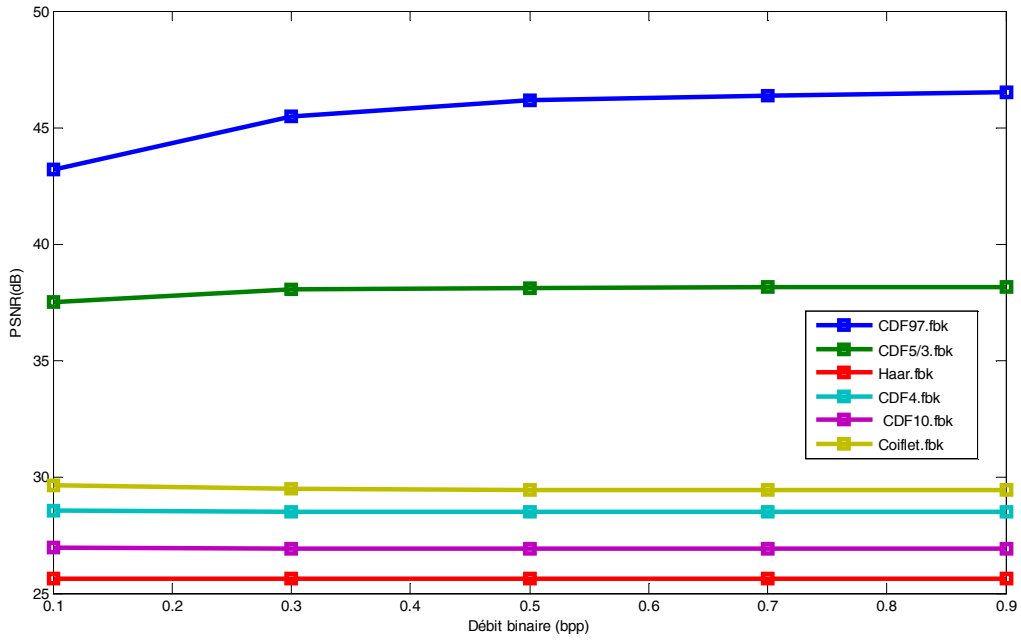

Figure III.3 : PSNR en fonction du débit binaire du filtre FBK .

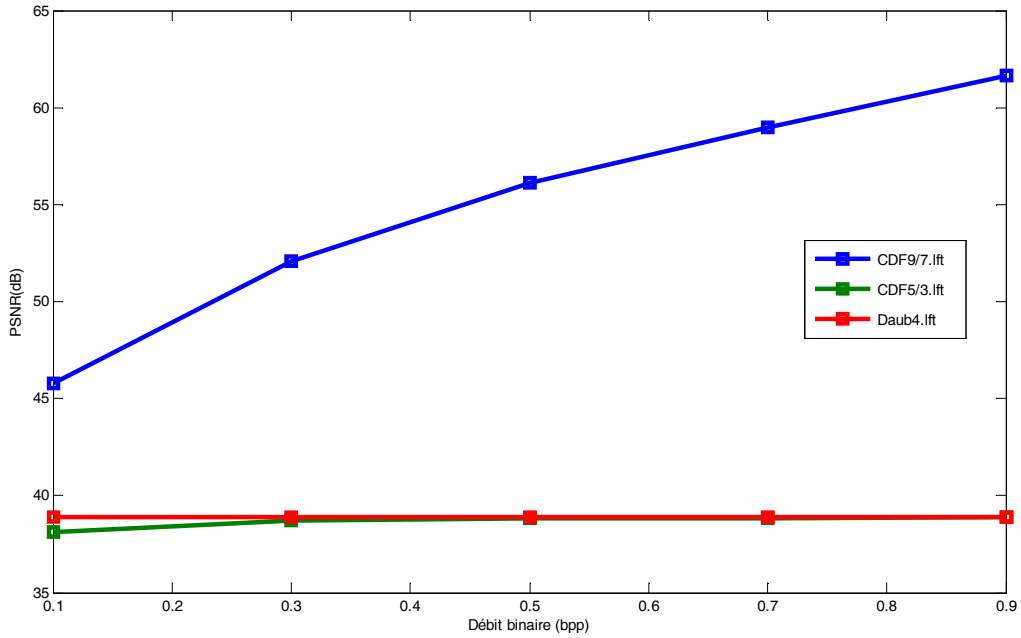

Figure III.4 : PSNR en fonction du débit binaire du filtre LFT.

III.5. Discussion :

L'analyse par ondelettes est un outil très puissant pour la compression d'image hyperspectrale. Les résultats se sont avérés utiles pour comprendre les effets des ondelettes, du niveau de décomposition. Les bonnes performances obtenues par notre algorithme ne sont pas seulement à cause de la supériorité de la transformée en ondelettes, mais aussi pour les caractéristiques du codeur SPIHT 3D. Sa structure en arbres de zéros, permet d'importants gains de codage des cartes de signification.

Les performances de notre algorithme ont été testées en utilisant une séquence de 224 images hyperspectrales et le PSNR a été utilisé comme mesure de qualité. Les résultats ont montrés que l'extension à la troisième dimension et que la compression de cette dimension spectrale améliore la qualité de compression.

Conclusion et Perspectives

Le schéma de compression d'images hyperspectrales que nous avons élaboré et mis en œuvre, est basée sur une transformée en ondelettes, associée à une quantification par arbres de zéros. L'originalité de notre technique, réside dans l'extension de ces algorithmes à la troisième dimension afin d'exploiter la nature 3D de ces images hyperspectrales et d'améliorer la qualité des images reconstruites après décompression.

Dans un premier temps, nous avons fait l'étude théorique et présenté l'intérêt de la transformée en ondelettes 3D (TOD 3D), qui peut compacter l'énergie sur un petit nombre de coefficients. En effet, elle permet d'exploiter de manière efficace à la fois les corrélations existantes au sein d'un canal (scène spatiale), mais également entre les canaux (scène spectrale), notamment entre les canaux (scène spectrale) caractérisée par une forte corrélation (D'où l'intérêt de faire l'extension à cette troisième dimension).
Les images transformées peuvent présenter des propriétés d'autosimilarités à différents niveaux d'échelles, qui sont préservées par la transformée en ondelettes. Par la suite, nous avons proposé un schéma de compression basé sur un codage SPIHT.

Dans un deuxième chapitre, nous avons fait la présentation du toolbox Qccpack et l'ensemble de ses routines de la bibliothèque Qccpack. Nous avons aussi abordé les ondelettes, vu leur efficacité et utilité dans l'obtention des résultats intéressants lors d'une compression d'image.

Et dans le dernier chapitre, nous sommes attachés au cœur de ce travail, c'est- à- dire au développement d'une méthode de compression d'images hyperspectrales. Nous avons présenté l'algorithme de compression et ces différentes étapes jusqu'à la reconstruction de la séquence.

Les résultats que nous avons obtenu montrent que notre technique permet d'avoir un bon compromis entre la qualité des images reconstruites et le débit binaire. En effet, nous avons obtenu un PSNR pour la majorité des filtres appliqués qui varie entre 30 et 62.83 dB pour des débits binaires enter 0.1 et 1 dB. Notre algorithme de compression basé sur les ondelettes TOD 3D et le codeur SPIHT 3D, tire profit de la forte corrélation spectrale et permet d'atteindre un gain de codage important.

A travers nos résultats et ceux publiés dans la littérature, nous pouvons avancer qu'une technique performante de compression pour des images hyperspectrales, passe par une

transformée en ondelettes 3D suivie d'un codeur approprié. Pour permettre, en effet d'exploiter de manière efficace à la fois les corrélations existantes au sein d'un canal, mais également entre les canaux. Nous insistons cependant sur le fait que la théorie de l'information imposant des limites à la compression sans pertes, celle-ci n'offre des taux que très modestes (par rapport aux enjeux), et ce malgré des codeurs de plus en plus sophistiqués. La compression avec pertes à condition bien entendu que celle-ci soit maîtrisée et compatible avec les applications hyperspectrales, apparaît comme la réponse la plus appropriée au problème d'archivage et de transmission des données hyperspectrales, dont les volumes augmentent de façon exponentielle.

Nos résultats prometteurs pour la compression d'images hyperspectrales, nous encouragent à poursuivre les recherches dans ce domaine. Nous pouvons tester notre algorithme pour évaluer l'incidence de la compression avec pertes sur différents traitements classiques sur l'imagerie hyperspectrale (Segmentation, Classification,....). Par ailleurs, nous souhaitons améliorer notre algorithme en lui insérant des fonctionnalités nouvelles. Pour ce faire nous proposons quelques perspectives pour d'autres études ultérieures dans le domaine de la compression d'images hyperspectrales :

- Définition d'une transformée en ondelette optimale pour un bon compromis performances/complexité. Des études ont montré l'intérêt d'une décomposition par ondelettes anisotropique par rapport à une décomposition classique isotropique. Le gain est de l'ordre de 8 dB en termes de PSNR par rapport à une décomposition isotropique classique. On propose d'explorer et de rechercher une décomposition quasi-optimale par ondelettes anisotropique fixe, quelle que soit l'image et le débit visé.
- Ajouter à la bibliothèque du toolbox Qccpack d'autres modules optionnels contenant plus de programmes et d'algorithme d'utilité pour la quantification, la compression et le codage des données.

- Qccpack est destiné seulement au prototypage et aux besoins de la preuve de concept plutôt que pour une utilisation directe dans les applications nécessitant un fonctionnement à grande vitesse vu le lent fonctionnement des routines de cette bibliothèque. Pour cela, il est nécessaire de trouver un système qui va régler ce problème de temps d'exécution de ces applications pour une meilleure utilisation du toolbox Qccpack.

Annexe A : AVIRIS

Le capteur AVIRIS :

AVIRIS est un acronyme pour le Spectromètre à imagerie infrarouge visible à l'air. AVIRIS est un instrument de premier plan dans le domaine de la télédétection terrestre, il a été mis au point en 1987 par la NASA. C'est un capteur optique qui est capable d'acquérir des images spectrales sur 224 bandes avec 512 pixels sur chaque ligne, en utilisant un système d'imagerie numérique à balayage whisk broom. La gamme spectrale couverte par les 224 canaux varie entre 0.4 et 2.45 µm. Cette largeur de gamme est large, elle couvre le visible, le proche infrarouge et le moyen infrarouge.

Le capteur AVIRIS est aéroporté soit en haute altitude soit en basse altitude et sa résolution spatiale dépend de l'altitude de l'avion (entre 4km – 20km). Il a été piloté sur quatre plates-formes d'avions: le jet ER-2 de la NASA, le turbopropulseur Twin Otter International, Proteus de Composites Scaled et la WB-57 de la NASA. L'ER-2 vole à environ 20 km au-dessus du niveau de la mer, à environ 730 km / h. L'avion Twin Otter vole à 4 km au-dessus du niveau du sol à 130 km / h. AVIRIS a volé partout aux Etats –Unis, le canada et l'Europe. Le tableau suivant résume les caractéristiques du capteur :

Caractéristiques du capteur AVIRIS	
Nom	**Airborne Visible Infrared Imaging Spectrometer**
Plateforme	**Aéroporté**
Constructeur	**NASA/JPL**
Nombre de bandes	**224**
Résolution spectrale	**~10 nm**
Game spectrale	**0.4 et 2.45 µm**
Echantillonnage spatial	**20 m × 20 m (à 20 km)** **4 m × 4 m (à 4 km)**
Résolution radiométrique	**12 bits** **16 bits après corrections radiométriques**

Figure A1 : Caractéristiques du capteur AVIRIS.

Annexe A : AVIRIS

Les images AVIRIS:

Le "cube image" montre les données obtenues par l'instrument AVIRIS monté sur un avion ER-2 de la NASA à une altitude variant entre 4000 mètres à 20000 mètres au- dessus de la scène à photographier (voir Figure A2.a).

Ce "cube image" est produit en combinant plusieurs canaux 2D. Le sommet du cube est une image en fausses couleurs. Dans une image satellitaire fausse couleur, le canal Vert est habituellement Représenté en nuance de bleu, le Rouge en nuance de vert et l'infrarouge en rouge. Ainsi, sur l'image satellitaire, la mer apparait en bleu, les nuages en blanc. Le sol, les routes et les zones urbaines apparaissent plutôt en gris, plus ou moins clair selon leur composition. Les côtés du cube sont les bords des 224 canaux spectraux d'AVIRIS. Les hautes coupes sont dans la partie visible du spectre (longueur d'onde de 400 nanomètres), et les basses coupes sont dans l'infrarouge (2500 nanomètres). Les côtés sont des pseudo-couleurs, allant du noir au bleu (faible réponse) et rouge (réponse élevée).

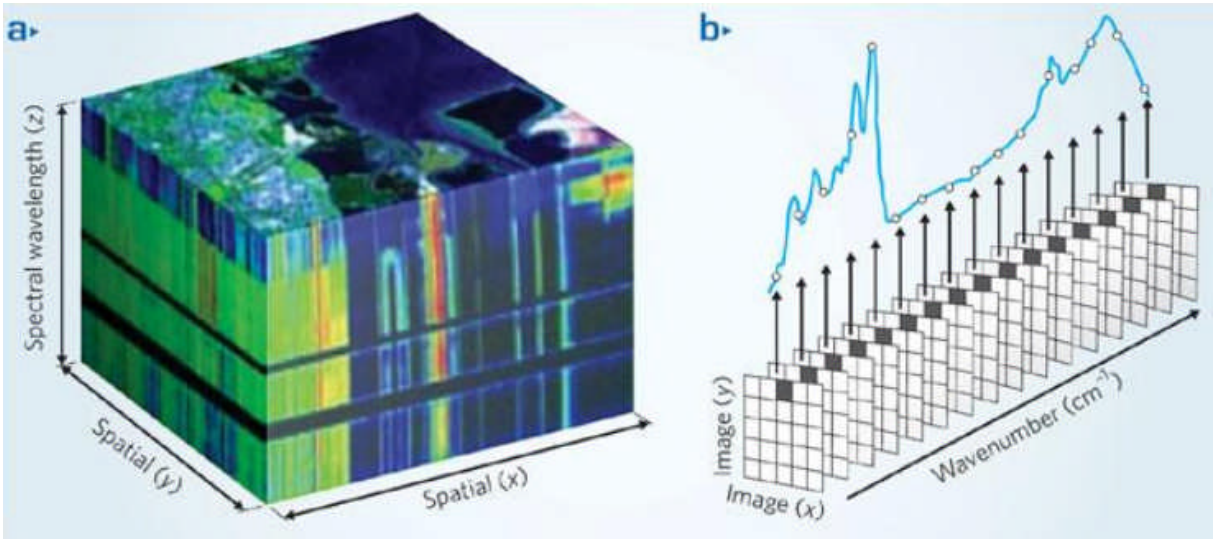

Figure A2: (a) Le cube image des données AVIRIS ; (b) Bandes spectrales contiguës et continues.

La signature spectrale comporte des informations uniques concernant les propriétés physiques, chimiques des matériaux et objets analysés. La résolution spectrale est le paramètre le plus important des images hyperspectrales. L'imagerie hyperspectrale permet d'obtenir des échantillons détaillés des spectres. Pour ce paramètre, l'essentiel tient à la continuité des canaux avec une largeur assez étroite (voir Figure A2.b).

Annexe A : AVIRIS

La possibilité des données hyperspectrales à surmonter les contraintes et les limites des images à faible résolution spectrale et spatiale font d'elle un outil très puissant pour la télédétection. Les images hyperspectrales fournissent des données à haute résolution spectrale, Les capteurs hyperspectraux ont suffisamment de résolution spectrale pour identifier, même de petites quantités de matériaux grâce à leurs signatures spectrales. Dans ces cas, les informations complémentaires fournies par l'imagerie hyperspectrale fournissent souvent des résultats pas possibles avec d'autre système d'imagerie.

Annexe B : Arbre de Zéros

Arbre de Zéros :

Les arbres de zéros pour les coefficients d'ondelettes ont été développés pour exploiter complètement la notion de multi-résolution associée aux ondelettes. Après la transformée en ondelettes, on observe que la localisation des coefficients significatifs est similaire entre les différentes sous-bandes. 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. C'est cette idée qui a été exploitée avec succès par J.Shapiroavec EZW (Embedded Zerotree of Wavelet coefficients) et une amélioration remarquable a été apportée quelques années plus tard par A.Said et W.A.Pearlman avec SPIHT (Set Partitioning In Hierarchical Trees). La définition d'un arbre de zéros varie en fonction des algorithmes.

Y. Choa développé l'idée des arbres de zéros de degré k. Un arbre de degré 0, est un arbre dont tous les coefficients sont égaux à zéro. Un arbre de degré 1, est un arbre dont tous les coefficients sauf a racine sont égaux à zéros et un arbre de degré 2, est un arbre dont tous les coefficients sauf la racine et ses enfants directs sont égaux à zéro. EZW utilise des arbres de degré 0, tandis que SPIHT utilise des arbres de degré 1 et 2. (Voir figures suivantes).

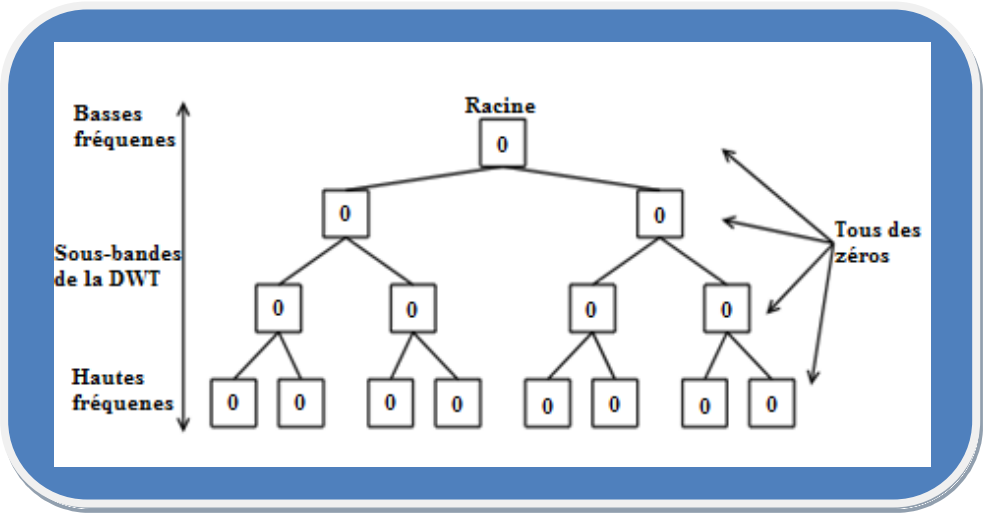

Figure B1: Arbres de zéros de degré 0.

Annexe B : Arbre de Zéros

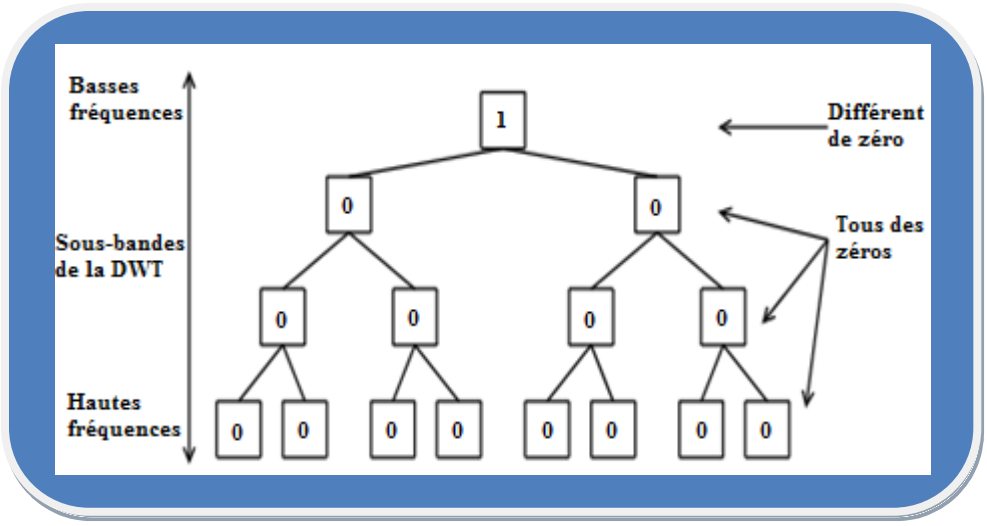

Figure B2: Arbres de zéros de degré 1.

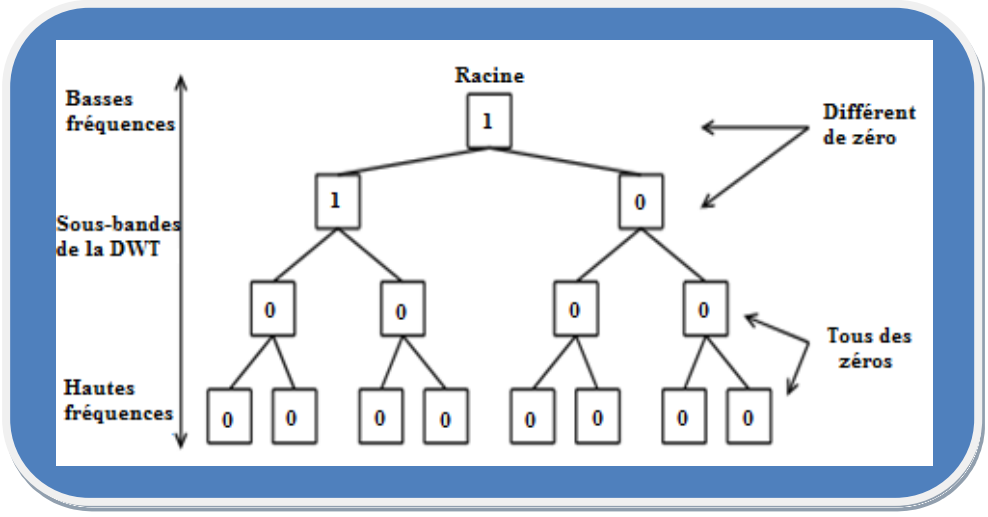

Figure B3: Arbres de zéros de degré 2.

Les méthodes basées sur les arbres de zéros présentent les avantages suivants:

- Corrélations inter-bandes bien exploitées.
- Représentation progressive de l'image.
- Codage avec perte vers codage sans perte.
- Faible complexité.
- Très bonnes performances de compression.

Bibliographie

[1]. DELAUNAY Xavier (2008). **Compression d'images satellite par post-transformées dans le domaine ondelettes.** Thèse de Doctorat, spécialité : Signal, Image, Acoustique et Optimisation, Université de Toulouse, 163 pages.

[2]. OUAHIOUNE Mohand (2011). **Compression des Images Hyperspectrales par la Transformée en Ondelettes 3D.** Mémoire de Magister en Electronique, Option : Télédétection, UMMTO, 134 pages.

[3]. Emmanuel CHRISTOPHE (2006). **Compression des Images Hyperspectrales et son Impact sur la Qualité des Données.** Thèse de Doctorat, spécialité : Signal et Image, école nationale supérieure de l'aéronautique et de l'espace, 192 pages.

[4]. E. Christophe1, D. Léger2, C. Mailhes1, 1 IRIT – TESA, 2 ONERA - DOTA. **Images hyperspectrales et critères qualité**. Tésa 14-16 Port St Etien ne 31000 Toulouse.

[5]. BELKADI Abdellah (2011). **Approche neuronale pour la classification des images hyper spectrale.** Mémoire de Magister en Informatique, Option : Télédétection, Analyse et Traitement Informatique des Données Spatiales, Université des sciences et de la technologie d'Oran Mohamed Boudiaf, 115 pages.

[6]. David Salomon (2004**). Data Compression. The Complete Reference.** Springer.

[7]. Mariem SOLTANI (2014). **Partitionnement des images hyperspectrales de grande dimension spatiale par propagation d'affinité.** Thèse de Doctorat, spécialité : Traitement du Signal et Télécommunications, UNIVERSITE DE RENNES 1, 89 pages.

[8]. ZIKIOU Nadia (2014). **Application d'un apprentissage SVM pour le codage d'images hyperspectrales,** Mémoire de Magister en Electronique, Option : Télédétection, UMMTO, 86 pages.

[9]. BENSAFI Noreddine (2016). **Compression d'images par tenseur 2d : Application aux images MSG.** Mémoire de Master Académique en Electronique, Option : Télécommunication et Réseau, UMMTO, 86 pages.

[10].OUAFI Abdelkrim (2008/2009). **Compression d'images avec pertes par codages imbriqués, Proposition d'une optimisation de l'algorithme EZW.** Thèse de Doctorat,

spécialité : Sciences en Electroniques, UNIVERSITE MOHAMED KHIDER BISKRA, 89 pages.

[11]. Taubman, D. Marcellin, MW. (2002). **JPEG2000 image compression : fondamentals, standards and practice**. Dordrecht : Kluwer Academic Publishers.

[12]. http : //qccpack.sourceforge.net/Documentation.