REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE MOULOUD MAMMERI, TIZI-OUZOU
FACULTE DE GENIE ELECTRIQUE ET DE L'INFORMATIQUE
DEPARTEMENT D'ELECTRONIQUE

MEMOIRE DE MAGISTER EN ELECTRONIQUE

OPTION : ***Télédétection***

Présenté par :

Mr OUAHIOUNE Mohand

THEME :

Compression d'images hyperspetrales par la transformée en ondelettes 3D

Devant les membres du jury :

Président :	Mr. LAGHROUCHE Mourad	Maître de conférences	UMMTO
Rapporteur :	Mr. AMEUR Soltane	Professeur	UMMTO
Examinateurs:	Mr. LAHDIR Mourad	Maître de conférences (B)	UMMTO
	Mr. HADDAB Salah	Maître de conférences	UMMTO
	Mme. AMEUR Zohra	Maître de conférences	UMMTO

Soutenu le : 29 / 09 / 2011

There, I've done my best. If that won't do, I shall have to wait until I can do better.

Victor Heerman, écrivain (1893-1977)

à ma mère, mon frère et ma sœur.

Remerciements :

Ce travail a été réalisé au **L**aboratoire d'**A**nalyse et de **M**odélisation des **P**hénomènes **A**léatoires (LAMPA) de L'UMMTO. Je remercie son directeur, Monsieur AMEUR Soltane pour son accueil.

Je tiens à exprimer toute ma reconnaissance à Monsieur AMEUR Soltane, Professeur à l'UMMTO et directeur du laboratoire de recherche LAMPA, pour avoir accepté d'être rapporteur de ma thèse.

Je remercie également vivement Monsieur LAHDIR Mourad, Maître de conférences (B) à l'UMMTO, pour avoir bien voulu être Co-rapporteur de ma thèse.

Un grand merci à Monsieur LAGHROUCHE Mourad, Maître de conférences à l'UMMTO, pour l'honneur qu'il me fait de présider mon jury de thèse.

Je suis également très reconnaissant à Madame AMEUR Zohra, Maître de conférences à l'UMMTO, pour l'honneur qu'elle me fait de participer à mon jury de thèse.

Je remercie également vivement Monsieur HADDAB Salah, Maître de conférences à l'UMMTO, pour l'honneur qu'il me fait de participer à mon jury de thèse.

Ce travail a été effectué sous la direction de Monsieur AMEUR Soltane et LAHDIR Mourad Professeur et Maître de conférences à l'université de Tizi-Ouzou. Je tiens à les remercier pour leur encadrement et les précieux conseils dont ils m'ont fait bénéficier.

Je tiens à remercier encore une fois LAHDIR Mourad pour sa disponibilité, sa simplicité, son honnêteté intellectuelle et sa gentillesse.

J'adresse également tous mes remerciements à tous ceux qui ont contribué à l'étude et à la rédaction de ce mémoire.

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 une transformée en ondelettes, associée à une quantification par arbres de zéros. 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 forte corrélation 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 (TOD 3D) 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 (SPIHT 3D) 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) de Yellowstone datant de 2006. Les résultats montrent que la qualité du PSNR varie entre 35-55 dB pour des rapports de compression allant de 100 à 10 en fonction de l'ondelette utilisée. Ces résultats montrent que l'exploitation de la troisième dimension (dimension spectrale) améliore sensiblement la qualité de compression. En effet, l'algorithme offre un gain bien supérieur à celui offert par sa version 2D, nos résultats montrent ainsi un gain allant jusqu'à plus de 10 dB pour tous les rapports de compression.

Mots-clés: Compression d'images hyperspectrales; Corrélation spectrale; TOD 3D; SPIHT 3D; JPEG2000; Télédétection.

Table des matières

Chapitre II : Transformée en Ondelettes

Chapitre III : Quantification par Zerotree

Chapitre IV : Tests & Résultats

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.

Dans ce mémoire nous proposons un schéma de compression qui s'inscrit dans le cadre d'une approche 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. La forte corrélation spectrale en imagerie hyperspectrale, nous a conduit à étendre cette transformée à la troisième dimension (λ), pour décorréler les trois dimensions. Cette transformée en ondelettes correspond à la première étape de la chaîne de compression classique, elle sera suivie par des algorithmes de quantification et de codage adaptés aux images 3D et qui ont prouvé leur efficacité dans le cas des images 2D.

Pour l'étude et la construction de ce schéma de compression/décompression pour images hyperspectrales, nous aborderons particulièrement les points suivants :

- Théorie des images hyperspectrales ; Proposer et justifier notre schéma de compression;

- L'élaboration et la mise en place d'une transformée en ondelettes 3D (TOD 3D) pour décomposer l'image hyperspectrale ;
- L'extension du codage SPIHT à la troisième dimension afin de quantifier et coder les coefficients ondelettes obtenus par la TOD 3D ;
- La construction d'une interface graphique sous MATLAB, afin de faciliter l'utilisation des algorithmes élaborés et tester la compression obtenue pour différents paramètres.

Ce mémoire s'articulera sur quatre 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.

Dans le **deuxième chapitre**, nous développerons une transformée en ondelettes 3D basée sur l'algorithme de Mallat, qui consiste en une décomposition qui s'appuie sur les filtres passe-haut et passe-bas discrets déduits de l'ondelette mère et de la fonction échelle associée, puis nous implémenterons cette TO 3D sous Matlab. Pour la compression d'image, la transformée représente le premier maillon de la chaîne, afin de compresser l'information, il faut compléter le cycle par une quantification et un codage.

Dans le **troisième chapitre** nous présenterons les méthodes de codage des coefficients ondelettes basés sur l'hypothèse des arbres de zéros. Ces méthodes de codage, utilisent un modèle simple pour caractériser les dépendances parmi les coefficients ondelettes localisées dans les sous-bandes ayant la même orientation. Nous choisirons d'implémenté la variante 3D du codage SPIHT (Set Partitionning in Hierarehical Tree). SPIHT produit directement des symboles binaires. Ainsi, nous pouvons choisir de ne pas implémenter un codage arithmétique afin de ne pas augmenter la complexité de l'algorithme.

Dans le **dernier chapitre** de ce mémoire, nous allons présenter l'interface graphique développée sous MATLAB (GUI MATLAB) regroupant les algorithmes décrit précédemment et décrire son utilisation pour la compression d'images hyperspectrales. Nous ferons l'expérimentation de cet algorithme de compression pour différents paramètres. Nous

évaluerons les résultats obtenus et nous verrons notamment l'apport de la méthode 3D (compression de la dimension spectrale) par rapport à la méthode 2D. Nous conclurons par quelques perspectives pour d'autres études ultérieures, afin d'améliorer l'algorithme de compression élaboré.

Imagerie Hyperspectrale

CHAPITRE 1

- Introduction aux images hyperspectrales.
- Propriétés des images hyperspectrales.
- Généralités sur la compression d'images.
- Démarche générale du schéma de compression proposé.

1. Introduction aux images hyperspectrales :

1.1. Notion de base de l'imagerie hyperspectrale :

La télédétection désigne l'ensemble des techniques qui permettent d'obtenir des informations sur des objets ou des phénomènes, par l'intermédiaire d'instruments de mesures (un satellite par exemple), sans contact direct avec les cibles étudiées, en utilisant les propriétés des ondes qu'ils émettent ou réfléchissent.

Quand les rayons du soleil frappent un objet, certaines longueurs d'onde du spectre sont réfléchies, d'autres sont absorbées. Les différents objets n'absorbent et ne réfléchissent pas la même partie du rayonnement spectral. En conséquence, le spectre du rayonnement émis diffère selon l'objet. La télédétection mesure le rayonnement réfléchi par les objets et permet donc de déterminer certaines propriétés de ces objets. Elle utilise le même principe que la vision humaine : exploiter la couleur et parfois la texture, pour identifier les objets, mais son champ d'analyse est élargi à des parties du spectre électromagnétique allant au-delà du domaine de la lumière visible.

Par exemple, les capteurs utilisés par le satellite SPOT 5, peuvent mesurer le rayonnement réfléchi dans le domaine visible (de 0.50 a 0.68 µm) et dans le domaine proche et moyen infrarouge (0.78 a 0.89 µm et de 1.58 a 1.75 µm).

- 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)

Pour le satellite LANDSAT 7, ses capteurs peuvent mesurer le rayonnement dans le domaine du visible, du proche et moyen infrarouge et le rayonnement dans l'infrarouge thermique.

- Bande 1 : (0.45-0.52 µm) Bleu
- Bande 2 : (0.52-0.60 µm) Vert
- Bande 3 : (0.63-0.69 µm) Rouge
- Bande 4 : (0.76-0.90 µm) Proche infrarouge
- Bande 5 : (1.55-1.75 µm) Moyen infrarouge
- Bande 6 : (10.4-12.5 µm) Infrarouge thermique
- Bande 7 : (2.08-2.35 µm) Moyen infrarouge

Ces capteurs électroniques permettent l'observation de la même scène dans plusieurs longueurs d'ondes. Ils sont appelés capteurs **multispectraux** et les images qu'ils fournissent sont appelées images multispectrales. Un très grand nombre de ces capteurs multispectraux se sont ainsi développés, le premier étant Landsat au début des années 70.

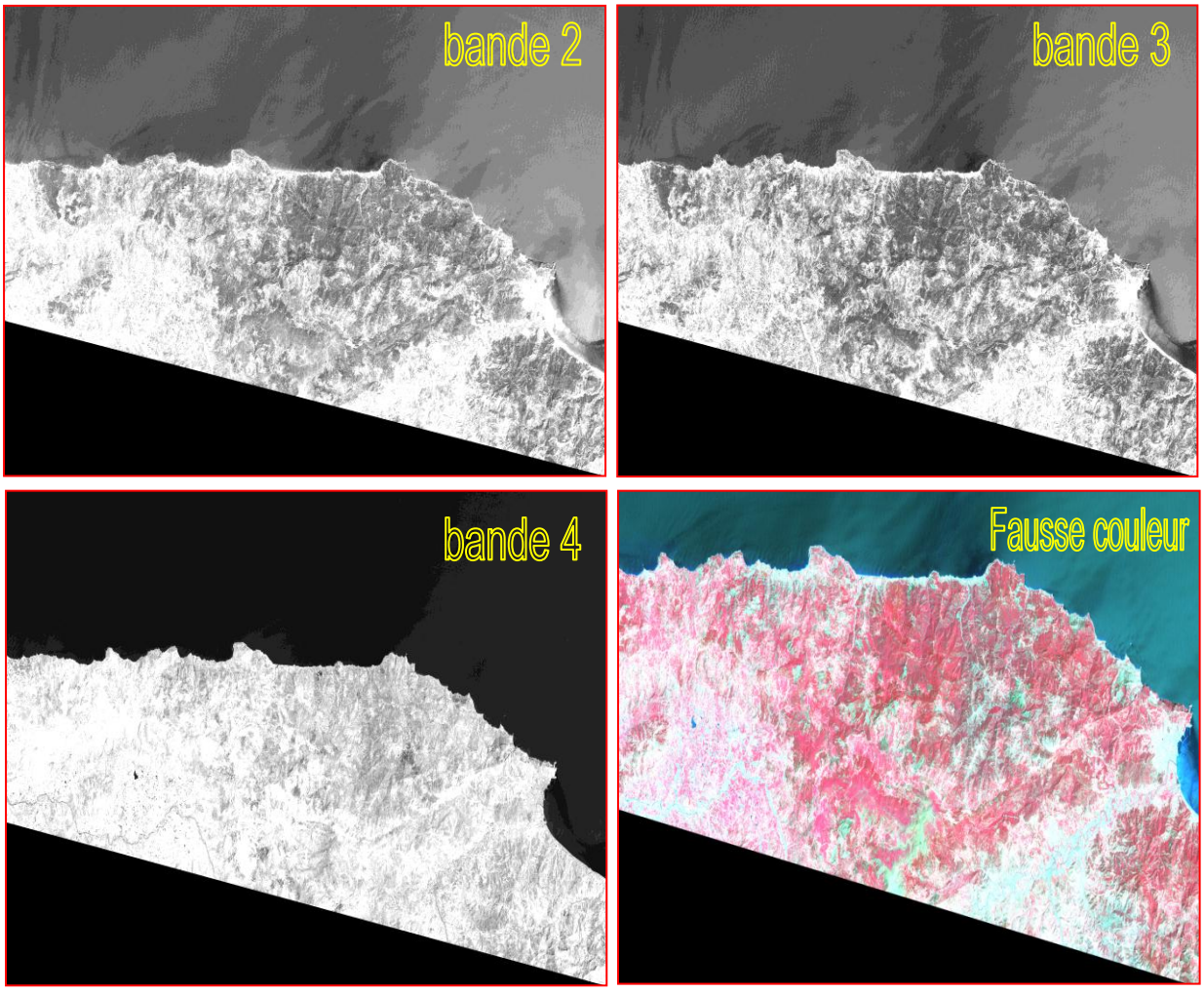


Fig.I.1 – Image **LANDSAT 7** du nord de la wilaya de **TIZI-OUZOU** bande verte (bande 2), rouge (bande 3), infrarouge (bande 4) et l'image fausse couleur obtenue à l'aide des trois bandes verte, rouge et proche infrarouge sous le logiciel ENVI 4.

L'évolution naturelle des capteurs d'images a conduit à l'acquisition non pas d'une, deux ou trois bandes spectrales mais plutôt de plusieurs centaines d'images spectrales, dont 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.

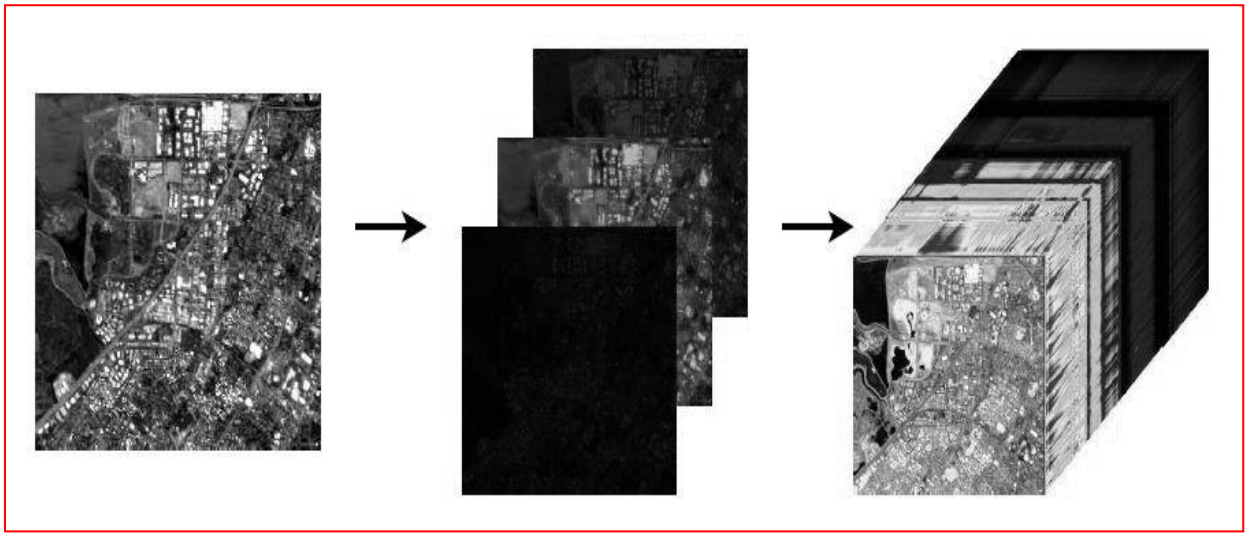


Fig.I.2 – Du monochrome à l'hyperspectral.

L'élément nouveau apporté par les images hyperspectrales est la signature spectrale. La signature spectrale comporte des informations uniques concernant les propriétés physiques, chimiques des matériaux et des objets analysés. 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). Nous pouvons dire qu'avec les images hyperspectrales, nous sommes effectivement capables d'acquérir des informations présentes dans l'ensemble de la signature spectrale, là où les systèmes classiques de télédétection, comme ceux d'imagerie multispectrale, où on ne propose que quelques portions de ce spectre (voir figure I.3). Cette nouvelle forme de télédétection permet de reconnaitre la plupart des objets dans l'image.

La résolution spectrale est le paramètre le plus important des images hyperspectrales. Elle permet d'obtenir des échantillons détaillés des spectres. Pour ce paramètre, l'essentiel tient à la continuité des canaux avec une largeur assez étroite. Parfois les capteurs hyperspectraux, comme CASI, ont une capacité de résolution flexible qui permet de choisir à la fois la largeur et le positionnement des canaux spectraux. Cette capacité est très pratique, selon l'exploitation que l'on souhaite faire.

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.

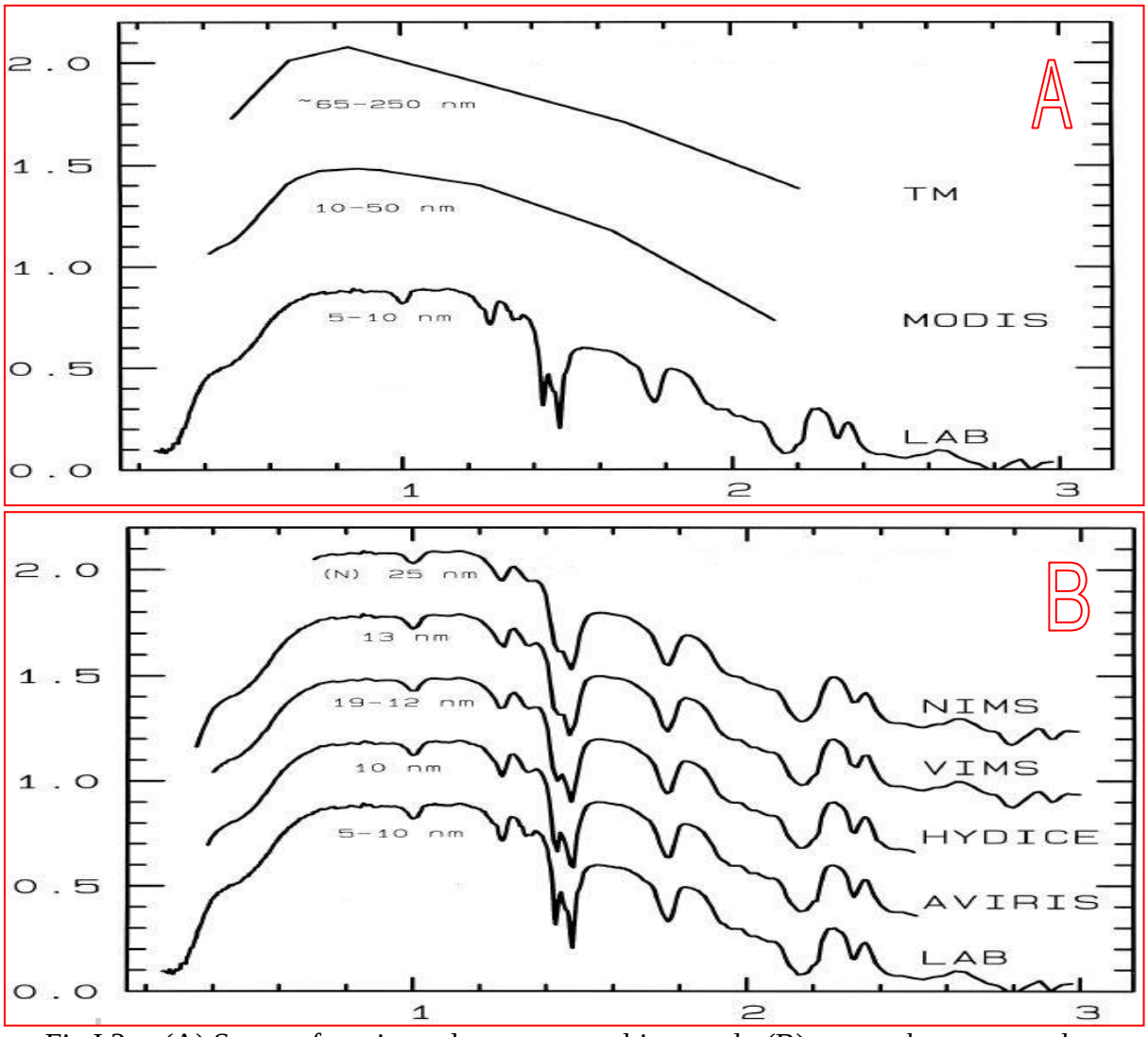


Fig.I.3 – (A) Spectre fournis par le capteur multispectral ; (B) spectre hyperspectral.

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 réaliser une avancée majeure en mettant en œuvre le système AVIRIS. L'AVIRIS était un capteur hyperspectral avancé, porté par un U2. 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 **[1].**

Actuellement, deux technologies sont opérationnelles : aéroportées et spatioportées.

1.2. Capteurs aéroportés :

Les capteurs aéroportés sont historiquement les premiers capteurs grâce auxquels l'imagerie hyperspectrale a été appliquée et développée. Les capteurs aéroportés trouveront toujours des applications, notamment lorsque de très hautes résolutions spatiales (De 1 à 4 m) sont nécessaires. Le tableau (I.1) recense les capteurs hyperspectraux aéroportés les plus connus.

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 Imagery Collection Experiment	USA	210	0.4 - 2.5
CASI	Compact Airborne Spectrographic Imager	Itres Research Canada	228	0.1 - 1.0
AISA	Airborne Imaging Spectrometer	Specim Ltd. Finland	228	0.43 - 1.0
MODIS	Moderate Resolution Imaging Spectrometer	NASA USA	36	0.41 - 14.24

Tableau.I.1 Capteurs hyperspectraux aéroportés.

1.3. 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), qui a été lancé dans le cadre de programme millénaire de la NASA [EO1/NASA]. Plusieurs programmes non gouvernementaux ont été également développés, parmi lesquels le capteur CASI (Compact Airborne Spectrographic Imager) qui est l'un des plus employé. Le tableau (I.2) 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 portés par satellites.

Dans ce qui suit nous allons décrire plus en détails les caractéristiques du capteur AVIRIS, étant donné que c'est les images issues de ce capteur que nous allons utiliser et essayer de compresser en utilisant une transformée en ondelettes 3D.

1.4. Le capteur AVIRIS :

AVIRIS (Airborne Visible / Infrared Imaging Spectrometer) est l'un des tout premier imageur à spectromètre, il a été mis au point en 1987 par la NASA. Ce capteur est capable d'acquérir des images spectrales dans 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 et 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 4m-20m). Le tableau (I.3) résume les caractéristiques du capteur :

Caractéristiques d'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

Tableau.I.3 Caractéristiques du capteur AVIRIS.

Les avantages :

Le capteur AVIRIS possède un certain nombre d'avantages par rapport à d'autres technologies pour obtenir des informations sur la Terre. Par exemple :

1. La flexibilité de la résolution spatiale grâce au porteur aéroporté (entre 4m-20m).
2. le nombre et la largeur des canaux spectraux; grâce à la technologie spécifique du système optoélectronique.
3. Une résolution radiométrique assez élevée (12bits).
4. Le système est calibré.
5. La possibilité de corrections géométriques par les systèmes GPS.

2. Propriétés des images hyperspectrales:

Les caractéristiques principales sont basées sur les différentes capacités du système d'imagerie. Il s'agit des caractéristiques spatiales, spectrales et radiométriques des images **[2].**

2.1. 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. 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.

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

2.2. Résolution spectrale :

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

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

2.3. Résolution radiométrique :

Le flux de la radiance qui arrive sur chaque détecteur, pour une longueur d'onde spécifique est une valeur analogique qui détermine les niveaux de gris de chaque pixel. La résolution radiométrique se mesure en nombre de bits. L'équation suivante définit le nombre du niveau de gris N par rapport au nombre de bits n :

$$N = 2^n$$

Par exemple pour un capteur qui a une résolution radiométrique de 10 bits, nous avons des pixels avec des valeurs de gris comprises entre 0 et 1023. Avec cette caractéristique qui est normalement élevée dans le cas de l'imagerie hyperspectrale, il est probable que deux matériaux très similaires apparaissent avec des valeurs légèrement différentes.

Avec la résolution radiométrique, 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).

2.4. La couverture spectrale :

Il s'agit de l'étendue du spectre électromagnétique qui est couverte par les capteurs, tels que l'ultraviolet, le visible, l'infrarouge réfléchissant, l'infrarouge thermique et les microondes.

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.

2.5. 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 spatiale de 20 mètres seulement.

Le tableau (I.4) présente les spécifications typiques pour un capteur hyperspectral.

Spectre	**400-2500 nm**
Résolution spatiale	**20 m**
Nombre de bandes	**20-250**
Résolution spectrale	**10 nm**
Quantification	**12 bits**

Tableau.I.4 spécificités d'un capteur hyperspectral.

2.6. Le concept du cube de données :

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]** de la manière suivante :

– la largeur de ce cube, mesurée en pixels, est liée à la résolution spatiale.

– la longueur du cube mesurée 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.4) montre un cube de données hyperspectrales, 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.

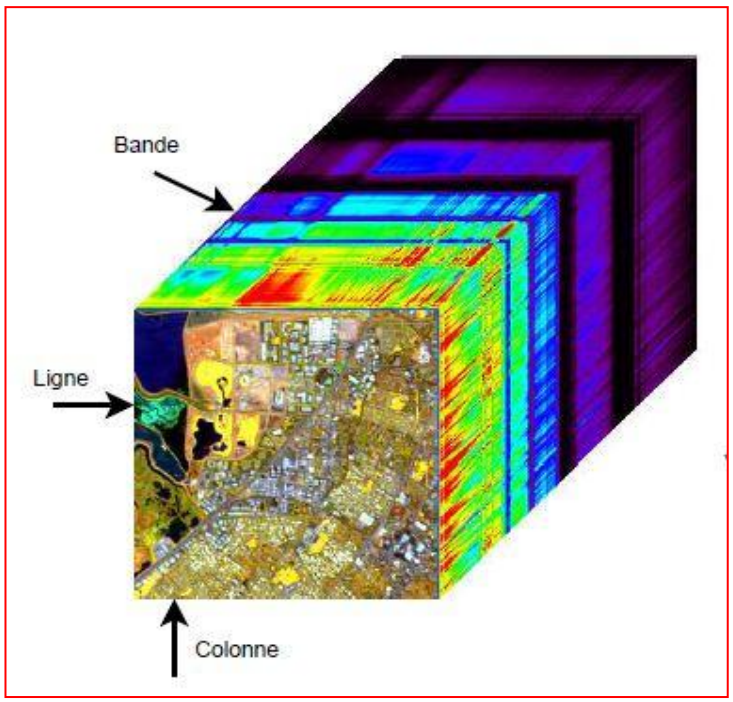


Fig.I.4 – Le cube hyperspectral.

Les images hyperspectrales constituent un flot de données considérables à transmettre. Élaborer des méthodes de compression efficaces est un enjeu majeur pour leur stockage et leur transmission.

2.7. 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.

Le facteur de corrélation entre les différentes bandes spectrales peut être calculé et une matrice de corrélation spectrale peut être ainsi obtenue. Le coefficient (i, j) de cette matrice de corrélation représente la corrélation entre les bandes i et j. Il est défini par l'équation suivante :

$$r(i,j) = \frac{\sum_{x=1}^{n_x}\sum_{y=1}^{n_y} Ic(x,y,\lambda_i)Ic(x,y,\lambda_j)}{\sqrt{\sum_{x,y} Ic(x,y,\lambda_i)^2}\sqrt{\sum_{x,y} Ic(x,y,\lambda_j)^2}} \quad --(1)$$

Le coefficient de corrélation se situe dans l'intervalle [−1; 1], une valeur de 1 indiquant une égalité entre les deux bandes. Les valeurs étant en fait fortement corrélées, il y a rarement apparition de coefficients négatifs en pratique.

Du fait de la taille importante des images hyperspectrales, on comprend vite l'intérêt 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 passer en revue les notions de base de la compression, en suite nous proposerons notre méthode et la mettrons en œuvre dans les chapitres II et III.

3. Généralités sur la compression d'image :

3.1. Notion de base de la compression :

Nous détaillons à travers ce paragraphe 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.) **[3].** De plus, les exigences en termes de qualité sont différentes. Il existe deux principales chaînes de compression :

- 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 :

1. les méthodes prédictives : celles-ci exploitent la redondance spatiale qui existe entre la valeur courante et les valeurs précédentes ou suivantes ;
2. 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.

- 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.). La figure (I.5) montre les principales étapes d'une chaîne de compression avec perte.

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 :

1. 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 ;
2. 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.

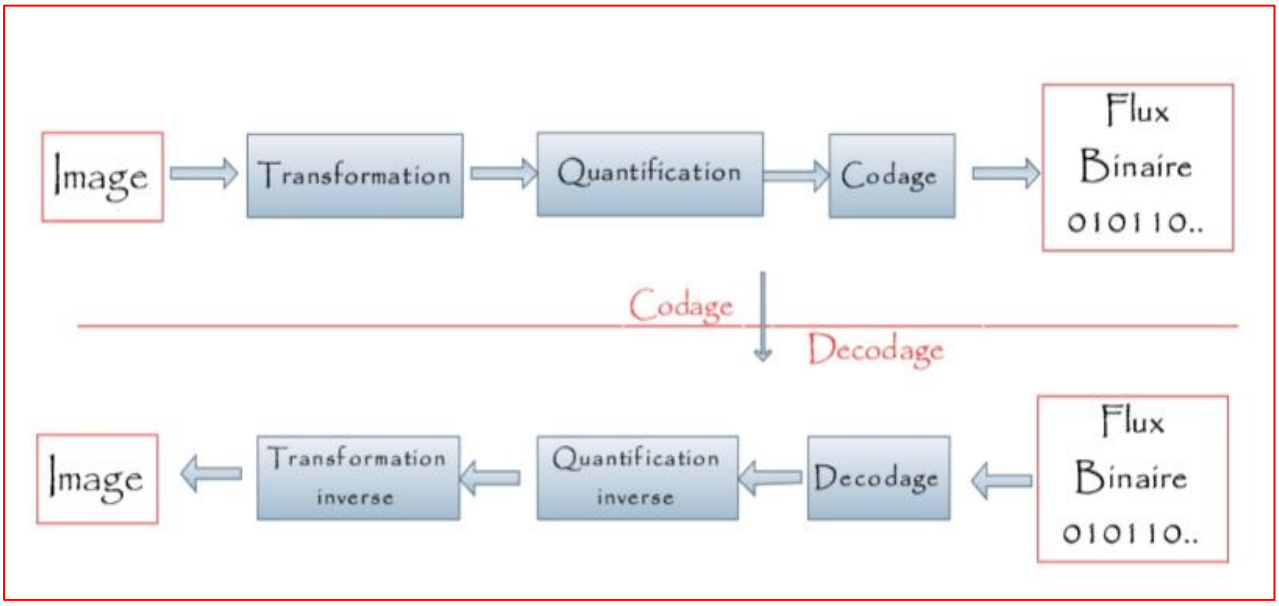


Fig.I.5 – Les 3 étapes classiques de compression d'images.

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 **[3].** 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.

3.2.1. Première étape : 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 Tx le signal projeté dans une base de fonction (fw) w ∈ R, on a :

$$Tx(w) =< X, fw >= \int_{-\infty}^{+\infty} X(t)\ fw(t)dt \quad --(2)$$

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.

3.2.2. Deuxième étape : 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.

3.2.3. Troisième étape : 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 (Fig.I.5) 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 (ou 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 allons maintenant voir ce qui se fait en pratique, dans ce qui suit 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 standard nous permettra de justifier le choix des méthodes et codes proposés dans notre schéma de compression pour images hyperspectrales.

3.3. Compression d'image : 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.

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 éliminés 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.
- 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 Huffman.

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.

3.3.2. JPEG 2000 :

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) **[4].**

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 correspondent 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.

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) **[4]** montre 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.

Fig.I.6 – (a) Image LENA ; (b) compression par JPEG ; (c) compression par JPEG 2000.

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.

4. Démarche générale du schéma de compression proposé :

Notre schéma de compression s'inscrit dans le cadre d'une approche 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 (TO 3D), est maintenant admise comme référence en compression d'images volumiques **[5].**

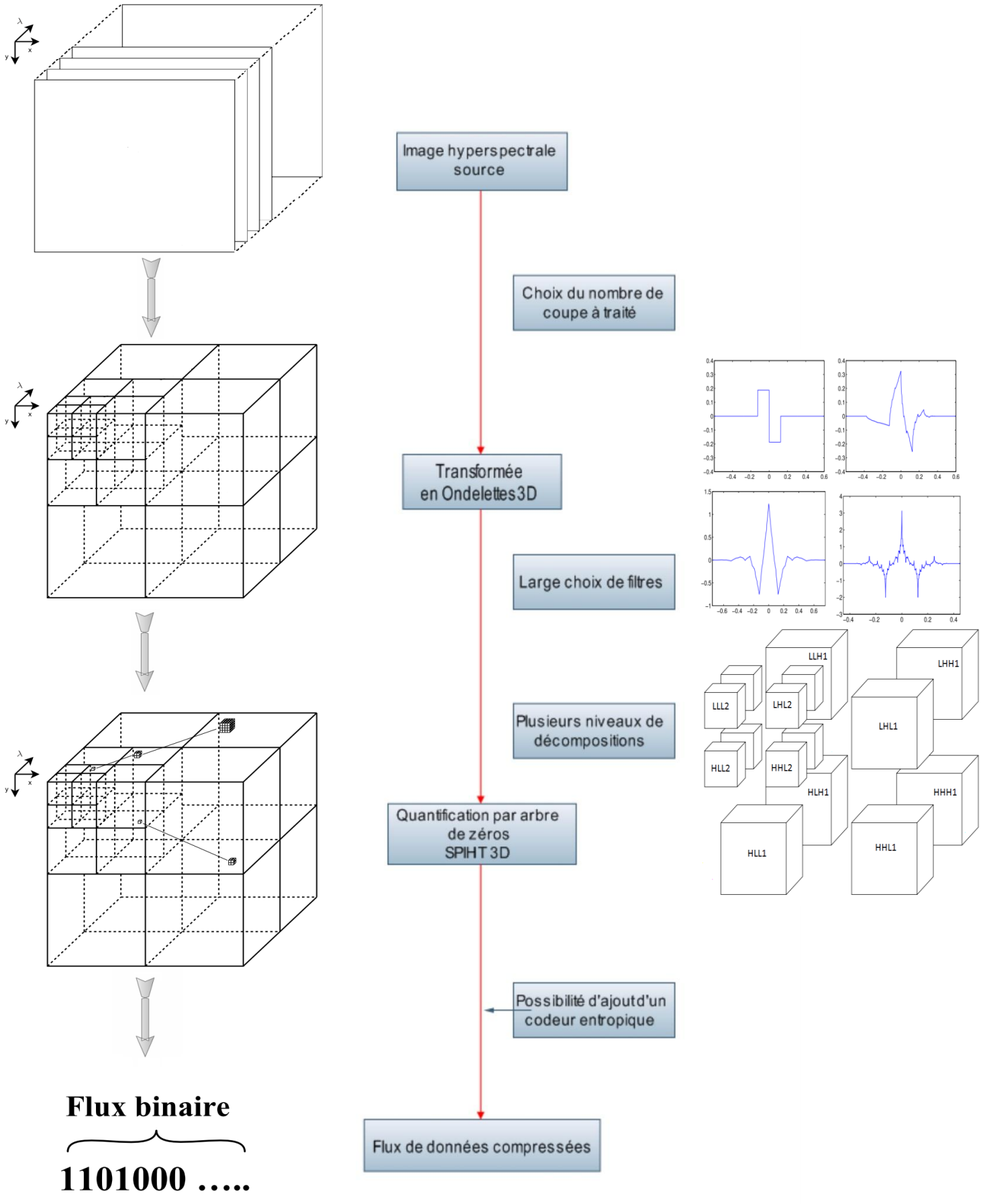
Le premier objectif de ce mémoire est de présenter les ondelettes, de développer une transformée en ondelettes **[6] [7]** et de faire l'extension 3D de cette transformée ainsi que son implémentation sous Matlab. Pour la compression d'images, la transformée représente le premier maillon de la chaîne, 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 **[8] [9] [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 **[9]** produit directement des symboles binaires. Ainsi, un codeur entropique n'est pas obligatoire même s'il est souvent implanté. 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.7) présente notre schéma de compression/décompression développé pour les images hyperspectrales 3D.

Fig.I.7 – Notre schéma de compression pour images hyperspectrales.



Comme on vient de le voir, notre schéma se compose de deux étapes : une transformation (TOD 3D) et un codage (SPIHT 3D). Nous allons décrire brièvement chacun d'eux dans ce qui suit :

4.1. Premier étape : Transformation TOD 3D :

Nous proposons de développer une transformée en ondelettes 3D basée sur l'algorithme de Mallat **[6]**, 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.

Notons aussi que nous allons définir un large choix de filtres utilisables, tel que les filtres de daubechies, les symlettes et d'autres filtres qui peuvent être utilisés par la transformation développée. Cependant l'utilisation du filtre CDF 9/7 pour la transformation en x et y dans le JPEG2000 a montré ses performances. D'autres auteurs 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 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.

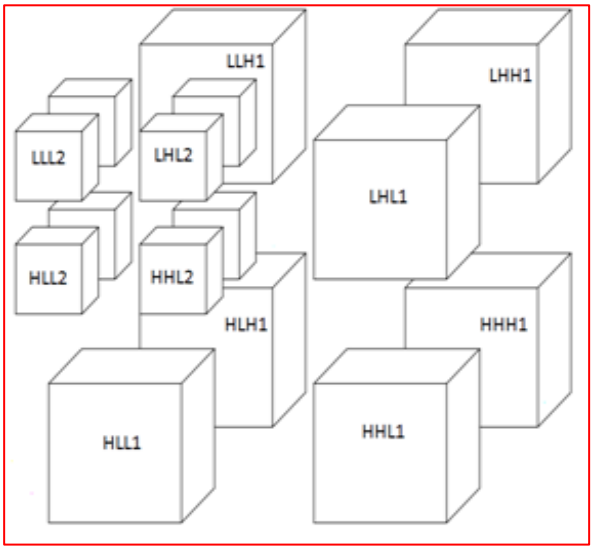


Fig.I.8 – Blocs de données 3D obtenus par une transformée sur 2 niveaux.

4.2. Deuxième étape : 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 **[9],** 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 3D utilise 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.

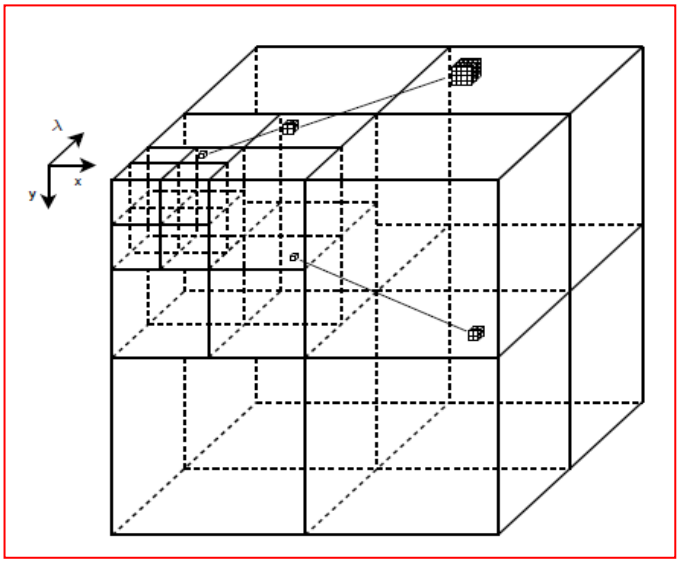


Fig.I.9 – Arbres de zéros du SPIHT 3D.

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

Transformée en Ondelettes

CHAPITRE 2

- Généralités sur les ondelettes.
- Transformée en ondelettes discrète.
- L'algorithme de la TOD 3D sous MATLAB.

1. Généralités sur les ondelettes :

1.1. Pourquoi les ondelettes ?

Que sont les ondelettes ? L'origine du mot *ondelette* n'est, en soi, pas très compliqué. Les ondelettes utilisées par Jean Morlet avaient, en effet, une allure de *petites ondes*. D'où cette appellation. Vers 1975, ce géophysicien, a créé des petites fonctions mathématiques ayant certaines propriétés intéressantes ; ceci afin de sonder les différentes couches géologiques. Morlet est donc un des artisans majeurs des ondelettes. Cependant, d'autres chercheurs travaillant dans des domaines de recherches différents, utilisaient à cette époque des outils forts semblables aux *petites ondes* de Morlet. Seulement, ces méthodes étaient souvent expérimentales et ne possédaient aucun formalisme rigoureux.

Ce formalisme émergera durant les années 80 des travaux de scientifiques tels que ceux d'Yves Meyer, de Stéphane Mallat, d'Ingrid Daubechies et de bien d'autres **[11] [12].** Les ondelettes sont, avant tout, un puissant outil d'analyse mathématique, mais avant de parler d'analyse en ondelettes commençant par parler de l'analyse de Fourier.

L'analyse de Fourier est un outil de base en traitement du signal, indispensable dans de nombreux domaines de recherche, mais elle montre vite des limites justifiées dès lors que l'on sort du cadre rigoureux de sa définition : le domaine des signaux stationnaires d'énergie finie. Dans l'analyse de Fourier tous les aspects temporels (début, fin, durée d'un événement), bien que présents dans la phase, deviennent illisibles dans le spectre. En particulier, la transformée de Fourier (TF) d'un morceau de musique qui ne permet pas de retrouver le rythme joué, mais simplement les notes présentes. Le spectre seul ne permet pas de dissocier deux partitions différentes ayant les mêmes notes. Or, on souhaiterait pourtant parfois réaliser à la fois une analyse en temps et en fréquence, pour retrouver la « portée musicale » associée à ces **signaux non stationnaires**.

L'étude des signaux non stationnaires nécessite donc le développement de méthodes spécifiques. La première solution, mise en place intuitivement au milieu du siècle, correspond aux **analyses de Fourier à fenêtre glissante** (FFG) introduites dès 1945 par D. Gabor avec l'idée d'un plan temps-fréquence où le temps deviendrait un paramètre complémentaire de la fréquence. À ces approches, s'est ajoutée la **transformée en ondelettes**, existant à l'état latent aussi bien en mathématiques qu'en traitement du signal, mais dont le véritable essor a commencé au début des années 1980.

Dans ce chapitre nous allons exposer quelques notions de base sur la **transformée en ondelettes** mais avant, nous allons commencer par des rappels sur la transformée de Fourier, puis la transformée de Fourier à fenêtre glissante montrant leurs limites sur des exemples classiques (signaux musicaux) et nous introduirons ainsi le concept de l'analyse multi-résolution comme outil de construction des ondelettes.

1.1.1. La transformée de Fourier :

C'est une généralisation de la décomposition des séries de Fourier à tous les signaux déterministes. Elle permet d'obtenir une représentation en fréquence (représentation spectrale) de ces signaux. Elle exprime la répartition fréquentielle de l'amplitude, de la phase et de l'énergie (ou de la puissance) des signaux considérés.

Définition :

Soit $f(t)$, un signal déterministe. Sa transformée de Fourier est une fonction, généralement complexe, de la variable f et qui est définie par :

$$\hat{f}(\xi) = \int_{-\infty}^{+\infty} e^{-\xi t} \; f(t)dt \quad --(1)$$

Du point de vue mathématique, un signal est une fonction $S(t)$, représentant une information d'origine variée (acoustique, optique, ...etc.). L'analyse de ce signal consiste à mettre en évidence certains phénomènes (transitoires, périodiques, ...etc.) qui le composent : Dans le cas de l'analyse par transformée de Fourier, les fonctions analysantes sont les ondes pures ($e^{i\xi t}$) de fréquence ξ. Projeter le signal sur ces fonctions (analyse spectrale) renseigne sur son caractère fréquentiel.

La transformée de Fourier peut être vue comme une représentation à base des sinusoïdes qui sont bien localisées en fréquences, mais pas en temps (ou espace) car leur support est infini (c'est une conséquence de leur périodicité). Donc les inconvénients de l'analyse par la transformée de Fourier proviennent du fait que les fonctions analysantes sont à support de longueur infinie, ce qui ne correspond pas à un signal physique qui est à support temporel fini.

On peut résumer la limite majeur de la transformée de Fourier dans le point suivant :

– Perte de localisation temporelle : Considérant un signal composé d'une succession de deux sinusoïdes de fréquences différentes, la première étant une sinusoïde d'une fréquence de 20Hz et la deuxième de 85Hz ; (c'est l'exemple d'un signal musical composé de deux notes jouées une après l'autre). La transformée de Fourier indique les fréquences, mais ne donne aucune information concernant l'ordre d'apparition des sinusoïdes, (voir Fig.II.1).

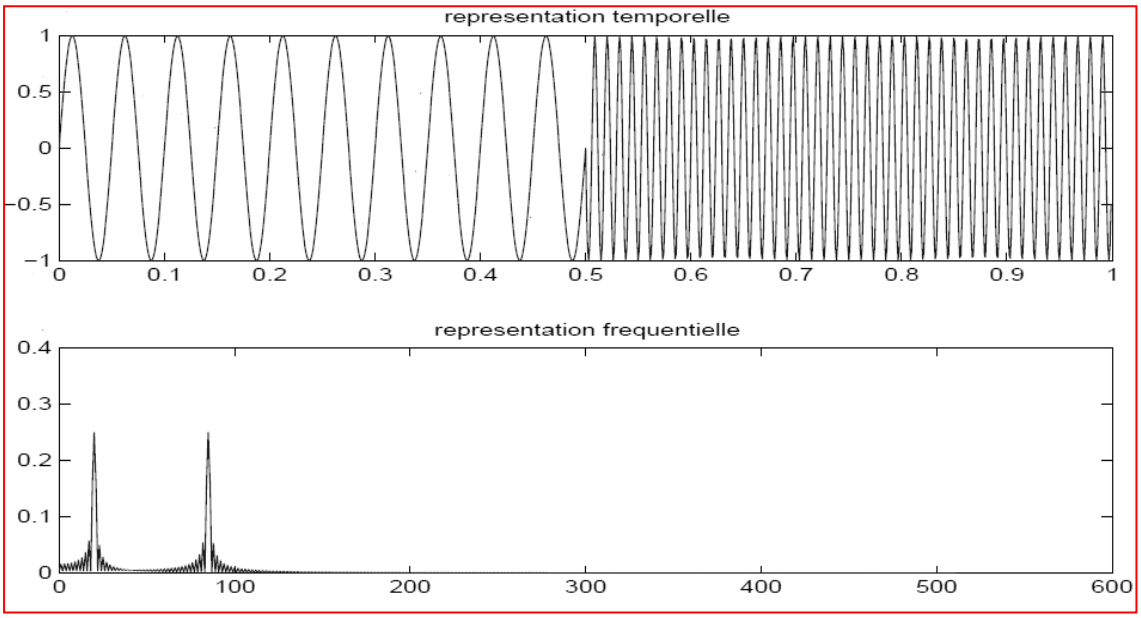


Fig.II.1 – Signal de deux sinusoïdes et module de sa transformée de Fourier.

L'analyse en fréquence ne nous informe pas sur la localisation temporelle du changement du régime dans le signal.

1.1.2. Transformée de Fourier à fenêtre glissante :

Pour répondre au problème de localisation de la transformée de Fourier, D. Gabor a introduit vers les années 40 l'approche de la transformée de Fourier à fenêtre glissante, il s'agit de "découper" le signal en petits "morceaux" et de calculer ensuite leurs transformée de Fourier. Dans la pratique cette méthode consiste à multiplier le signal S(t) par une fonction (fenêtre) localisée dans le temps ou l'espace (à support compact ou à décroissance rapide) $\mathbf{g(t-\mu)}$; ainsi la transformée de Fourier à fenêtre glissante est définie par :

$$T^{fen}S(\xi,\mu) = \int_{-\infty}^{+\infty} S(t)\ \ \overline{g(t-\mu)}\, e^{-i\xi(t-\mu)}dt \quad --(2)$$

$T^{fen}\,S(\xi,\mu)$ renseigne sur la nature fréquentielle du signal S au voisinage du point µ.

Les fenêtres $g_{\mu,\xi}(t) = g(t-\mu)e^{i\xi t}$ sont appelées les *atomes de Gabor*. Elles sont construites en translatant une fenêtre de base *g* en temps et en fréquences. L'analyse par cette transformée permet de retrouver à la fois les fréquences (les notes) et l'information temporelle (l'ordre dans lequel elles sont jouées).

1.1.2.1. Limites de la TF à fenêtre glissante :

Considérons par exemple le signal S2 = S1 +δ0.6 + δ0.7. Ce signal est défini par deux phénomènes différents : Deux sinusoïdes successives (défini précédemment) et deux pics de Dirac qui représentent des défauts locaux d'enregistrement (perturbations).

L'analyse par transformée de Fourier à fenêtre glissante ne permet pas de mettre en évidence simultanément les deux phénomènes et ceci est dû à la taille de la fenêtre qui est fixe, ce qui implique qu'on aura une résolution temporel et fréquentiel fixe.

1.1.2.2. Principe d'incertitude de Heisenberg :

Il joue un rôle important dans la théorie du traitement du signal car il donne une limite à la précision de la localisation des différentes représentations d'un signal. Ainsi le principe nous dit que l'énergie d'un signal et celle de sa transformée de Fourier ne peuvent être localisées avec une précision arbitraire à la fois en temps et en fréquence.

Théorème d'incertitude de Heisenberg.

Si f est dans L^2, alors on peut définir son écart-type en temps σ_t et l'écart-type (en fréquence) de sa transformée de Fourier σ_w. Alors on a:

$$\sigma_w * \sigma_t \geq 1/4$$

On est donc contraint à un compromis entre résolution temporelle et fréquentielle.

Autrement dit, il est impossible de trouver en pratique une fenêtre g qui soit bien localisée à la fois en temps et en fréquence. Ceci est justifié par le principe d'incertitude d'Heisenberg, qui interdit à un signal donné d'avoir une localisation arbitraire précise en temps et en fréquence.

La localisation dans le plan temps-fréquence de la transformation peut être estimée par les **variances** de la **fonction analysante** dans l'espace temporel et dans l'espace fréquentiel, elle est représentée sous forme d'une boîte de **Heisenberg**.

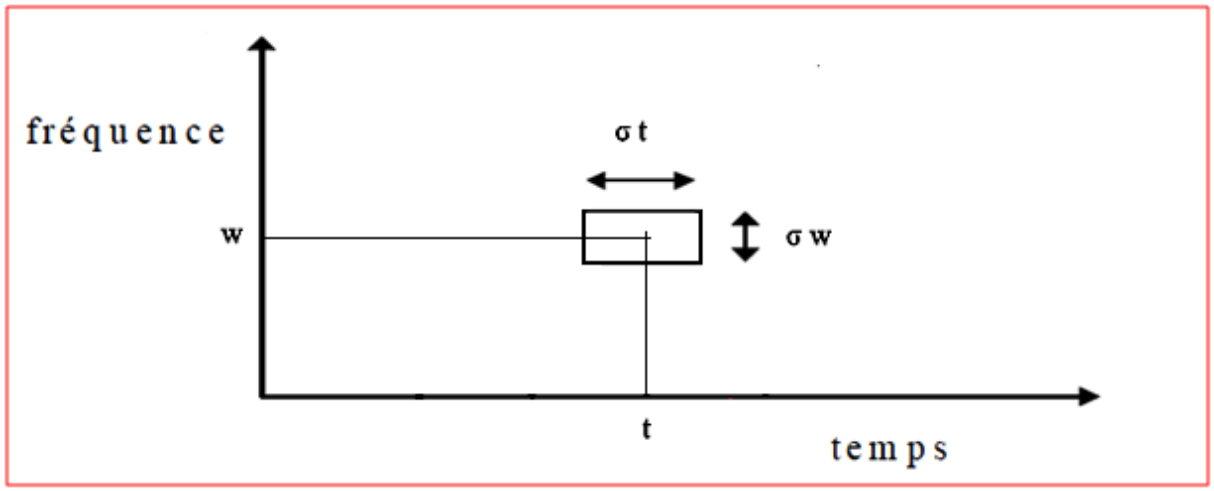


Fig.II.2 – Boîte d'Heisenberg

Dans notre cas où on analyse un signal par une fenêtre, on peut estimer la résolution temporelle et fréquentielle comme ceci :

Soit $g_{\mu,\xi}(t)$ les fonctions (fenêtres) que l'on utilise pour analyser le signal dans le plan temps fréquences; ces fenêtres sont appelées des *atomes de Gabor* :

$$g_{\mu,\xi}(t) = g(t-\mu)e^{i\xi t} \quad --(3)$$

L'énergie de $g_{\mu,\xi}(t)$ est localisée au voisinage de μ sur une largeur propre à la fonction g et qui est donné par σ_t .
De même pour évaluer la translation en fréquences, on peut prendre la transformée de Fourier de $g_{\mu,\xi}(t)$ comme suite :

$$\hat{g}_{\mu,\xi}(\omega) = \hat{g}(\omega-\xi)\, e^{-i\mu(\omega-\xi)} \quad --(4)$$

L'énergie de $g_{\mu,\xi}(w)$ est donc localisée au voisinage de la fréquence ξ sur un intervalle de largeur σ_w.

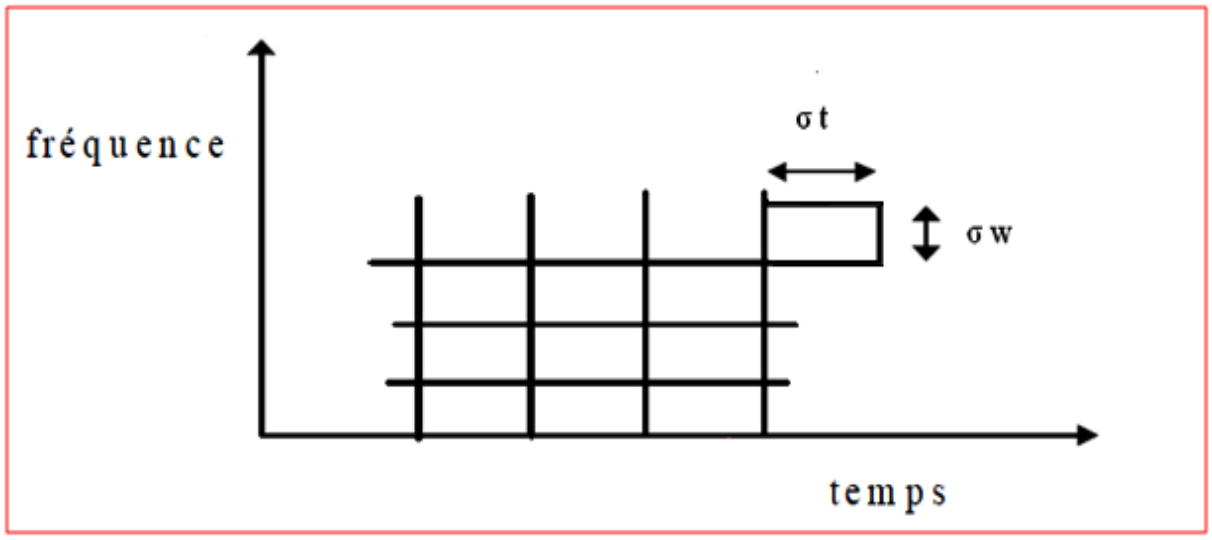


Fig.II.3 – Atome **fixe** de la Transformée de Fourier à fenêtre.

On voit bien que dans le cas où on analyse un signal par une fenêtre que l'on déplacerait le long de l'axe du temps, on a une conséquence immédiate qui est alors une résolution fixe en temps et en fréquences, en d'autres termes, la largeur de la fenêtre en temps σ_t et en fréquence σ_w seraient fixées et bornées inférieurement par l'équation Heisenberg.

La meilleure localisation possible sera obtenue quand la surface des rectangles sera minimum, c'est à dire quand on aura l'égalité dans l'équation. On peut montrer qu'alors g est nécessairement une gaussienne et les atomes $g_{\mu,\xi}$ sont alors appelés *fonctions de Gabor*.

1.1.2.3. Discussion :

La taille constante de la fenêtre, fixe la largeur de bande du filtre dans l'espace des fréquences et par conséquence la résolution fréquentielle de l'analyse. Le principe d'incertitude impose ensuite leur précision temporelle. Quels que soient les fréquences aux instants scrutés, la précision de notre outil demeure constante, les basses et hautes fréquences sont vues de la même manière.

1.1.3. Les ondelettes :

La transformée en ondelettes, est comme celle de Fourier, un passage d'une représentation à une autre ; mais comme la transformée à fenêtres, elle permet aussi de mesurer les variations dans le temps des composantes fréquentielles (spectrales) d'un signal. Néanmoins, la résolution temps-fréquence de la transformée en ondelettes est différente. En effet, dans la transformée de Fourier à fenêtre, la largeur en temps et en fréquences des atomes avec lesquels on analyse le signal est fixe et dépend de l'atome utilisé.

L'idée de l'ondelette est de pouvoir faire varier les largeurs en temps et en fréquences d'une fonction tout en la translatant le long du signal, comme dans la transformée de Fourier à fenêtre. La transformée en ondelette d'une fonction f en un point (t,ω) du plan temps-fréquences ne dépend donc que des valeurs de $f(t)$ et $\hat{f}(\omega)$ dans le rectangle de Heisenberg centré en (t,ω). L'avantage de faire varier ces largeurs devient alors évident : on minimise le nombre de translations en temps et en fréquences de la fenêtre en optimisant la largeur de celle-ci. Ainsi dans les basses fréquences, une grande largeur en fréquences n'est pas nécessaire, on peut donc utiliser des rectangles plus larges en temps. Aux hautes fréquences, on va utiliser des rectangles plus larges en fréquences et plus localises en temps. On peut voir cela comme une adaptation de l'ondelette à l'échelle qu'on lui impose : plus la fenêtre est petite dans le temps, plus l'ondelette va être compressée et osciller rapidement. Le contraire se produira lorsque la fenêtre est dilatée. Ainsi, les petites et grandes fenêtres enregistreront respectivement les variations rapides et moyennes du signal.

1.1.3.1. La transformée en ondelettes :

Une ondelette mère ψ est une fonction de base, que l'on va translater et dilater pour recouvrir le plan temps-fréquences et analyser le signal **[12].** L'ondelette doit être une fonction de moyenne nulle ; En d'autres termes, ψ doit être une onde ! Ce qui s'écrit mathématiquement par :

$$\int_{-\infty}^{+\infty} \psi(t)\, dt = 0 \qquad --(5)$$

On introduit alors les facteurs de translation μ et d'échelle s :

$$\psi_{u,s}(t) = \frac{1}{\sqrt{s}}\, \psi\left(\frac{t-u}{s}\right) \qquad --(6)$$

OU :

– Le paramètre μ 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 Fig.II.4).

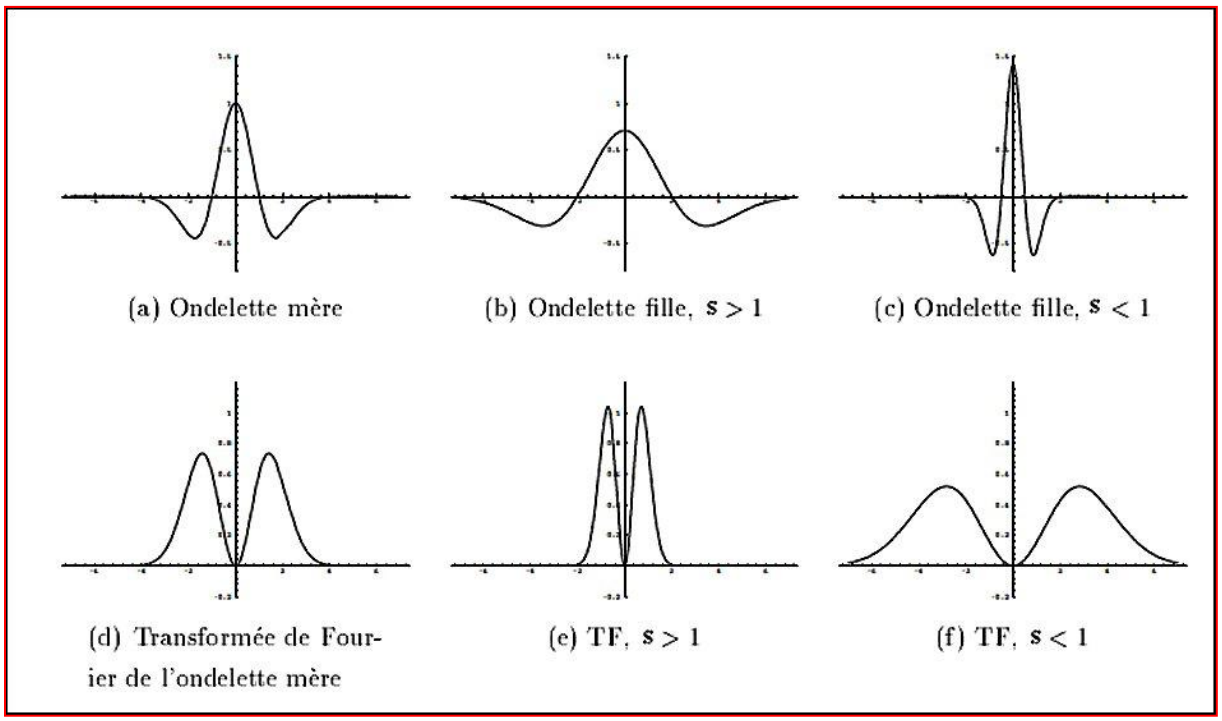


Fig.II.4 – Ondelettes pour plusieurs facteurs d'échelle et leur transformée de Fourier.

On se déplace sur le plan temps-fréquence en faisant varier *s* et *u*. La valeur que l'on obtient est alors la décomposition spectrale locale de f. L'énergie de $\psi_{u,s}$ est concentrée autour de *u* sur une largeur $s * \sigma_t$ ou σ_t est la largeur du rectangle de l'ondelette *mère*. Pour sa transformée de Fourier $\widehat{\psi}_{u,s}$ son énergie est concentrée autour d'une fréquence centrale n *et* son intervalle est d'une largeur de σ_w/s .

En conclusion, les rectangles de Heisenberg sont pour l'analyse en ondelette, de largeur temporelle $s * \sigma_t$ et de largeur fréquentielle σ_w / s. Ou σ_t et σ_w correspondent aux largeurs du rectangle de Heisenberg de l'ondelette *mère*.

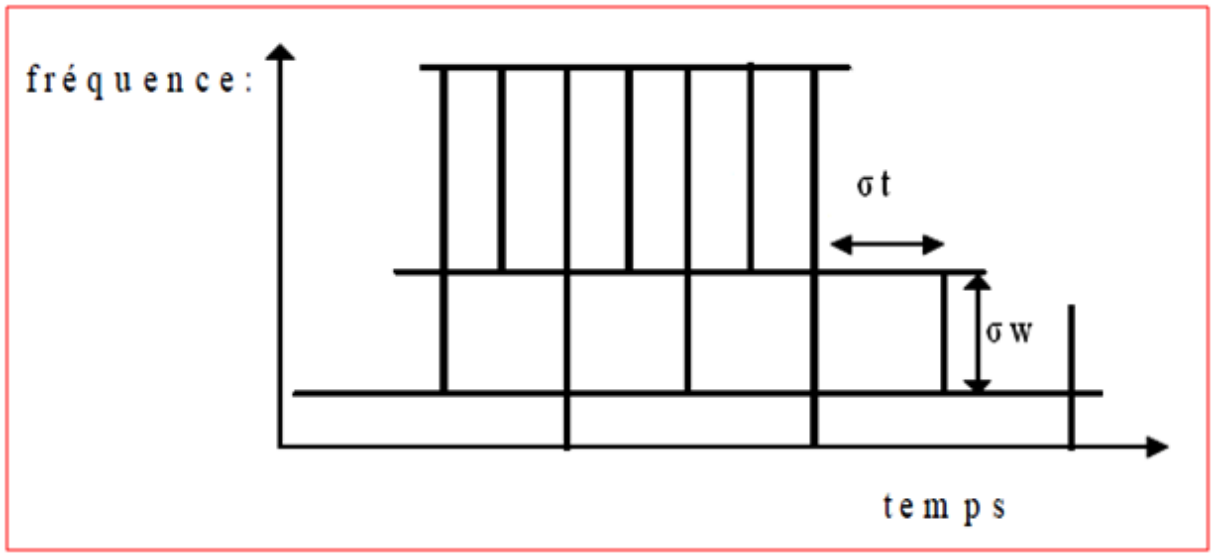


Fig.II.5 – Atome de la Transformée en ondelettes.

La résolution en fréquence de la transformée en ondelettes dépend du facteur de dilatation *s*. Par le principe d'Heisenberg, on peut donc choisir arbitrairement celle-ci suivant ce que l'on désire analyser.

1.1.3.2. Discussion :

Pour l'**analyse de Fourier**, σ_w est fixé à l'échantillonnage, σ_t est égal à la durée du signal (aucune localisation temporelle). Quant à l'**analyse de Gabor**, les valeurs de $\sigma_w$et σ_t sont constantes, imposées par la taille de la fenêtre et pour les **ondelettes**, ces valeurs évoluent suivant les fréquences en gardant une énergie constante. La surface $\sigma_w * \sigma_t$ est limitée à $\frac{1}{4}$ par le principe d'incertitude.

Apres avoir introduit l'idée qui a conduit à la transformée en ondelettes, nous allons présenter ce qu'est une ondelette ainsi que ses principales propriétés. La transformée elle-même et ses propriétés seront présentées dans la prochaine section.

1.2. Ondelette analysante:

Nous avons vu en introduction qu'une ondelette doit être une fonction localisée aussi bien en temps qu'en fréquence. Ceci se traduit dans le domaine fréquentiel par la condition d'admissibilité suivante :

1.2.1. Définition 1 :

Une fonction $\psi \in L^1(\mathrm{R}) \cap L^2(\mathrm{R})$ est une ondelette si elle vérifie la condition d'admissibilité :

$$C_\psi = \int_{-\infty}^{+\infty} \frac{|\widehat{\psi}(\xi)|^2}{\xi} \, d\xi < \infty \quad --(7)$$

On dit que ψ est admissible ou ψ est une ondelette analysante.

1.2.2. Définition 2 :

Une ondelette ψ est dite de classe C^m, m ∈ N si elle satisfait les propriétés suivantes :

(a) si m ≥ 0, $\psi \in L^\infty(\mathrm{R})$ avec ψ, ainsi que toutes ses dérivées jusqu'à l'ordre m appartiennent à $L^\infty(\mathrm{R})$.

(b) ψ, ainsi que toutes ses dérivées, jusqu'à l'ordre m, sont à décroissance rapide à l'infini.

(c) $\int_{-\infty}^{+\infty} t^k \psi(t)\, dt = 0$ pour $0 \leq k \leq m$.

Les conditions (a), (b) et (c) expriment respectivement la régularité, la localisation et le caractère oscillant de l'ondelette. Aussi impose-t-on souvent des contraintes supplémentaires de régularité, de décroissance rapide ou de compacité suivant le besoin. La condition d'admissibilité est relativement souple, un assez grand nombre de fonctions peuvent convenir et la propriété suivante simplifie énormément cette condition dans certains cas :

1.2.3. Définition 3:

Soit $\psi \in L^1(R)$:

1- Si ψ est admissible, alors $\hat{\psi}(0) = \int_{-\infty}^{+\infty} \psi(t)dt = 0.$

2- Si $\hat{\psi}(0) = 0$, $\psi \in L^2(R)$ et si la dérivée de $\widehat{\psi}$ est bornée, alors ψ est admissible.

On considère la famille $\psi(\mu, s)$ $s > 0$, $\mu \in R$ obtenue par translation et dilatation de l'ondelette analysante ψ et on définit $\psi_{u,s}$ par :

$$\psi_{u,s}(t) = \frac{1}{\sqrt{s}} \psi\left(\frac{t-u}{s}\right)$$

-Le facteur $\frac{1}{\sqrt{s}}$ est un facteur de normalisation qui assure que $\psi_{u,s}$ a une norme L^2 égale à 1.

-La fonction ψ est appelée ondelette mère et les $\psi_{u,s}$ sont **les ondelettes**.

On peut classer les transformées en ondelettes selon la famille à laquelle appartiennent les fonctions analysantes choisies. Les transformées obtenues sont suivant les cas **discrètes** ou **continues**, **redondantes** ou non.

1.3. Transformée en ondelettes continue:

La transformée en ondelettes d'une fonction est une représentation de cette fonction sur la base d'ondelettes définie précédemment. Ainsi, de même que la TF continue qui est une décomposition atomique d'atomes e^{iwt}, la TO continue (ou TOC) est une décomposition d'atomes $\psi_{u,s}$ comme l'exprime la définition suivante :

$$T^{ond} f(\mu, s) = \frac{1}{\sqrt{s}} \int_{-\infty}^{+\infty} f(t) \quad \psi\left(\frac{t-u}{s}\right) dt \quad --(8)$$

Les transformées continues sont obtenues en prenant le facteur d'échelle s et le pas de translation μ dans l'ensemble des nombres réels.

On peut interpréter cette expression comme une projection du signal sur la famille de fonctions analysantes $\psi_{u,s}$ construite à partir d'une fonction "mère". $T^{ond} f(\mu, s)$ représente donc la corrélation (produit scalaire) de f avec $\psi_{u,s}$ et qui est appelé coefficient d'ondelettes.

Le facteur d'échelle s et le pas de translation μ sont des réels et la transformation en ondelettes est continue et donc redondante. Le plan temps fréquence est sur-analysé.

Il est donc évident qu'une discrétisation de la transformée doit être envisagée si on souhaite obtenir une transformation non redondante. Le pavage temps-fréquence obtenu par la transformation en ondelettes (voir Fig.II.5) suggère une méthode de discrétisation exponentielle pour les échelles et pour le temps. On posera donc :

$$s = {s_0}^m \ \ et \ \ \mu = n\mu_0 {s_0}^m \quad avec\ s_0, \mu_0 \in Z \qquad --(9)$$

L'expression de la transformée en ondelettes discrète est donnée par :

$$T^{ond} f(m,n) = {s_0}^{-\frac{m}{2}} \int_{-\infty}^{+\infty} f(t) \ \ \psi({s_0}^{-m} t - n u_0) dt \qquad --(10)$$

Dans le cas où on choisit s_0 = 2 et μ_0 = 1, alors on parle de transformée dyadique.

L'approche discrète a le mérite de traiter le problème de l'échantillonnage de l'espace temps-fréquence avec rigueur et elle fournit une mesure de l'éventuelle redondance de la transformation obtenue. De plus, dans ce cas, les algorithmes de calcul conduisent souvent à des résultats exacts sur des intervalles donnés de l'espace temps-fréquence. Nous étudierons plus en détail le cas des transformées discrètes qui sont d'ailleurs pratiquement les seules utilisées en traitement d'images.

Parmi les transformées discrètes on distingue les transformées redondantes, dont les trames d'ondelettes et les transformées non redondantes parmi les plus utilisées, il y a les bases orthogonales et biorthogonales. Les paquets d'ondelettes peuvent appartenir suivant le cas à l'une ou l'autre famille.

Pour résumer, on peut donner le classement sommaire suivant des différents types d'ondelettes **[12]** :

Transformées redondantes :	***Transformées non redondantes :***
Transformée continue,	*Analyse multirésolution : base orthonormée,*
Trame d'ondelettes,	*Analyse multirésolution : base biorthogonale,*
Paquet d'ondelettes.	*Paquet d'ondelettes.*

2. Transformée en ondelettes discrète :

2.1. Développement de l'algorithme de la Transformée en Ondelettes Discrètes :

On a vu que la transformée de Fourier permet d'extraire la composante spectrale d'un signal continu mais elle est incapable d'identifier la dépendance temps-fréquence du signal.

La transformée de Fourier à fenêtre glissante a été développée pour extraire la composante spectrale d'un signal dans un certain intervalle de temps déterminé par une fenêtre; Cependant elle est limitée dans la gamme de fréquence qu'elle peut analyser en raison de la taille fixe des fenêtres utilisées. La fenêtre doit être variable avec la fréquence, de tel sorte qu'elle puisse zoomer en avant pour mesurer les hautes fréquences et zoomer en arrière pour mesurer les basses fréquences.

C'est dans cette optique qu'a été développé la transformée en ondelettes ainsi que les fonctions ondelettes générées par dilatation et translation d'une fonction unique (ondelette mère) **[6] [7]** selon l'équation suivante :

$$\psi_{a,b}(t) = a^{-1/2}\ \psi\left(\frac{t-b}{a}\right) \quad a \in R^{+}, b \in R \qquad ---(11)$$

Le développement de ces fonctions sert de base de fonctions pour la transformée en ondelettes continue (TOC) donnée par :

$$W_{a,b}(t) = \ TOC_{f\ a,b}(t) = \int_{-\infty}^{+\infty} f(t) \quad \psi_{a,b}(t)\ \ dt \qquad --(12)$$

Afin de développer la transformée en ondelette discrète (TOD), les fonctions ondelettes doivent être discrétisées; les fonctions ondelettes de l'équation (11) seront modifiées avec les relations suivantes :

$$a = a_0^j\ ,\ \ b = n\ b_0\ a_0^j; \quad j,n\ \in Z, \quad a_0 > 1, \quad b_0\ \neq 0 \qquad --(13)$$

Les valeurs de a_0 et b_0 sont fixées arbitrairement à 2 et à 1 ; dans ce cas on parle de **transformée dyadique.** i.e.: $a = 2^j\ \ et\ b = n\ 2^j$ $--(14)$

Les fonctions ondelettes de l'équation (11) deviennent :

$$\psi_{j,n}(t) = \sqrt{2^{-j}}\ \ \psi\left(2^{-j}t - n\right) \qquad ---(15)$$

La transformée en ondelettes discrète est alors donnée par :

$$TOD_f\ (j,n) = \int_{-\infty}^{+\infty} f(t)\ \ \sqrt{2^{-j}}\ \ \psi\left(2^{-j}t - n\right)\ \ dt\ \ \in\ L^2\ \ --(16)$$

L'analyse multirésolution se base sur la capacité à reconstruire $f(t)$ en partie et non au complet. Quand $f(t)$ est représentée dans différentes résolutions, le signal reconstruit contient différentes bandes de fréquences du signal original $f(t)$. La représentation basse résolution de $f(t)$ est construite avec la fonction échelle $\phi(t)$. Cette représentation de $f(t)$ contient tout le spectre de fréquences allant jusqu'à une certaine fréquence de coupure, ce qui est similaire à un filtrage passe bas de $f(t)$.

Un espace vectoriel basse résolution $V_j \subset L^2\,(R)$ est créé pour contenir toutes les fonctions avec la résolution j, $j\ \in Z$. Plus j augmente, plus l'espace vectoriel basse résolution V_j contient plus de détails. En d'autre mots plus j augmente plus V_j contient des fonctions dont la composante spectrale est de fréquence élevée.

En conséquence, des sous-espaces concentriques se forment avec le sous espace V_i contenant tous les sous-espaces V_j de telle sorte que $i < j$.

$$\text{i.e.:}\ \ \forall\, j \in Z,\quad V_j \subset V_{j-1}\ --(17)$$

Egalement, V_j ont la propriété que pour chaque fonction $f(t) \in V_j$, une version contractée est contenue dans le sous espace V_{j-1}

$$\forall j \in Z,\qquad f(t) \in V_j \Leftrightarrow f\left(\frac{t}{2}\right) \in V_{j-1}\ --(18)$$

Une fonction échelle unique $\phi\ (t) \in V_0$ est crée, ces translations $\phi\ (t-n), n \in Z$, forment la base orthonormée pour V_0, en conséquence la base orthonormée pour V_j est :

$$\phi_{\ j,n}(t) = \sqrt{2^{-j}}\,\phi\left(2^{-j}t - n\right), n \in Z\ --(19)$$

Que A_j soit un opérateur dans $f(t) \in L^2(R)$, qui crée une projection orthonormée de $f(t)$ sur le sous espace V_j :

$$A_j f(t) = \sum_{n=-\infty}^{+\infty}\ < f(u)\phi_{\ j,n}(u) > \phi_{\ j,n}(t)\ --(20)$$

$$\text{Ou :} \quad < f(u), \phi_{j,n}(u) \geq \int_{-\infty}^{+\infty} f(u)\, \phi_{j,n}(u) du \; --(21)$$

Cette projection est similaire à la projection d'un vecteur à 3 dimensions sur un plan à 2 dimensions ; le vecteur à 2 dimensions résultant de la projection devient la représentation la plus proche du vecteur à 3 dimensions. De même $A_j f(t)$ est la plus proche représentation de $f(t)$ dans les sous espaces V_j.

Que A_j^d soit un opérateur sur $f(t)$ qui donne le produit scalaire discret de l'équation (20)

$$A_j^d f(n) =< f(t), \phi_{j,n}(t) >= \sqrt{2^{-j}} < f(t)\phi(2^{-j}(t - 2^{+j}n)) > n \in Z \; --(22)$$

Le produit de convolution est défini par :

$$f^* g(x) = \big(f(t)^* g(t)\big)(x) = \int_{-\infty}^{+\infty} f(t)\, g(x-t) dt \; --(23)$$

L'équation (22) peut être écrite comme :

$$A_j^d f(n) = \sqrt{2^{-j}} \int_{-\infty}^{+\infty} f(t)\, \phi\left(2^{-j}\left(t - 2^j n\right)\right) dt \; --(24)$$

$$A_j^d f(n) = \left(f(t) \; * \sqrt{2^{-j}}\, \phi\left(-2^{-j} t\right)\right) \; \left(2^j n\right), \qquad n \in Z \; --(25)$$

$A_j^d f(n)$ peut être vu comme une convolution de $f(t)$ avec $\phi(t)$ échantillonnée uniformément à intervalles de $2^{-j}n$. Plus la fonction échelle devient contractée, ce qui correspond aux hautes fréquences, plus le taux d'échantillonnage décroit.

L'équation (25) représente la discrétisation du filtrage passe bas de $f(t)$ qui correspond à une approximation discrète de $f(t)$.

La première discrétisation de l'approximation de $f(t)$ qui contient la plus part des informations est donnée par $A_0^d f(n)$ (i.e. résolution 0) pour des buts de normalisation. Les résolutions supérieures de la version discrète de $f(t)$ c.à.d. ($A_{-1}^d f(n)$, $A_{-2}^d f(n)$, ……) contient plus d'information de $f(t)$.

On peut passer d'une fonction haute résolution à une fonction basse résolution puisque $\phi_{j,n}(t)$ est un membre des fonctions de V_{j-1}.

$$\text{i.e.:} \quad \phi_{j,n}(t) \in V_j \subset V_{j-1} \qquad --(26)$$

En conséquence, on peut construire $\phi_{j,n}(t)$ avec la base orthonormale de V_{j-1} :

$$\phi_{j,n}(t) = \sum_{n=-\infty}^{+\infty} < \phi_{j,n}(u), \phi_{j-1,k}(u) > \phi_{j-1,k}(t) \qquad --(27)$$

Le produit scalaire de (55) peut être écrit comme ceci :

$$< \phi_{j,n}(u), \phi_{j-1,k}(u) > = 2^{-j}\sqrt{2} \int_{-\infty}^{+\infty} \phi(2^{-j}u - n)\ \phi(2^{-j+1}u - k)\, du \quad --(28)$$

On faisant le changement de variable suivant : $\left(\frac{t}{2} = 2^{-j}u - n\right)$ dans l'équation précédente, on obtient :

$$\int_{-\infty}^{+\infty} \frac{2^{-j}\sqrt{2}}{2^{-j}\,2} \phi(t/2)\ \phi\left(\frac{2^{-j+1}}{2^{-j+1}}(t + 2n) - k\right) dt = \int_{-\infty}^{+\infty} \sqrt{2^{-1}}\phi(2^{-1}t)\ \phi(t - (k - 2n)\, dt$$

$$= < \phi_{-1,0}(t), \phi_{0,k-2n}(t) >$$

$$\int_{-\infty}^{+\infty} \sqrt{2^{-1}}\phi(2^{-1}t)\ \phi(t - (k - 2n)\, dt = < \phi_{-1,0}(t), \phi_{0,k-2n}(t) > \quad --(29)$$

On multiplie en suite $f(t)$ aux deux extrémités de l'équation (27), on obtient :

$$< f(t), \phi_{j,n}(t) > = \sum_{n=-\infty}^{+\infty} < \phi_{-1,0}(u), \phi_{0,k-2n}(u) >< f(t), \phi_{j-1,k}(t) > -(30)$$

On définit la réponse impulsionnelle d'un filtre discret $H(\omega)$:

$$\forall\, n \in Z, \qquad h(n) = < \phi_{-1,0}(u), \phi_{0,n}(u) > \quad ---(31)$$

Et $\tilde{H}(\omega)$ est le filtre miroir dont la réponse impulsionnelle $\tilde{h}(n) = h(-n)$; L'équation (30) devient :

$$A_j^d f(n) = \sum_{n=-\infty}^{+\infty} \tilde{h}(2n - k)\, A_{j-1}^d f(k) \qquad ---(32)$$

$$A_j^d f(n) = \tilde{h}(k) \quad * \quad A_{j-1}^d f(k)(2n) \qquad ---(33)$$

L'approximation discrète $A_j^d f(n)$ à la résolution j peut être calculée à partir de l'approximation discrète de haute résolution suivante en faisant la convolution $A_{j-1}^d f(n)$ par un filtre miroir $\widetilde{H}(\omega)$. Le filtre $H(\omega)$ est l'unique caractéristique de la fonction échelle.

W_j, définit la différence entre deux espaces vectoriels basse résolution V_j et V_{j-1}, en d'autre termes, l'union de W_j et V_j donne V_{j-1}. L'espace W_j a la propriété intéressante d'être orthogonale à V_j c.à.d. $W_j \perp V_j$. Comme conséquence de cette orthogonalité, toute fonction de V_{j-1} peut être écrite uniquement comme somme d'une fonction de V_j et d'une fonction de W_j.

$$V_{j-1} = V_j \oplus W_j \qquad ---(34)$$

L'espace V_j contient les fonctions de basse résolution et W_j contient les fonctions de la gamme voisine. Les fonctions échelles sont utilisées pour construire une base orthogonale de V_j et les fonctions ondelettes sont utilisées pour construire une base orthogonale de W_j .

c.à.d.:

$$\phi_{j,n}(t) = \sqrt{2^{-j}}\,\phi\,(2^{-j}t - n), \quad \forall n \in Z \qquad forme\ une\ base\ orthogonale\ dans\ V_j$$
$$\psi_{j,n}(t) = \sqrt{2^{-j}}\,\psi\,(2^{-j}t - n), \quad \forall n \in Z \qquad forme\ une\ base\ orthogonale\ dans\ W_j$$

Les fonctions ondelettes et les fonctions échelles sont de prés liées, tel que des relations restrictives peuvent être calculées et qui peuvent aider aux développements de différentes fonctions échelles et ondelettes.

De l'équation (27), on peut écrire :

$$\phi_{j,n}(t) = \sum_{n=-\infty}^{+\infty} < \phi_{-1,0}(u), \phi_{0,k-2n}(u) > \ \phi_{j-1,k}(t)$$

$$\Longrightarrow \qquad \phi_{j,n}(t) = \sum_{n=-\infty}^{+\infty} h(k-2n)\ \phi_{j-1,k}(t) \qquad --(35)$$

On définit $g(n)$ comme la réponse impulsionnelle du filtre discret $G(\omega)$:

$$\forall\, n \in Z, \qquad g(n) = < \psi_{-1,0}(u), \psi_{0,n}(u) > \qquad ---(36)$$

Une relation similaire à (27) peut être écrite pour ondelette $\psi_{j,n}(t) \in V_{j-1}$.

i.e.: $$\psi_{j,n}(t) = \sum_{n=-\infty}^{+\infty} < \psi_{-1,0}(u), \psi_{0,k-2n}(u) > \ \phi_{j-1,k}(t)$$

$$\Longrightarrow \qquad \psi_{j,n}(t) = \sum_{n=-\infty}^{+\infty} g(k-2n)\ \phi_{j-1,k}(t) \qquad --(37)$$

Les filtres $H(\omega)$ et $G(\omega)$ peuvent être déterminés à partir des fonctions échelles et ondelettes et vice versa. Cependant les filtres discrets et les fonctions ondelettes doivent satisfaire les relations contraignantes mentionnées dans [Malt89]. Ces relations permettent aux mathématiciens le développement des différentes ondelettes.

La TOD est basé sur la décomposition de $A_{j-1}^d f(n)$ en deux séquences $A_j^d f(n)$ et $D_j^d f(n)$. La première séquence a été développée précédemment, la deuxième séquence va être étudiée maintenant.

Multipliant les deux extrémités de l'équation précédente par $f(t)$, on obtient alors :

$$< f(t), \psi_{j,n}(t) > = \sum_{n=-\infty}^{+\infty} g(k-2n) \ < f(t), \phi_{j-1,k}(t) >$$

$$D_j^d f(n) = \sum_{n=-\infty}^{+\infty} \tilde{g}(2n-k)\, A_{j-1}^d f(k) \qquad ---(38)$$

$$D_j^d f(n) = \tilde{g}(k) \ * \ A_{j-1}^d f(k)(2n) \qquad ---(39)$$

Ces relations sont comparables aux équations (32) et (33), cette séquence peut aussi s'écrire sous forme de l'équation (25) :

$$D_j^d f(n) = \sqrt{2^{-j}} \int_{-\infty}^{+\infty} f(t)\ \ \psi\left(2^{-j}\left(t - 2^j n\right)\right) dt$$

$$D_j^d f(n) = \left(f(t) \ * \sqrt{2^{-j}}\, \psi\left(-2^{-j} t\right)\right) \ \left(2^j n\right), \qquad n \in Z \ --(40)$$

$D_j^d f(n)$ peut-être aussi vu comme une version de $f(n)$ filtré par un filtre passe haut et échantillonné à intervalle de $2^j n$.

$A_j^d f(n)$ fait référence à l'approximation de $f(t)$ et $D_j^d f(n)$ fait reference aux détails de $f(t)$.

De l'équation (33) et (39) on peut représenter graphiquement la décomposition de $A_{j-1}^{d} f(n)$ avec les bancs de filtres comme le montre la figure (II.6) :

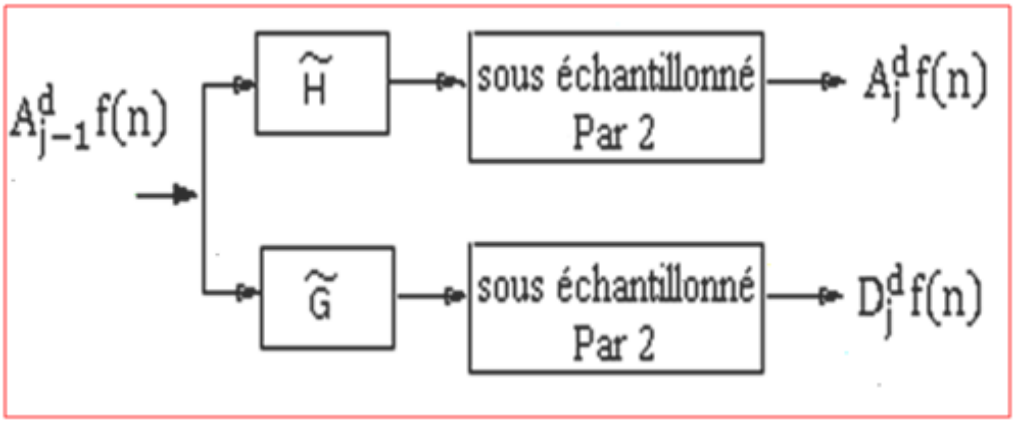


Fig.II.6 – Transformée en Ondelettes Discrète.

Une fois la séquence $A_0^d f(n)$ à été décomposée en j séquences détails et j séquences lisses (approximation) ; le signal original peut être reconstruit à partir des séquences obtenues par décomposition.

V_{j-1} peut être écrit uniquement comme la somme des fonctions de V_j et des fonctions de W_j :

$$\phi_{j-1,n}(t) = \sum_{n=-\infty}^{+\infty} < \phi_{j,k}(u), \phi_{j-1,n}(u) > \phi_{j,k}(t) + \sum_{n=-\infty}^{+\infty} < \psi_{j,k}(u), \phi_{j-1,n}(u) > \psi_{j,k}(t)$$

$$\Longrightarrow \quad \phi_{j-1,n}(t) = \sum_{n=-\infty}^{+\infty} h(n-2k)\ \phi_{j,k}(t) + \sum_{n=-\infty}^{+\infty} g(n-2k)\ \psi_{j,k}(t)$$

On multiplie en suite les deux extrémités de cette équation par $f(t)$ et on obtient :

$$A_{j-1}^{d} f(n) = \sum_{n=-\infty}^{+\infty} h(n-2k)\, A_j^d f(k) + \sum_{n=-\infty}^{+\infty} g(n-2k)\, D_j^d f(k) \qquad -(41)$$

On insérant des zéros entre chaque échantillons de $A_j^d f(k)$ et de $D_j^d f(k)$, on sur-échantillonne ces deux séquences :

$$A_j^{du} f(2k) = A_j^d f(k) \quad ; D_j^{du} f(2k) = D_j^d f(k)$$

On pose m=2k et on obtient :

$$A_{j-1}^d f(n) = h(m) * A_j^{du} f(m)(n) + g(m) * D_j^{du} f(m)(n) \qquad -(42)$$

$A_{j-1}^d f(n)$ est reconstruit à partir des séquences $A_j^d f(n)$ et $D_j^d f(n)$ sur-échantillonne à qui on fait une convolution avec les filtres $H(\omega)$ et $G(\omega)$ puis sont additionnés comme le montre la figure (II.7).

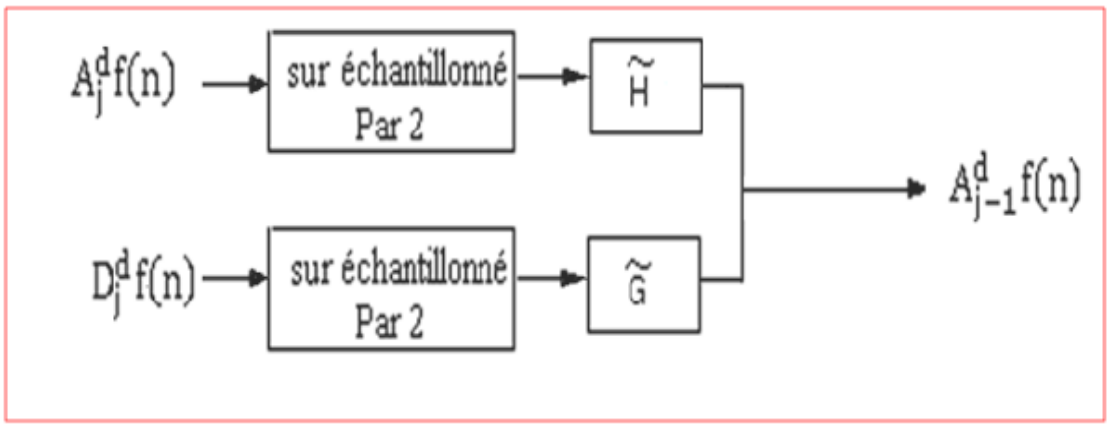


Fig.II.7 – Transformée en Ondelettes Discrète Inverse.

Nous venons de voir le développement théorique de la TOD 1D (algorithme de Mallat) **[6];** et nous avons vu que la TOD 1D revient à appliquer un filtre passe bas et un filtre passe haut au signal. Dans ce qui suit nous allons faire l'extension de la TOD aux données à trois dimensions afin de pouvoir l'appliquer aux cubes hyperspectraux.

2.2. Extension de la TOD aux signaux à plusieurs dimensions :

Signaux à deux dimensions :

Afin que la TOD puisse être exploité sur les images, on doit l'étendre à deux dimensions ; Comme discuter auparavant l'espace vectoriel V_j est une approximation multirésolution de $L^2(R)$. De même, on définit l'espace vectoriel V_j^2 comme une approximation multirésolution de $L^2(R^2)$. En plus, on définit V_j^2 comme approximation séparable

de $L^2(R^2)$, ou V_j^2 peut être décomposé en un produit de deux espaces vectoriels $V_j^2 = V_j \times V_j$. Chaque espace vectoriel unidimensionnel est une approximation multirésolution de $L^2(R)$. La fonction échelle de V_j^2 peut être séparée en deux fonctions échelles unidimensionnelles.

$$\phi(x,y) = \phi(x)\ \ \phi(y) \qquad ----(43)$$

Ou : $\phi(x)$ est la fonction échelle unidimensionnelle de V_j .

L'espace vectoriel qui lie entre différente résolution à deux dimensions est défini en tant que W_j^2. Une base orthogonale pour W_j^2 peut être construite en utilisant trois versions de fonctions ondelettes :

$$\psi^1(x,y) \quad ; \ \psi^2(x,y) \ ; \quad \psi^3(x,y) \quad ----(44)$$

Ces fonctions ondelettes peuvent être séparées en des fonctions échelles et des fonctions ondelettes unidimensionnelles comme suit :

$$\psi^1(x,y) = \phi(x)\ \ \psi(y) \quad ; \ \psi^2(x,y) = \psi(x)\,\phi(y) \ ; \quad \psi^3(x,y) = \psi(x)\ \psi(y)$$

Même développement que le cas 1D, on note A_j^d l'operateur de la fonction échelle et D_j^{d1}, D_j^{d2}, D_j^{d3} les operateurs des trois fonctions ondelettes.

Ces operateurs effectuent une discrétisation du filtrage de $f(x,y)$, le long des axes X et Y. Avec le même développement que le cas 1D, ces opérateurs peuvent être écris comme une décomposition de $A_{j-1}^d f(n,m)$ en quatre séquences à deux dimensions (quatre sous images).

$$A_j^d f(n,m) = \left(\tilde{h}(k)\ \tilde{h}(l) * A_{j-1}^d f(k,l)\right)(2n,2m) - (45)$$
$$D_j^{d1} f(n,m) = \left(\tilde{h}(k)\ \tilde{g}(l) * A_{j-1}^d f(k,l)\right)(2n,2m) - (46)$$
$$D_j^{d2} f(n,m) = \left(\tilde{g}(k)\ \tilde{h}(l) * A_{j-1}^d f(k,l)\right)(2n,2m) - (47)$$
$$D_j^{d3} f(n,m) = \left(\tilde{g}(k)\ \tilde{g}(l) * A_{j-1}^d f(k,l)\right)(2n,2m) - (48)$$

Quand la transformée 2D est séparable alors $A_{j-1}^d f(n,m)$ est séparable dans les directions n et m ; la transformée 2D peut alors être écrite comme une cascade de deux transformées

unidimensionnel (1D) dans les deux directions qui sont perpendiculaires l'une à l'autre. Par conséquent, **la TOD 2D d'une image digitale finie peut être définie comme une cascade d'une TOD 1D sur les lignes et d'une TOD 1D sur les colonnes de l'image.**

Quand une image est décomposée en un niveau, les quatre séquences (sous-images) obtenues contiennent différents types d'informations sur les fréquences des lignes et des colonnes. La sous-image basse résolution apparait comme une version sous échantillonnée reconnaissable de l'image originale ; Les autres sous-images sont moins reconnaissables avec des informations hautes fréquences soit des lignes, soit des colonnes, soit des deux.

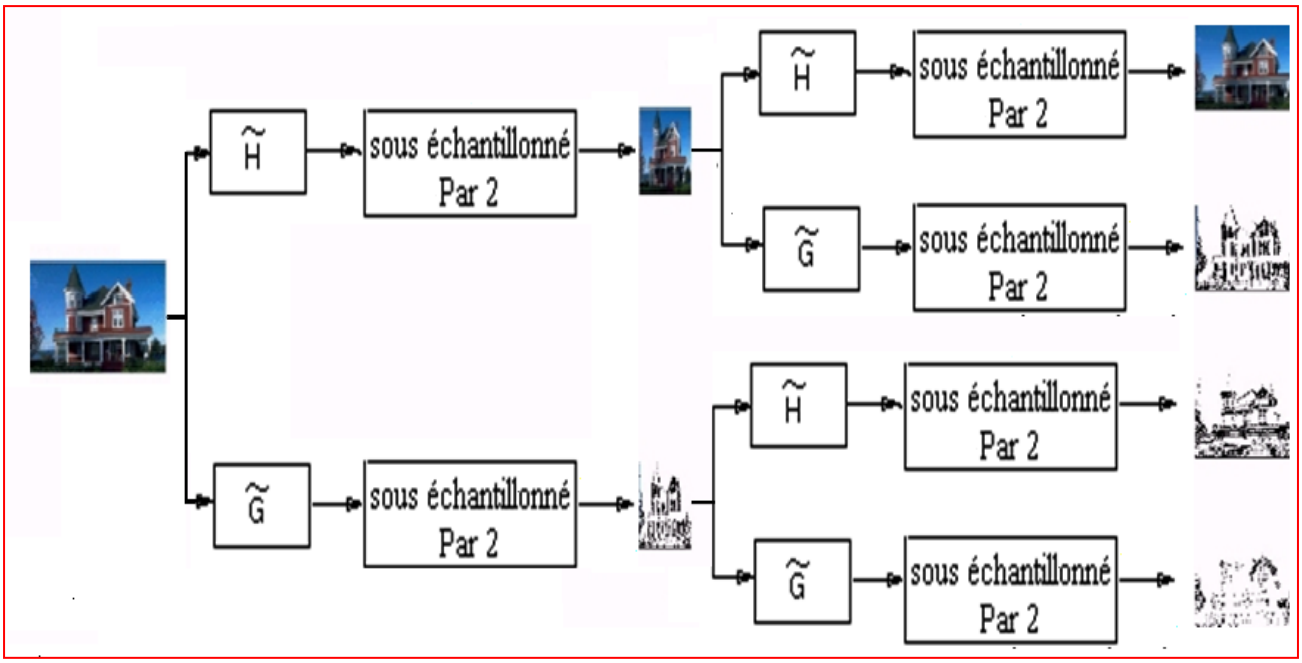


Fig.II.8 – Transformée en Ondelettes 2D.

Comme on vient de le voir la TOD peut être étendue à plusieurs dimensions en utilisant des filtres séparables, ou chaque dimension est filtrée et échantillonnée séparément. Par conséquent, la TOD peut être appliquée à une image 3D en utilisant des filtres séparables comme dans le cas des images 2D. Nous allons dans ce qui suit présenter l'implémentation de la TOD 3D sous Matlab en utilisant des filtres séparables.

3. L'algorithme de la TOD 3D sous Matlab :

La TOD peut être étendue à plusieurs dimensions en utilisant des filtres séparables, ou chaque dimension est filtrée et échantillonnée séparément. Pour la TOD 3D cela revient à effectuer une cascade de TOD 1D sur les trois axes x, y et z, puis répéter l'opération sur le signal 'approximation' obtenu pour obtenir la TOD à plusieurs niveaux (échelles). Après une TOD 3D nous obtenons 8 blocs de données, le bloc LLL concentre la plus grande partie des données de départ, les 7 autres blocs contiennent les blocs détails.

Dans ce qui suit nous allons implémenter la TOD 3D sous Matlab, pour ce, nous allons commencer par implémenter les bancs de filtres.

3.1. Banc de filtres 1D :

L'analyse en banc de filtres décompose le signal d'entrée x (n) en deux signaux c (n) et d (n). Le signal c (n) représente l'approximation (basses fréquences) de x (n), alors que le signal d (n) représente les détails (hautes fréquences) de x (n). L'analyse en banc de filtres, filtre x (n) en utilisant un filtre passe-bas et un filtre passe-haut. On note le filtre passe-bas par AF1 (filtre d'analyse 1) et le filtre passe-haut par AF2 (filtre d'analyse 2). Comme le montre la figure (II.9), la sortie de chaque filtre est ensuite sous-échantillonné par 2 pour obtenir les deux signaux sous-bandes, c (n) et d (n).

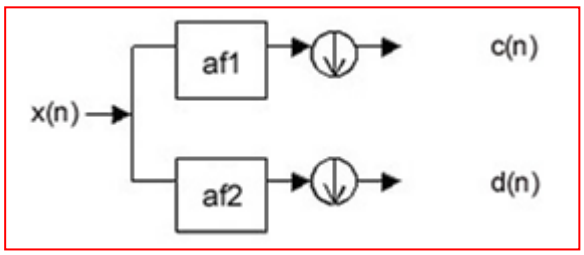


Fig.II.9 – Analyse en banc de filtres 1D.

Nous allons implémenter l'analyse en banc de filtres 1D sous Matlab dans le script **(abf1D.m).** Le programme utilise la fonction Matlab **upfirdn** (dans la boîte à outils Signal Processing), pour mettre en œuvre le filtrage et le sous-échantillonnage :

Table 1: fonction abf1D.m

```
function [lo, hi] = abf1D(x, af)
% x – vecteur (pair)
% af : filtres d'analyse
% af(:, 1): filtre passe-bas
% af(:, 2): filtre passe-haut
%
% lo: signal basse fréquence
% hi: signal Haute fréquence

N = length(x);
L = length(af)/2;

% filter passe-bas
lo = upfirdn(x, af(:,1), 1, 2);
lo(1:L) = lo(N/2+[1:L]) + lo(1:L);
lo = lo(1:N/2);

% filter passe-haut
hi = upfirdn(x, af(:,2), 1, 2);
hi(1:L) = hi(N/2+[1:L]) + hi(1:L);
hi = hi(1:N/2);
```

A l'aide de la fonction upfirdn, on convolue le signal d'entrée x(n) avec chacun des filtres, puis on sous-échantillonne le signal obtenu (c.à.d. on prend un échantillon sur deux). De plus, le signal d'entrée x (n) est de longueur N. Alors, pour plus de commodité, il faut avoir les deux signaux sous-bande c (n) et d (n) chacun de longueur N / 2. Cependant, ces deux signaux vont dépasser cette longueur par L / 2 où L est la longueur des filtres d'analyse.

Pour éliminer cette durée excessive, les dernier échantillons (L / 2) de chaque sous-bande sont ajoutés aux premiers (L / 2) échantillons. Cette procédure (extension périodique) peut créer des artefacts indésirables au début et à la fin des signaux sous-bandes. Cependant, c'est la solution la plus commode.

Lorsque les filtres d'analyse et les filtres de synthèse sont exactement symétriques, une procédure différente (extension symétrique) peut être utilisée, qui évite les artefacts liés à l'extension périodique.

La synthèse en banc de filtres combine les deux signaux sous-bande c (n) et d (n) pour obtenir un seul signal y (n). Les filtres de synthèse en premier sur-échantillonnent chacun des deux signaux sous-bandes. Les signaux sont ensuite filtrés par un filtre passe-bas et un filtre passe-haut. On note le filtre passe-bas par SF1 (filtre de synthèse 1) et le filtre passe-haut par SF2 (filtre de synthèse 2). Les signaux sont additionnés pour obtenir le signal y (n).

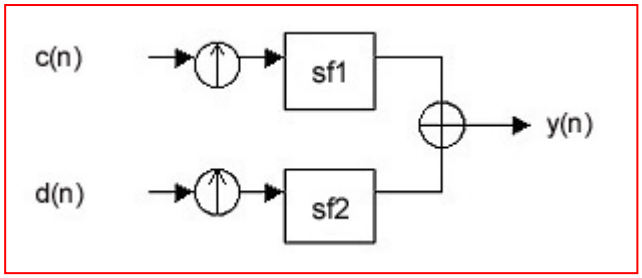


Fig.II.10 – synthèse en banc de filtres 1D.

La synthèse en banc de filtres 1D est implémenté dans le script **(sbf1D.m)** suivant:

Table 2: fonction sbf1D.m

```
function y = sbf1D(lo, hi, sf)
% lo - signal basse fréquence
% hi - signal Haute fréquence
% sf - filtres de synthèses
% sf(:, 1) - filtre passe-bas
% sf(:, 2) - filtre passe-haut
%
% y - Sortie

N = 2*length(lo);
L = length(sf);

lo = upfirdn(lo, sf(:,1), 2, 1);
hi = upfirdn(hi, sf(:,2), 2, 1);
y = lo + hi;
y(1:L-2) = y(1:L-2) + y(N+[1:L-2]);
y = y(1:N);
```

Il existe de nombreux ensembles de filtres qui peuvent être utilisés, nous allons citer ici comme exemple *Debauchies 8* :

g (af1)	h (af2)	rg (sf1)	rh (sf1)
-0,230377813308897	-0,010597401785069	-0,010597401785069	0,230377813308897
0,714846570552916	0,032883011666885	0,032883011666885	0,714846570552916
-0,630880767929860	0,030841381835560	0,030841381835560	0,630880767929860
-0,027983769416859	-0,187034811719093	-0,187034811719093	-0,027983769416859
0,187034811719093	-0,027983769416859	-0,027983769416859	-0,187034811719093
0,030841381835560	0,630880767929860	0,630880767929860	0,030841381835560
-0,032883011666885	0,714846570552916	0,714846570552916	0,032883011666885
-0,010597401785069	0,230377813308897	0,230377813308897	-0,010597401785069

Tableau.II.1 Filtres de Debauchies 8.

Pour utiliser la transformée en ondelettes pour les données 3D, nous devons mettre en œuvre une version 3D du banc de filtres.

3.2. Banc de filtres 3D :

Dans le cas 3D, l'analyse en Banc filtre 1D est appliquée à tour de rôle à chacune des trois dimensions (c.à.d. pour les trois axe x, y et z). Si les données sont de taille N1*N2*N3, nous devons appliquer l'analyse 1D à la première dimension. Nous obtenons deux ensembles de données, chacun de taille N1/2 par N2 par N3. Ensuite, appliquer l'analyse 1D à la deuxième dimension, nous obtenons quatre ensembles de données, chacun de taille N1/2 par N2/2 par N3. Pour finir, l'application de l'analyse 1D à la troisième dimension donne huit ensembles de données, chacun de taille N1/2 par N2/2 par N3/2. Ceci est illustré dans le schéma ci-dessous.

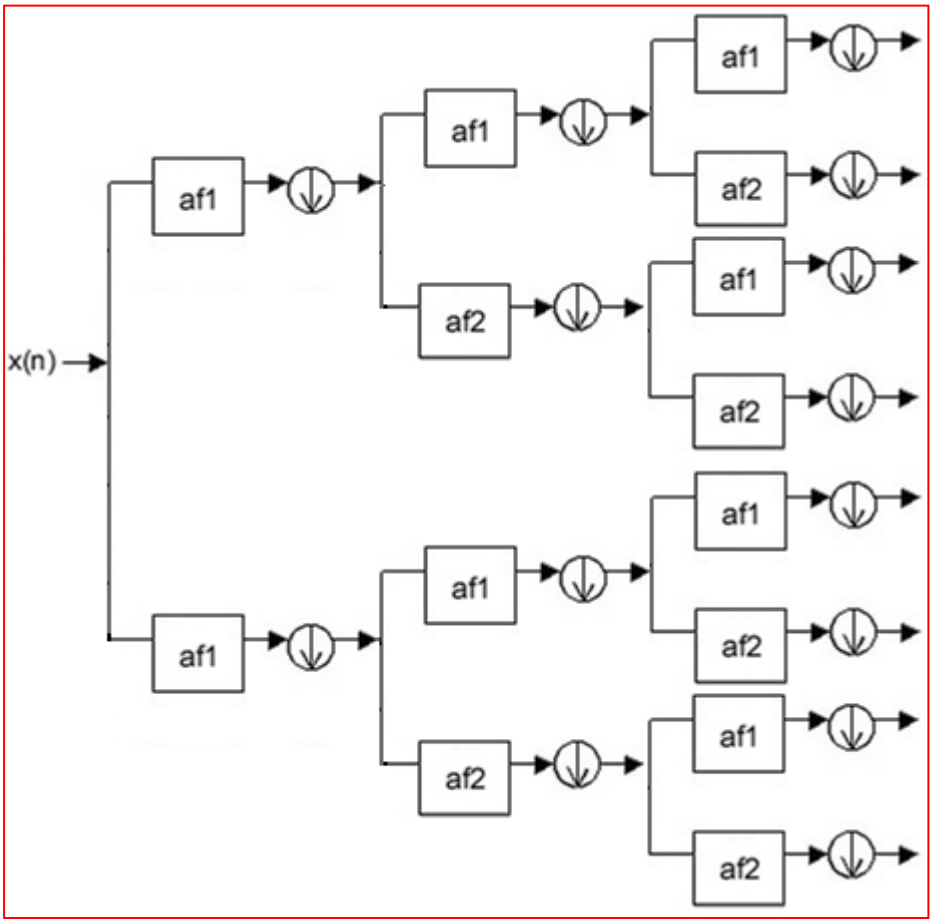


Fig.II.11 – Analyse en Banc filtre 3D.

L'analyse en banc de filtre 3D est implémentée avec la fonction **(abf3D.m).** Cette fonction fait appel à une sous-fonction **(abf1D-M.m),** qui applique une analyse en banc filtre 1D le long d'une seule dimension. Cette sous fonction est en fait la fonction **(abf1D.m)** qu'on a vu précédemment, mais modifie de tel sorte qu'elle prenne en compte la dimension à laquelle elle est appliquée, comme on va le voir dans le script suivant :

Table 3: fonction abf1D-M.m

```
function [lo, hi] = abf1D-M(x, af, d)
% Entrée:
% x - matrice N1xN2xN3, ou: min(N1,N2,N3) > 2*length(filtre)
% (Ni sont pair)
% af - filtres d'analyses pour les colonnes
% af(:, 1) - filtre passe-bas
% af(:, 2) - filtre passe-haut
% d - dimension filtrée (d = 1, 2 or 3)
% Sortie:
% lo, hi – signal  passe-bas, passe-haut

lpf = af(:, 1); % filtre passe-bas
hpf = af(:, 2); % filtre passe-haut

%Permuter les dimensions de x tel que la dimension d est la première
p = mod(d-1+[0:2], 3) + 1;
x = permute(x, p);

% filtrage le long de la  dimension 1
[N1, N2, N3] = size(x);
L = size(af, 1)/2;

for k = 1:N3
    lo(:, :, k) = upfirdn(x(:, :, k), lpf, 1, 2);
end
lo(1:L, :, :) = lo(1:L, :, :) + lo([1:L]+N1/2, :, :);
lo = lo(1:N1/2, :, :);

for k = 1:N3
    hi(:, :, k) = upfirdn(x(:, :, k), hpf, 1, 2);
end
hi(1:L, :, :) = hi(1:L, :, :) + hi([1:L]+N1/2, :, :);
hi = hi(1:N1/2, :, :);

% permutation inverse de x
lo = ipermute(lo, p);
hi = ipermute(hi, p);
```

La fonction **(abf3D.m)** suit le schéma de décomposition donnée dans la figure précédente, elle utilise la fonction **(abf1D-M.m)** dans les trois directions x, y et z pour obtenir les deux variables : **lo** qui est l'approximation de la donnée 3D et **hi** qui est un tableau de cellules contenant les sept autres détails de la donnée 3D.

Table 4: fonction abf3D.m

```
function [lo, hi] = abf3D(x, af)
% Entrée:
% x - matrice N1xN2xN3, ou: min(N1,N2,N3) > 2*length(filtre)
% (Ni sont pair)
% af - filtres d'analyses
% af(:, 1) - filtre passe-bas
% af(:, 2) - filtre passe-haut
% Sortie:
% lo, hi – signal  passe-bas, passe-haut

% filtrage le long de la  dimension 1
[L, H] = abf1D_M(x, af, 1);

% filtrage le long de la  dimension 2
[LL LH] = abf1D_M (L, af, 2);
[HL HH] = abf1D_M (H, af, 2);

% filtrage le long de la  dimension 3
[LLL LLH] = abf1D_M (LL, af, 3);
[LHL LHH] = abf1D_M (LH, af, 3);
[HLL HLH] = abf1D_M (HL, af, 3);
[HHL HHH] = abf1D_M (HH, af, 3);

lo = LLL;
hi{1} = LLH;
hi{2} = LHL;
hi{3} = LHH;
hi{4} = HLL;
hi{5} = HLH;
hi{6} = HHL;
hi{7} = HHH;
```

3.3. Transformé en ondelettes 3D :

La transformée en ondelettes 3D d'une donnée 3D, est mise en œuvre par l'itération de l'analyse en banc de filtre 3D sur l'approximation de la donnée 3D (c.à.d. sur LLL). Dans ce cas, à chaque échelle, il y a sept sous-bandes détails. La fonction suivante **(TOD3D.m),** calcule la TOD 3D d'un cube de données, en appelant à plusieurs reprises la fonction **(abf3D.m),** appliquée à chaque fois à la sous bande approximation (LLL) obtenue dans le passage précédant**.**

Table 5: fonction TOD3D.m

```
function w = TOD3D(x, J, af)
% Entrée:
% x  - matrice N1xN2xN3,
% af - filtres d'analyses
% J  - niveau de décomposition
% Sortie:
% w  - coefficients d'ondelettes

for k = 1:J
    [x w{k}] = abf3D(x, af);
end
w{J+1} = x;
```

W est un tableau de cellules de Matlab, pour j = 1... J, d = 1... 7, w (j) (d) est l'un des sept ensembles de données produits à l'étape j. w (J +1) est la sous-bande passe-bas du cube de données produits à la dernière étape.

Pour la transformée en odelette inverse TODI 3D nous allons suivre la même démarche que pour la TOD 3D et faire appel à trois fonctions, qui sont **(sbf1D-M.m), (sbf3D.m)** et **(TODI3D.m).** Elles sont données dans ce qui suit :

Table 6: fonction sbf1D-M.m

```
function y = sbf1D_M(lo, hi, sf, d)
% sf - filtres de synthèses
% d - dimension filtrée (d = 1, 2 or 3)

lpf = sf(:, 1); % filtre passe-bas
hpf = sf(:, 2); % filtre passe-haut

Permuter les dimensions de x tel que la dimension d est la première
p = mod(d-1+[0:2], 3) + 1;
lo = permute(lo, p);
hi = permute(hi, p);

[N1, N2, N3] = size(lo);
N = 2*N1;
L = length(sf);

for k = 1:N3
    y(:, :, k) = upfirdn(lo(:, :, k), lpf, 2, 1) + upfirdn(hi(:, :,
k), hpf, 2, 1);
end
y(1:L-2, :, :) = y(1:L-2, :, :) + y(N+[1:L-2], :, :);
y = y(1:N, :, :);

% permutation inverse de x
y = ipermute(y, p)
```

Table 7: fonction sbf3D.m

```
function y = sbf3D(lo, hi, sf )
% Entée:
% lo, hi - signal  passe-bas, passe-haut
% sf - filtres de synthèses
% Sortie:
% y - sortie

LLL = lo;
LLH = hi{1};
LHL = hi{2};
LHH = hi{3};
HLL = hi{4};
HLH = hi{5};
HHL = hi{6};
HHH = hi{7};

% filtrage le long de la  dimension 3
LL = sbf1D_M (LLL, LLH, sf, 3);
LH = sbf1D_M (LHL, LHH, sf, 3);
HL = sbf1D_M (HLL, HLH, sf, 3);
HH = sbf1D_M (HHL, HHH, sf, 3);

% filtrage le long de la  dimension 2
L = sbf1D_M (LL, LH, sf, 2);
H = sbf1D_M (HL, HH, sf, 2);

% filtrage le long de la  dimension 1
y = sbf1D_M (L, H, sf1,1);
```

Table 8: fonction TODI3D.m

```
function y = TODI3D(w, J, sf)

% Entrée:
% w  - coefficients d'ondelettes
% J  - niveau de décomposition
% sf - filtres de synthèses
% Sorie:
% y - sortie

y = w{J+1};
for k = J:-1:1
    y = sbf3D (y, w{k}, sf);
end
```

Quantification par Zerotree

CHAPITRE 3

- Préambule.
- L’algorithme EZW.
- L’algorithme SPIHT.
- Extension aux images 3D du SPIHT.
- L’algorithme SPIHT 3D sous MATLAB.

1. Préambule :

La transformée en ondelette permet comme décrit dans le chapitre précédent, de représenter les images sous forme de coefficients ordonnés en bandes de fréquences. Pour la compression d'images, la transformée représente le premier maillon de la chaîne, afin de comprimer l'information, il faut compléter le cycle par la quantification et le codage.

Après la première étape de la transformée, les coefficients produits par l'implantation de la TO 3D peuvent être codés avec ou sans perte. Nous distinguons deux types d'approches pour les standards de compression actuels basés sur la TO : l'approche inter-bandes qui utilise les redondances inter-échelles entre les sous-bandes pour coder les coefficients d'ondelettes et celle intra-bandes dans laquelle les sous-bandes sont codées de façon indépendante.

L'approche inter-bandes exploite complètement la notion de multi-résolution associée aux ondelettes. Leur schéma de codage utilise un modèle simple pour caractériser les dépendances inter-bandes parmi les coefficients d'ondelettes localisés dans les sous-bandes ayant la même orientation. Le modèle est basé sur l'hypothèse des arbres de zéros, ces méthodes appliquent une quantification par approximation successive pour améliorer la précision de la représentation des coefficients d'ondelettes. En effet, les bits les plus significatifs sont transmis en premier. Ainsi, pour augmenter la qualité de l'image reconstruite, il suffira d'ajouter de la précision en utilisant des bits supplémentaires.

EZW est le premier algorithme inter-bande développé pour les images 2D, il a été proposé en 1993 par Jérôme Shapiro. Il permet de transmettre les coefficients sous forme d'une suite de bits, obtenue par enchâssement progressif des bits des coefficients les plus significatifs. En commençant par les bits les plus importants, l'algorithme EZW permet de faire de la transmission progressive d'images, puisque le décodeur peut s'arrêter n'importe où dans la suite de bits transmise et produire la meilleure image reconstruite possible avec cette suite de bits.

Dans ce qui suit, nous allons faire une description assez simple du codage EZW (Embedded Zerotree Wavelet coding) **[6]** proposée par SHAPIRO, ainsi que de la variante la plus populaire de ce codage, qui est le SPIHT (Set Partitionning in Hierarehical Tree) **[7]** réalisé par A.SAID et W.PEARLMAN. Ensuite, nous allons faire l'extension du SPIHT aux images 3D, afin de l'utiliser avec la TOD 3D dyadique développée précédemment.

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

2.1. Introduction au codage EZW :

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 représentation 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.

Le codage EZW exploite complètement la notion de multi-résolution associée aux ondelettes. Quand une image est transformée par ondelettes, l'énergie dans les sous-bandes diminue pendant que l'échelle diminue (haut résolution). Ainsi les coefficients d'ondelettes seront plus petits en moyenne dans les sous-bandes plus hautes que dans les sous-bandes inférieures. De plus les grands coefficients d'ondelettes sont plus importants que les plus petits.

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 rebalayée 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.

Dans la représentation de l'image dans le domaine ondelettes, chaque coefficient peut être considéré en tant qu'ayant quatre descendants dans la prochaine plus haute sous-bande (voir figure (III.1)). Ainsi que pour les quatre descendants, chacun a également quatre fils dans la prochaine plus haute sous-bande et nous voyons un arbre à quatre descendants émerger; chaque racine a quatre branches, ce processus se poursuit jusqu'aux fréquences les plus hautes. Cela est vrai sauf pour le nœud racine qui a seulement 3 enfants, alors que tous

les autres nœuds à l'exception des extrémités en possèdent 4. Le lien parent enfant à 2D est le suivant :

$$O(i, j) = \begin{Bmatrix} (2i, 2j), (2i, 2j + 1), \\ (2i + 1, 2j), \\ (2i + 1, 2j + 1) \end{Bmatrix} \quad --(1)$$

Où O (i, j) représente l'ensemble de coordonnées de tous les enfants du nœud (i, j).

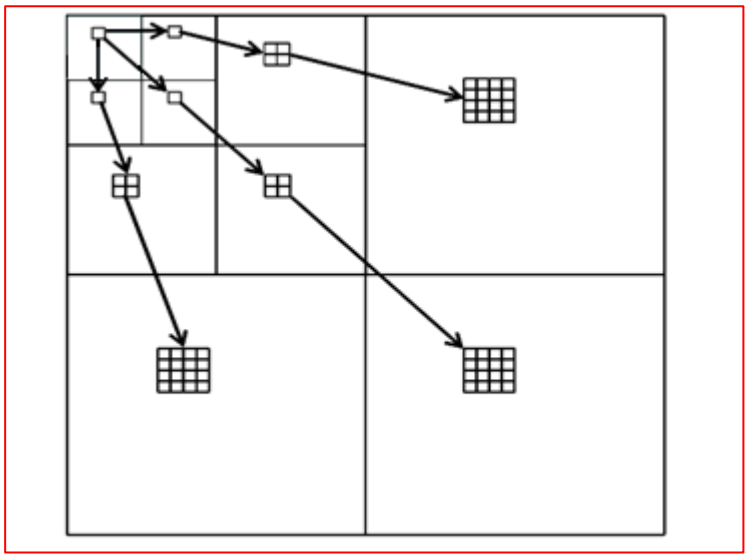


Fig.III.1 – Représentation de l'organisation en arbre des coefficients d'ondelettes.

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.

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.

2.2. Comment marche l'algorithme ?

2.2.1. Codage :

2.2.1.1. Signification des coefficients de la TO :

Avant que nous puissions exploiter le concept de relation sous-bande que nous venons de voir pour une compression efficace des coefficients ondelettes, il est nécessaire d'introduire le concept de signification. On dit qu'un coefficient ondelettes de grandeur $|X|$ est significatif par rapport à un seuil donné **Tn,** si $|X| > \mathbf{Tn}$ autrement on dit qu'il est non significatif.

Dans EZW, la signification des coefficients ondelettes sont d'abord examinés avec la plus grande valeur du seuil dans la première passe, puis progressivement, le seuil est abaissé d'un facteur 2 dans les passages suivants. Avant de commencer, tous les coefficients sont considérés comme non significatifs et progressivement, les coefficients significatifs seront détectés. A chaque passage, il y aura une carte qui indique la signification des coefficients ondelettes et cette carte a besoin d'être codée de manière efficace. La carte de signification pour chaque seuil a une entrée **"0"** si le coefficient est non significatif et une entrée **"1"** s'il est significatif. Il convient de noter que la signification est décidée qu'en ce qui concerne l'amplitude du coefficient par conséquent, le signe (positif ou négatif) doit être inclus dans le processus d'encodage.

Les coefficients sont analysés pour déterminer la signification de la manière illustrée dans la figure (III.2) ; La matrice de coefficients est parcourue par l'une des méthodes suivantes : **'Raster scan'** ou **'Morton scan'**. Ces méthodes de parcours ont été choisies de manière à préserver l'ordre d'importance des coefficients traités. Ainsi, pour les deux types, on commence par parcourir les coefficients de basse fréquence et on avance graduellement vers les détails (hautes fréquences). La différence entre elles, est la façon avec laquelle est parcourue une même sous bande.

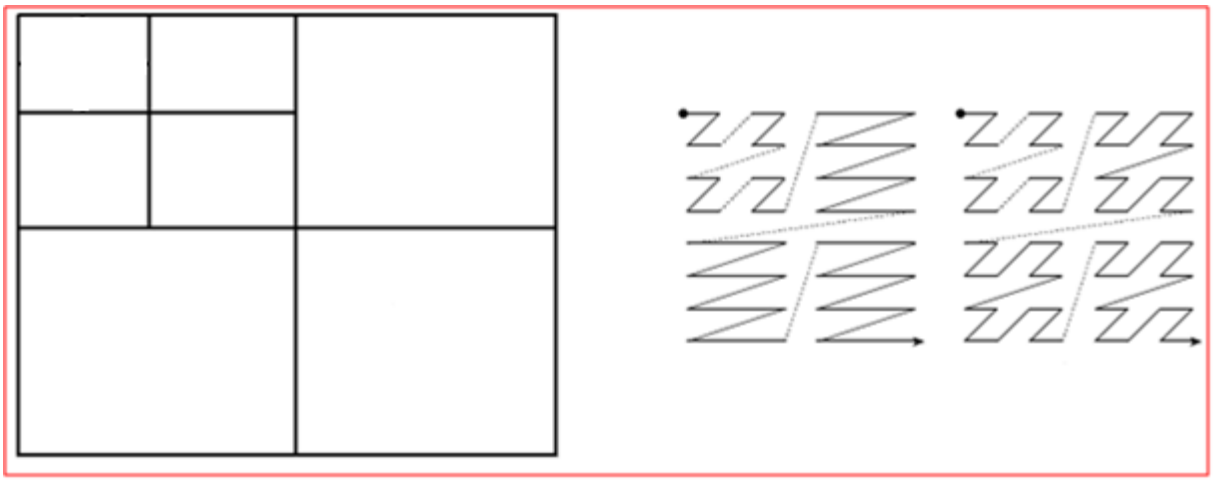


Fig.III.2 – Méthodes de parcours **'raster scan'** à gauche et **'morton scan'** à droite.

2.2.1.2. Codage de la carte de signification :

Nous allons maintenant voir comment coder efficacement pour chaque passe la carte de signification. Pour cela, la relation hiérarchique des coefficients présentés dans la figure (III.1) est utilisée. Comme vu précédemment la structure de données, appelé 'Zerotree' est définie comme coefficient non significatif, dont tous les descendants sont également négligeables. Le concept du 'Zerotree' repose sur l'hypothèse que si un coefficient de la TO à une échelle grossière est insignifiant par rapport à un seuil donné, alors tous ses descendants de fréquences plus élevées sont susceptibles d'être négligeables en regard de ce même seuil. Bien, que cela peut ne pas être toujours vrai. Il convient donc de noter que tous les coefficients non significatifs peuvent ne pas être des 'Zerotree'. Il est possible qu'un coefficient soit non significatif, mais a des descendants significatifs, ces coefficients sont appelés 'Zero isole'.

Ainsi, quatre symboles sont utilisés pour coder la carte de signification, chacun des coefficients parcourus est comparé (en valeur absolue) au seuil **Tn**, si le coefficient est supérieur au seuil il est codé **'Positif '** ou **'Négatif '**, sinon il est soit **'Zero isole'** ou bien **'Zerotree'**, on part ainsi de X symboles à coder a un dictionnaire de quatre symboles :

- Positif (P) : indique que la valeur absolue du coefficient traité est supérieure au seuil et que son signe est positif.
- Négatif (N) : indique que la valeur absolue du coefficient traité est supérieure au seuil et que son signe est négatif.

- Zéro isolé (T) : indique que la valeur absolue du coefficient traité est inférieure au seuil et qu'il existe parmi ses descendants ceux qui sont signifiants.
- Zerotree (Z) : indique que la valeur absolue du coefficient traité est insignifiante par rapport au seuil considéré ainsi que tous les coefficients qui lui succèdent dans l'arbre de descendance.

Le codage Zerotree réduit le coût du codage de la carte de signification à l'aide de similitude présente dans les différentes sous-bands. On isole ainsi des zones vastes de coefficients non significatifs, c'est ce qui permet d'atteindre de bonnes performances de compression.

2.2.1.3. Quantification par approximation successive :

La quantification par approximation successive effectue un codage des coefficients TO par étapes successives. Un seuil initial pour examiner la signification est d'abord mis en place tels que :

$$T0 = 2^{|\log_2 \max(im(x,y)|} \quad --(2)$$

Où im(x,y) désigne les coefficients d'ondelettes, x et y étant leur position dans l'espace. Le symbole 'max' correspond à la valeur maximale de tous les coefficients de la représentation en ondelettes de l'image.

Puis pour chaque étape de codage, le seuil est réduit de moitié et on examine la signification une fois de plus. La séquence des seuils qui sont appliqués pour chaque étape successive sont : T0, T1, T2,……. Tn-1 où n est le nombre de l'étape.

La description qui vient d'être faite de l'EZW fait partie de la **passe dominante,** elle est utilisée pour coder les coefficients qui n'ont pas encore été jugés significatifs par rapport à un seuil **Tn**. Cette passe dominante est suivie d'une **passe secondaire,** dans laquelle les coefficients jugés significatifs sont scannés et leurs amplitudes sont raffinées avec plus de précision. Ces coefficients subissent cette autre comparaison sur un autre seuil **Ts** qui est proportionnel au seuil du passage dominant.

$$\mathbf{Ts}= 3*2^{n-1} \quad --(3)$$

Où n est la puissance du seuil du passage dominant.

Ce passage secondaire permet au décodeur dans le cas d'une compression avec pertes d'informations, d'avoir plus de précision sur la valeur du coefficient codé. La reconstruction ne sera pas parfaite mais, elle sera de loin meilleure que si on ne code qu'avec le passage dominant. Pour cela on attribue le bit **"1"** si le coefficient est supérieur à **Ts** et le bit **"0"** si le coefficient est inférieur à **Ts**.

Le processus de codage alterne entre passe dominante et passe secondaire et le seuil est réduit de moitié après chaque passage dominant. Le codage s'arrête lorsqu'un certain taux de compression est atteint.

Le codage peut être résume dans l'organigramme suivant :

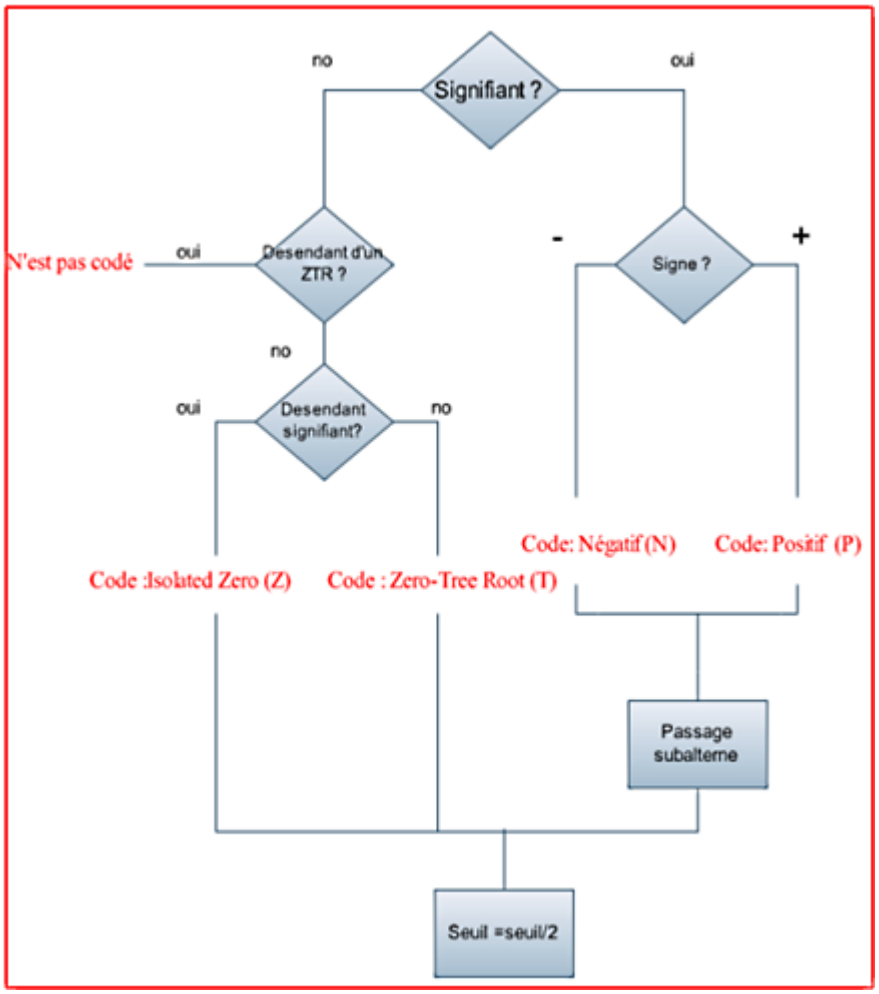


Fig.III.3 – Principe du codage EZW.

2.2.2. Décodage :

En premier lieu, le décodeur crée une matrice de dimension égale à l'image traitée à partir du nombre du niveaux de décomposition. Ensuite, comme le codeur, il calcule le premier seuil dont la puissance lui a été transmise. Le parcours de la matrice 'image' commence alors et selon les symboles lus par le décodeur un traitement est effectué :

- Si le symbole est 'Positif (P), la valeur du seuil est additionnée au contenu de la case en cours.
- Si le symbole est 'Négatif (N), Le seuil est retranché du coefficient parcouru.
- Si le symbole est 'Zerotree' (R), tout l'arbre associé à ce coefficient sera ignoré par rapport au seuil courant.
- Si le symbole est 'Zero isolé' (Z) : Alors il existe au moins un coefficient appartenant à l'arborescence du coefficient étudié qui est signifiant par rapport au seuil courant d'où, aucun coefficient ne sera ignoré dans cette arborescence.

A la fin du parcours, le seuil est divisé par 2 et l'algorithme reprend. Si le seuil atteint la valeur 1, la reconstruction sera parfaite sans aucune perte, mais au cas où l'on désire arrêter avant le décodage idéal, on peut avoir recours au traitement secondaire qui permettra plus de précision au niveau de la compression avec pertes d'informations.

2.3. 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 **[6]** :

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

Fig.III.4 – Exemple de matrice de coefficients.

Maintenant on va voir comment fonctionne le codeur :

Scan 1:

Etape 1 : Déterminer **To** = $2^{|\log_2(63)|}$ =32 **;** ce seuil est envoyé au récepteur comme un en-tête avant que tout bit de données soit envoyé.

Etape 2 : Balayage de tous les coefficients et comparer leurs valeurs avec (32).

Etape 3 : Attribuer à chaque coefficient un symbole (c.à.d. construire le passage dominant) et placer les symboles dans une liste appelée D1. Quand le symbole **T** est attribué à un coefficient alors tous ses descendants ne sont pas analysés. Le résultat est montré dans le tableau (III.1). Le tableau indique les coefficients qui sont scannés et l'ordre dans lequel ces coefficients ont été scannés. Si le symbole **T** est attribué à un coefficient alors on attribue **(X)** à tous ses descendants pour indiquer qu'ils ne seront pas analysés.

Valeurs du coefficient	Symbole
63 (0,0)	POS
-34 (1,0)	NEG
-31 (0,1)	IZ
23 (1,1)	ZTR
49 (2,0)	POS
10 (3,0)	ZTR
14 (2,1)	ZTR
-13 (3,1)	ZTR
15 (0,2)	ZTR
14 (1,2)	IZ
-9 (0,3)	ZTR
-7 (1,3)	ZTR
7 (4,0)	Z
13 (5,0)	Z
3 (4,1)	Z
4 (5,1)	Z
-1 (2,4)	Z
47 (3,4)	POS
-3 (2,5)	Z
2 (3,5)	Z

Tableau.III.1 – Résultat du premier passage dominant.

Premier passage dominant est **D1: PNZTPTTTTZTTTTTTTPTT**

Etape 4 : Maintenant, la deuxième partie de l'algorithme pour ce scan est de quantifier les coefficients ayant les symboles **P** ou **N**. **Ts** = $3*2^4$ = 48, Ce seuil secondaire permet d'avoir plus de précision sur la valeur du coefficient codé. On assignant le bit **"1"** si le coefficient est supérieur à **Ts** (l'intervalle [48,64]) et le bit **"0"** si le coefficient est inférieure à **Ts** (l'intervalle de [32,48]).

Premier passage subalterne est **S1: 1 0 1 0**

Pour les autres scans on aura :

D2: ZTNPTTTTTTTT

S2: 100110

D3:ZZZZZPPNPPNTTNNPTPTTNTTTTTTTTPTTTPTTTTTTTTTPTTTTTTTTTTTT

S3: 10011101111011011000

D4: ZZZZZZZTZTZNZZZZPTTPTPPTPNPTNTTTTTPTPNPPPPTTTTTPTPTTTPNP

S4: 1101111101100100000111011010001001010110 0

D5: ZZZZZTZZZZZTPZZZTTPTTTTNPTPPTTPTTTNPPNTTTTPNNPTTPTTPPTTT

S5: 10111100110100010111110101101100100000000110110110011000111 1

D6: ZZZTTZTTTZTTTTTNNTTT

On remarque que pour **Tn**=l, on a effectué juste le passage dominant, ceci s'explique par le fait que si on arrive à ce stade, c'est que la reconstruction est parfaite, donc on n'a pas besoin de raffinement, d'où la non utilisation du passage secondaire.

3. 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 TO 2D (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. En fait, bien que l'arbre de zéros soit spécifié par les coordonnées de la racine, la racine n'est pas incluse dans l'arbre. Le second type d'arbre (type B) est similaire mais exclut les quatre enfants de la racine. Les arbres de type B contiennent uniquement les petits-enfants, arrières petits-enfants, ... de la racine.

Une autre différence est l'ordre de traitement des coefficients qui est dépendante des données. Alors que les coefficients sont traités en zigzag dans chaque sous-bande pour l'EZW. 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 degré 2). Il est à noter que dans les deux cas, rien n'est dit sur la valeur du coefficient à la racine qui peut être significatif.

Pour caractériser les relations parent enfant dans les sous bandes. Les ensembles suivants de coordonnées sont utilisés :

- O(i, j) : Ensemble des coordonnées de tous les **enfants** du nœud (i, j).Il s'exprime de la même façon que celui de l'EZW.
- D(i, j) : Ensemble des cordonnées de tous les **descendants** du nœud (i, j) (**type A** d'arbres de zéros).
- L(i, j)=D(i, j)− O(i, j) : L'ensemble des **descendants à l'exception des enfants** (**type B** d'arbre de zéros).

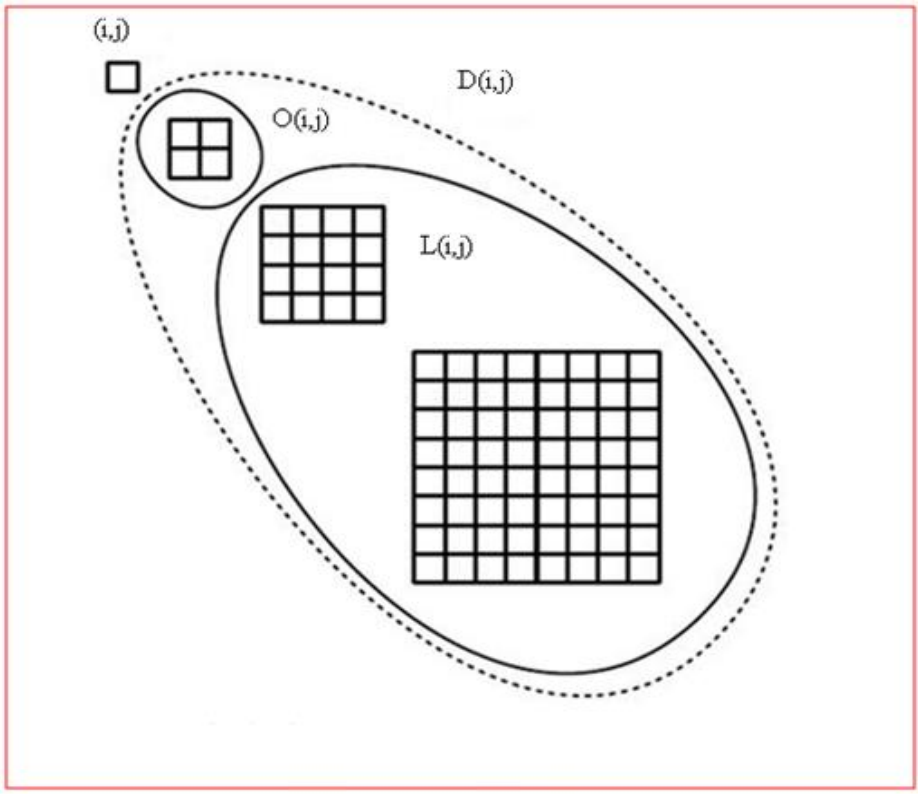


Fig.III.5 – Terminologie SPIHT pour les descendants.

3.1. Fonctionnement du SPIHT:

Pour réaliser pratiquement le codage, SPIHT stocke l'information de signification dans 3 listes ordonnées :

- La Liste des Coefficients Significatifs (**LCS**),
- La Liste des Coefficients Non significatifs (**LCN**),
- La Liste des Ensembles Non significatifs (**LEN**).

Ce sont des listes de coordonnées (i,j), dans la **LCN** et **LCS** ils représentent des coefficients, et dans la **LEN** ils représentent soit l'ensemble D(i,j) (une entrée de type A) ou l'ensemble L(i,j) (une entrée de type B).

La **LCN** contient les coordonnées de coefficients qui étaient insignifiants dans la précédente passe. Dans le passage de la passe courante, ils sont testés, et ceux qui sont significatifs sont déplacés vers la **LCS**.

Similairement, les ensembles de **LEN** sont évalués dans leur ordre d'entrée, et quand un ensemble est trouvé significatif, il est supprimé de cette liste puis est partitionné. Les nouveaux ensembles avec plus d'un élément sont ajoutés à la fin de **LEN** avec le type (A ou B), alors que les simples coefficients sont ajoutés à la fin de **LCS** ou **LCN** suivant leur signification. La liste **LCS** contient les coordonnées des coefficients qui seront visités dans la prochaine passe de raffinement. La passe de raffinement transmet le $\mathbf{n^{ième}}$ bit le plus significatif des entrées de **LCS**.

Nous allons donner maintenant l'algorithme de base du SPIHT 2D. Pour cela, nous définissons l'opérateur de signification σTn qui évalue la signification d'un sous-ensemble E pour un seuil donné Tn :

$$\sigma Tn\,(E)=\begin{cases}1 \text{ si } \exists w \in E : |w| \geq Tn \\ 0 \text{ si } \forall w \in E : |w| < Tn\end{cases} \quad --(4)$$

Le seuil Tn pour le premier plan de bit est égale à 2^n ou n = |log2(wmax)|. Tous les coefficients d'ondelettes sont scannés afin de trouver le maximum de w(i,j) après l'exécution de la transformée en ondelettes discrète. Pour le passage suivant on divise le seuil Tn par 2.

3.2. Algorithme SPIHT 2D:

1. Initialisation :

   Sortie n = |log2(wmax)| ⇔ Tn =$2^{|\log2(wmax)|}$; LCS = ∅ ; LCN = {(i, j) ∈LL}. LEN contient les mêmes coefficients que LCN excepté ceux qui n'ont pas de descendants.

2. Passe de signification :

   2.1. Pour chaque (i, j) ∈ LCN faire :

   2.1.1. Sortie σTn (i, j)

   2.1.2. si σTn (i, j)=1 alors mettre (i, j) dans LCS et coder le signe de w(i, j)

   2.2. Pour chaque (i, j) ∈ LEN faire :

   2.2.1. Si l'entrée est de **type A**

   a) Sortie σTn (D(i, j))

   b) si σTn (D(i, j)) = 1 alors :

   - Pour chaque (l,m) ∈ O(i, j) faire :
     - Sortie σTn (l,m)
     - si σTn (l,m)=1 alors mettre (l,m) dans LCS et coder le signe de w(l,m)
     - si σTn (l,m)=0 alors mettre (l,m) à la fin de LCN
   - Si L(i, j) ≠ ∅ alors mettre (i, j) à la fin de LEN comme une entrée de type B et aller à 2.2.2 sinon supprimer (i,j) de LEN

   2.2.2. Si l'entrée est de **type B**

   a) Sortie σTn (L(i, j))

   b) si σTn (L(i, j)) = 1 alors :

   - Mettre chaque (l,m) ∈ O(i, j) à la fin de LEN comme une entrée de type A
   - supprimer (i, j) de LEN.

3. Passe de raffinement :

Pour chaque coefficient (i, j) ∈ LCS à l'exception de ceux ajoutés au cours de la dernière passe. Ecrire le nième bit le plus significatif de |w(i, j)|.

4. Modification du pas de quantification : Tn ← Tn/2 et aller à l'étape 2.

Pour obtenir l'algorithme de décodage, il suffit simplement de remplacer le mot **Sortie** par **Entrée** dans l'algorithme précédent. De plus, le décodeur exécute une tâche supplémentaire en modifiant l'image reconstruite. Pour un seuil Tn donné, quand un coefficient est déplacé dans la LCS, il est évident que Tn < w(i, j) < 2Tn $=2^{n+1}$. Ainsi, le décodeur utilise cette information plus le bit de signe juste après l'insertion dans la LCS pour mettre $\widehat{w}$(i, j)= ±1.5 Tn. De manière identique, pendant la passe de raffinement le décodeur ajoute ou soustrait Tn/ 2 à $\widehat{w}$(i, j) quand on reçoit les bits de la représentation binaire de |w(i,j)|. De cette manière, la distorsion baisse à la fois pendant les 2 passes.

Enfin, on notera que contrairement à l'EZW, SPIHT produit directement des symboles binaires. Ainsi, un codeur arithmétique n'est pas obligatoire même s'il est souvent implanté pour améliorer les performances de codage. L'ensemble des améliorations proposées dans SPIHT par rapport à l'EZW, en fait la référence des méthodes de codage inter-bande (c.à.d. celles qui utilisent les redondances inter-échelles entre les sous-bandes pour coder les coefficients d'ondelettes).

3.3. Exemple :

Nous allons maintenant voir le déroulement du SPIHT sur la même matrice de décomposition en TO 2D donnée par SHAPIRO :

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

Fig.III.6 – Exemple de SHAPIRO.

Comme on l'a vu précédemment l'algorithme SPIHT maintient trois listes de coefficients : la liste des coefficients significatifs **LCS**, la liste des coefficients non significatifs **LCN** et la liste des ensembles non significatifs **LEN**. La fonction σTn (i, j) est égale à 0 si tous les descendants de (i, j) sont en dessous du seuil Tn (arbre de zéros) et 1 dans le cas contraire. SPIHT considère deux types d'arbres de zéros : le type A ou tous les descendants ne sont pas significatifs et le type B ou tous les descendants, à l'exception d'au moins un des enfants, n'est pas significatif.

Le déroulement de l'algorithme SPIHT sur les données de la figure (III.6) est décrit dans le tableau (III.2). Quelques précisions sont données pour différentes étapes :

1. Initialisation des listes : la LCS est vide, la LCN contient les 4 coefficients de plus basse fréquence et la LEN contient la même chose que la LCN à part le coefficient (0,0) (valeur 63) qui n'a pas de descendants, le seuil de départ est Tn=32.

2. (0,0) qui a la valeur 63 est significatif, il est mis dans la LCS. Le signe de cette valeur est également émis, et noté +. De même pour (1,0) (valeur-34) qui est significatif et son signe est noté -. Pour (0,1) et (1,1) ils ne sont pas significatifs donc on les laisse dans LCN.

3. et 4. Un des descendants de (1,0) est significatif (49) on teste donc les 4 enfants de (1,0) et on classe chaqu'un soit dans LCS ou dans LCN et on émit leur signes. Comme un des enfants de (1,0) est significatif, il change de type (B) et mis à la fin de la LEN. On fait de même pour (0,1).

5. Tous les descendants de (1,1) sont en dessous du seuil, on ne fait donc rien de particulier, on passe à l'élément suivant dans la LEN.

6. (1,0) est de type B, on regarde donc L(1,0) qui n'est pas significatif, on ne fait rien et on passe à l'élément suivant dans la LEN.

7. (0,1) est de type B, on regarde donc L(0,1) qui est significatif (présence de la valeur 47), on partage l'ensemble et on ajoute (0,2), (1,2), (0,3), (1,3) à la fin de la LIS comme élément de type A et on enlève (0,1) de la LIS.

8. On teste donc les 4 ensembles type A obtenus. Pour (1,2) il a un des descendants qui est significatif on teste alors les 4 enfants de (1,2) et on classe chaqu'un soit dans LCS ou dans LCN et on émit son signe. Il n'a pas de petits-enfants, alors on le retire de la LIS.

9. Au final on obtient les listes LCN, LCS et LEN et ces listes constituent les listes d'entrées pour le passage suivant ou le seuil sera de 16.

Coeff. testé	Sortie	Action	Liste	
			LIS = {(1,0)A,(0,1)A, (1,1)A} LIP= {(0,0),(1,0),(0,1),(1,1)} LSP = Ø	(1)
(0,0)	1+	(0,0) dans LSP	LIP = {(1,0),(0,1),(1,1)} LSP = {(0,0)}	(2)
(1,0)	1-	(1,0) dans LSP	LIP = {(0,1),(1,1)} LSP = {(0,0),(1,0)}	(2)
(0,1)	0	Rien		(2)
(1,1)	0	Rien		(2)
D(1,0) (2,0) (3,0) (2,1) (3,1)	1 1+ 0 0 0	Tester descendants (2,0) dans LSP (3,0) dans LIP (2,1) dans LIP (3,1) dans LIP Change de type	 LSP = {(0,0),(1,0),(2,0)} LIP = {(0,1),(1,1),(3,0)} LIP = {(0,1),(1,1),(3,0),(2,1)} LIP ={(0,1),(1,1),(3,0),(2,1),(3,1)} LIS = {(0,1)A, (1,1)A, (1,0)B}	(3) (4)
D(0,1) (0,2) (1,2) (0,3) (1,3)	1 0 0 0 0	Tester descendant (0,2) dans LIP (1,2) dans LIP (0,3) dans LIP (1,3) dans LIP Change de type	 LIP = {(0,1),(1,1),(3,0),(2,1),(3,1),(0,2)} LIP = {(0,1),(1,1),(3,0),(2,1),(3,1),(0,2),(1,2)} LIP={(0,1),(1,1),(3,0),(2,1),(3,1),(0,2),(1,2),(0,3)} LIP={(0,1),(1,1),(3,0),(2,1),(3,1),(0,2),(1,2),(0,3),(1,3)} LIS = {(1,1)A, (1,0)B, (0,1)B}	
D(1,1)	0	Rien		(5)
L(1,0)	0	Rien		(6)
L(0,1)	1	Ajouter des ensembles	LIS = {(1,1)A, (1,0)B, (0,2)A, (1,2)A, (0,3)A, (1,3)A}	(7)
D(0,2)	0	Rien		(8)
D(1,2) (2,4) (3,4) (2,5) (3,5) L(1,2)= Ø	1 0 1+ 0 0	tester descendant (2,4) dans LIP (3,4) dans LSP (2,5) dans LIP (3,5) dans LIP enlever (1,2)	 LIP={(0,1),(1,1),(3,0),(2,1),(3,1),(0,2),(1,2),(0,3),(1,3),(2,4)} LSP = {(0,0),(1,0),(2,0),(3,4)} LIP={(0,1),(1,1),(3,0),(2,1),(3,1),(0,2),(1,2),(0,3),(1,3),(2,4),(2,5)} LIP={(0,1),(1,1),(3,0),(2,1),(3,1),(0,2),(1,2),(0,3),(1,3),(2,4),(2,5),(3,5)} LIS = {(1,1)A, (1,0)B, (0,2)A, (0,3)A, (1,3)A}	(8)
D(0,3) D(1,3)	0 0	Rien Rien		(8)
			LIS = {(1,1)A, (1,0)B, (0,2)A, (0,3)A, (1,3)A} LIP = {(0,1),(1,1),(3,0),(2,1),(3,1),(0,2),(1,2),(0,3),(1,3),(2,4),(2,5),(3,5)} LSP = {(0,0),(1,0),(2,0),(3,4)}	(9)

Tableau.III.2 Tableau montrant étape par étape le déroulement du premier passage de l'algorithme SPIHT 2D.

4. Extension aux images 3D du SPIHT :

L'implémentation de la version 2D du SPIHT qui vient d'être présenté, peut être appliquée séparément sur chacun des différents canaux de l'image 3D, après que ces dernières aient subi une transformation en ondelettes 2D. Dans notre étude nous essayons de tirer partie de la nature 3D des images à compresser, il est donc indispensable d'étendre le SPIHT à trois dimensions, afin qu'il puisse s'adapter à la TO 3D développée précédemment et donc d'exploiter la redondance qu'il y a dans cette troisième dimension.

L'extension aux images 3D volumiques d'EZW et du SPIHT s'appuie sur des relations parents/enfants dans trois dimensions au lieu de deux **[2] [13].** Ceci est lié à l'utilisation de la TO 3D dyadique présentée précédemment.

Dans le cas 3D le nœud racine de l'arbre (correspondant aux coefficients de la sous-bande LLLd pour une TO 3D à d niveaux de décomposition) a seulement 7 enfants, alors que tous les autres nœuds à l'exception des extrémités en possèdent 8. En d'autres termes, à l'exception du nœud racine et des extrémités de l'arbre le lien parent enfant 3D est le suivant :

$$O(i,j,k) = \left\{ \begin{matrix} (2i,2j,2k),(2i,2j+1,2k),(2i+1,2j,2k), \\ (2i+1,2j+1,2k),(2i,2j,2k+1),(2i+1,2j,2k+1), \\ (2i,2j+1,2k+1),(2i+1,2j+1,2k+1) \end{matrix} \right\} --(5)$$

Où O (i, j, k) représente un ensemble de coordonnées de tous les enfants du nœud (i, j, k).Ceci est illustré par la figure (III.7) :

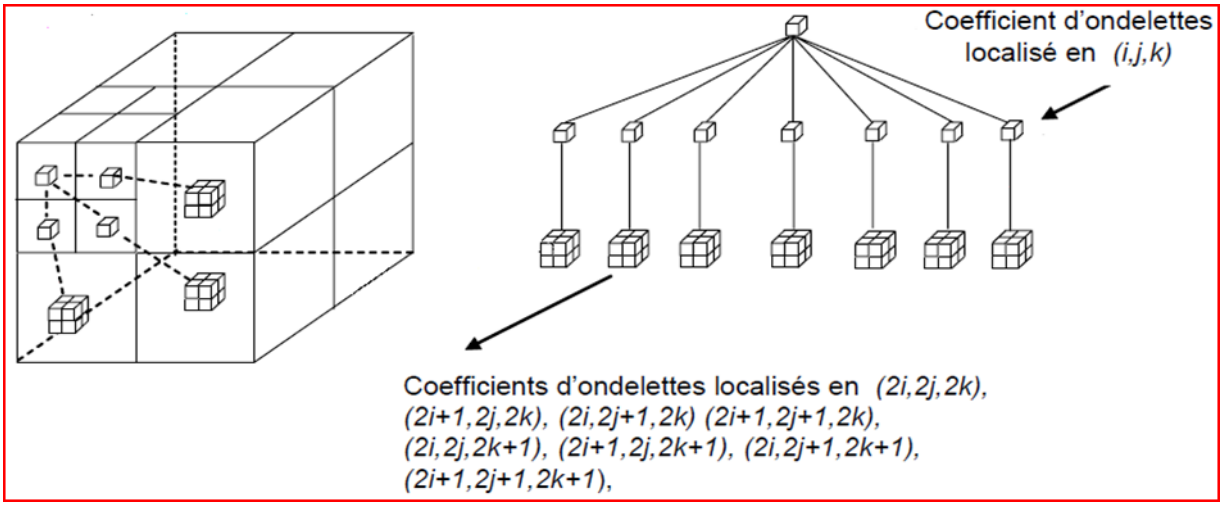


Fig.III.7 – Le nœud racine de l'arbre 3D

L'algorithme SPIHT 3D utilise aussi les ensembles suivants de coordonnées (étendu à 3D) dans la méthode de codage qui sera présentée par la suite :

- O (i, j, k) : Ensemble des coordonnées de tous les enfants du nœud (i, j, k).
- D (i, j, k) : Ensemble des cordonnées de tous les descendants du nœud (i, j, k) (type A d'arbres de zéros)
- L (i, j, k) = D (i, j, k) − O (i, j, k) Ensemble des descendants à l'exception des enfants (type B d'arbre de zéros).

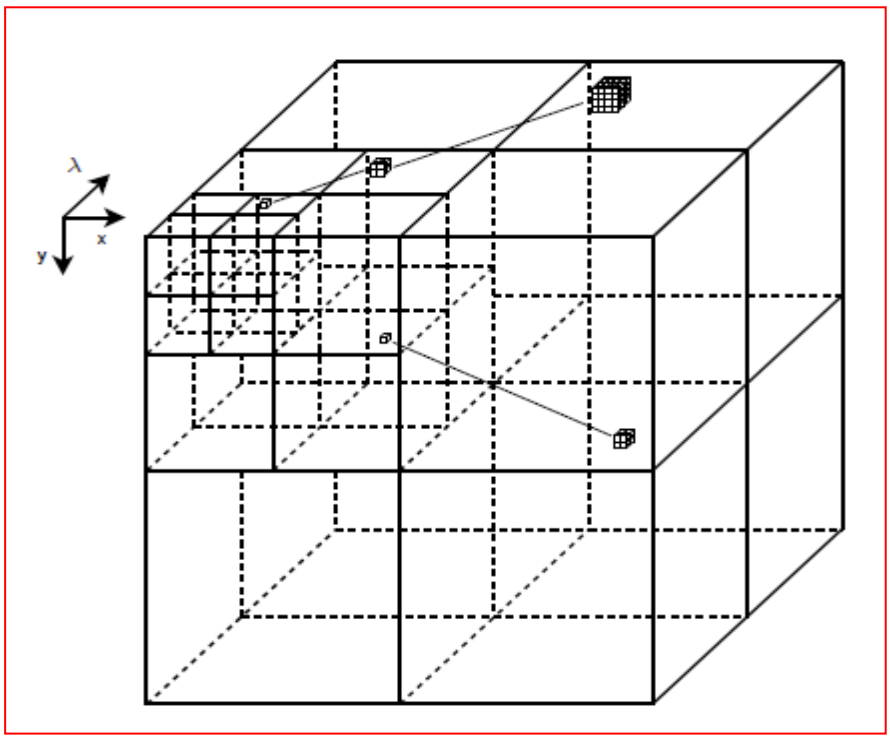


Fig.III.8 – Les relations parents-enfants SPIHT 3D.

L'implémentation du SPIHT 3D est très similaire au SPIHT 2D, le tri des pixels est le même que dans le SPIHT 2D, sauf que les arbres de zéros sont à trois dimensions. La passe principale et de raffinement sont identiques et le résultat est un flux de bits. Avant de présenter l'algorithme écrit sous MATLAB, nous allons d'abord décrire les grandes étapes afin de faciliter la compréhension de l'algorithme du SPIHT 3D.

Etape 1 : Calcul du seuil de départ **Tn**, donnée par : $Tn = 2^{\lfloor \log_2 \max(im(x,y,\lambda)) \rfloor}$.

Etape 2 : Définition des listes de coordonnées (x,y,λ) de **LCN**, **LCS** et **LEN.**

Etape 3 : Comparer la valeur des pixels de la **LCN** au seuil **Tn** :

Si c'est positif alors il sera codé '11'.

Si c'est négatif alors il sera codé '10'.

Sinon, il sera code '0'.

Etape 4 : Pour chaque ensemble de **LEN** comparer au seuil **Tn** le descendant ayant la plus grande valeur absolue.

Si type A : S'il y a des descendants supérieur à **Tn** sortie '1' et tester les huit enfants du nœud (x,y, λ) ; pour chacun d'eux :

S'il est positif, il sera codé '11'.

S'il est négatif, il sera codé '10'.

Sinon il sera code '0'.

Si les enfants ont des descendants, mettre le nœud à la fin de la **LEN** en tant que **type B.**

Si type B : S'il y a des ceoficients qui sont supérieurs à **Tn** sortie '1' et mettre chaqu'un des huit enfants du nœud (x,y, λ) comme entrée **type A.**

Si non signifiant, sortie 0.

Etape 5 : Passe de raffinement.

Etape 6 : Décrémenter n et retourner à l'étape 3.

Dans ce qui suit nous allons présenter et commenter l'algorithme SPIHT 3D écrit sous MATLAB, ce code est une adaptation au cas 3D de l'algorithme SPIHT 2D écrit par **JING TIAN**.

5. L'algorithme SPIHT 3D sous Matlab:

```
function out = func_SPIHT_Enc(M, max_bits, level)
%
%    input: M : Donnée 3D dans le domaine ondelettes
%           max_bits: Nombre de bits max
%           level: niveaux de décomposition
%
%    output: out: flux de bits
%
%---------- Initialization -----------------
bitctr = 0;
out = 2*ones(1,max_bits);
n_max = floor(log2(abs(max(max(max(M))))));
Bits_Header = 0;
Bits_LSP = 0;
Bits_LIP = 0;
Bits_LIS = 0;
```

Initialisation des conteurs qui seront utilisés dans le code et calcul du seuil de départ **Tn**, donnée par cette équation : $Tn = 2^{\lfloor \log_2 \max(im(x,y,\lambda)) \rfloor}$.

Où : $n\_max = \lfloor \log_2 \max(im(x,y,\lambda)) \rfloor$.

```
%---------- output bit stream header ----------------
out(1,[1 2 3 4 5]) = [size(M,1) n_max level size(M,2) size(M,3) ];
bitctr = bitctr + 40;
index = 6;
Bits_Header = Bits_Header + 40;
```

Définition de l'entête nécessaire pour permettre le décodage. (Taille image, niveaux de décomposition, max bits).

Prendre en compte la taille de l'entête dans les conteurs.

```
%---------- Initialize LIP, LSP, LIS ---------------
temp = [];
bandsize1 = round (2.^(log2(size(M, 1)) - level + 1));
bandsize2 = round (2.^(log2(size(M, 2)) - level + 1));
bandsize3 = round (2.^(log2(size(M, 3)) - level + 1));

temp1 = 1 : bandsize1;
for i = 1 : bandsize2
     temp = [temp; temp1];
end
LIP(:, 1) = temp(:);
temp = [];

temp1 = 1 : bandsize2;
for i = 1 : bandsize1
     temp = [temp; temp1];
end
temp = temp';
LIP(:, 2) = temp(:);
```

```
len = size(LIP,1);
LIP(:,3) = ones(len,1);
LIS = LIP;

pstart = 1;
pend = bandsize2 / 2;
for i = 1 : bandsize1 / 2
LIS(pstart : pend, :) = [];
     pdel = pend - pstart + 1;
     pstart = pstart + bandsize2 - pdel;
     pend = pend + bandsize2 - pdel;
end

len2 = size(LIS,1);
tLIP=LIP;
tLIS=LIS;
for i=1:bandsize3-1
     LIP=[LIP;tLIP];
     LIP(i*len+1:(i+1)*len,3) = (i+1)*ones(len,1);
End

for i=1:bandsize3/2-1
     LIS=[LIS;tLIS];
     LIS(i*len2+1:(i+1)*len2,3) = (i+1)*ones(len2,1);
End

lenT = size(LIS,1);
for i=bandsize3/2:bandsize3-1
     LIS=[LIS;tLIP];
     LIS(lenT+1:lenT+len,3) = (i+1)*ones(len,1);
     lenT=lenT+len;
end

LIS(:, 4) = zeros((bandsize3/2*len2 + bandsize3/2*len), 1);
LSP = [];
n = n_max;
```

Définition des listes de coordonnées (i,j,λ) de **LCN**, **LCS** et **LEN.**

```
%----------- CODING --------------------------------
while(bitctr < max_bits)

      % Sorting Pass
      LIPtemp = LIP;temp = 0;
      for i = 1:size(LIPtemp,1)
          temp = temp+1;
          if (bitctr + 1) >= max_bits
             if (bitctr < max_bits)
                out(length(out))=[];
             end
             return
          end
          if abs(M(LIPtemp(i,1),LIPtemp(i,2),LIPtemp(i,3))) >= 2^n
             out(index) = 1; bitctr = bitctr + 1;
             index = index +1; Bits_LIP = Bits_LIP + 1;
             sgn = M(LIPtemp(i,1),LIPtemp(i,2),LIPtemp(i,3))>=0;
             out(index) = sgn; bitctr = bitctr + 1;
```

```
            index = index +1; Bits_LIP = Bits_LIP + 1;
            LSP = [LSP; LIPtemp(i,:)];
            LIP(temp,:) = []; temp = temp - 1;
        else
            out(index) = 0; bitctr = bitctr + 1;
            index = index +1;
            Bits_LIP = Bits_LIP + 1;

        end
    end
```

Vérifiez la signification de tous les coefficients de LCN

Signifiant : sortie 1, sortie un bit de signe, et déplacer le coefficient affecté à la LCS.

Non Signifiant : sortie 0.

```
    LIStemp = LIS; temp = 0; i = 1;
    while ( i <= size(LIStemp,1))
    temp = temp + 1;
        if LIStemp(i,4) == 0
            if bitctr >= max_bits
                return
            end
            max_d = func_Descendant(LIStemp(i,1),LIStemp(i,2),
                                    LIStemp(i,3,LIStemp(i,4),M);
```

Fonction qui permet de trouver le descendant ayant la plus grande valeur absolue du pixel (i,j,λ). (Voir code de la fonction).

```
            if  max_d >= 2^n
                out(index) = 1; bitctr = bitctr + 1;
                index = index +1; Bits_LIS = Bits_LIS + 1;
                x = LIStemp(i,1); y = LIStemp(i,2); z = LIStemp(i,3);

                if (bitctr + 1) >= max_bits
                    if (bitctr < max_bits)
                        out(length(out))=[];
                    end
                    return
                end

                if abs(M(2*x-1,2*y-1,2*z-1)) >= 2^n
                    LSP = [LSP; 2*x-1 2*y-1 2*z-1];
                    out(index) = 1; bitctr = bitctr + 1;
                    index = index +1; Bits_LIS = Bits_LIS + 1;
                    sgn = M(2*x-1,2*y-1,2*z-1)>=0;
                    out(index) = sgn; bitctr = bitctr + 1;
                    index = index +1; Bits_LIS = Bits_LIS + 1;
                else
                    out(index) = 0; bitctr = bitctr + 1;
                    index = index +1; Bits_LIS = Bits_LIS + 1;
                    LIP = [LIP; 2*x-1 2*y-1 2*z-1];
                end
                if (bitctr + 1) >= max_bits
```

```
    if (bitctr < max_bits)
       out(length(out))=[];
    end
    return
end
if abs(M(2*x-1,2*y,2*z-1)) >= 2^n
    LSP = [LSP; 2*x-1 2*y 2*z-1];
    out(index) = 1; bitctr = bitctr + 1;
    index = index +1; Bits_LIS = Bits_LIS + 1;
    sgn = M(2*x-1,2*y,2*z-1)>=0;
    out(index) = sgn; bitctr = bitctr + 1;
    index = index +1; Bits_LIS = Bits_LIS + 1;
else
    out(index) = 0; bitctr = bitctr + 1;
    index = index +1; Bits_LIS = Bits_LIS + 1;
    LIP = [LIP; 2*x-1 2*y 2*z-1];
End



if (bitctr + 1) >= max_bits
   if (bitctr < max_bits)
       out(length(out))=[];

   end
   return
end
if abs(M(2*x,2*y-1,2*z-1)) >= 2^n
    LSP = [LSP; 2*x 2*y-1 2*z-1];
    out(index) = 1; bitctr = bitctr + 1;
    index = index +1; Bits_LIS = Bits_LIS + 1;
    sgn = M(2*x,2*y-1,2*z-1)>=0;
    out(index) = sgn; bitctr = bitctr + 1;
    index = index +1; Bits_LIS = Bits_LIS + 1;
else
    out(index) = 0; bitctr = bitctr + 1;
    index = index +1; Bits_LIS = Bits_LIS + 1;
    LIP = [LIP; 2*x 2*y-1 2*z-1];
end
if (bitctr + 1) >= max_bits
    if (bitctr < max_bits)
        out(length(out))=[];
    end
    return
end
if abs(M(2*x,2*y,2*z-1)) >= 2^n
    LSP = [LSP; 2*x 2*y 2*z-1];
    out(index) = 1; bitctr = bitctr + 1;
    index = index +1; Bits_LIS = Bits_LIS + 1;
    sgn = M(2*x,2*y,2*z-1)>=0;
    out(index) = sgn; bitctr = bitctr + 1;
    index = index +1; Bits_LIS = Bits_LIS + 1;
else
    out(index) = 0; bitctr = bitctr + 1;
    index = index +1; Bits_LIS = Bits_LIS + 1;
    LIP = [LIP; 2*x 2*y 2*z-1];
end
if (bitctr + 1) >= max_bits
    if (bitctr < max_bits)
        out(length(out))=[];
    end
    return
```

```
end
if abs(M(2*x-1,2*y-1,2*z)) >= 2^n
   LSP = [LSP; 2*x-1 2*y-1 2*z];
   out(index) = 1; bitctr = bitctr + 1;
   index = index +1; Bits_LIS = Bits_LIS + 1;
   sgn = M(2*x-1,2*y-1,2*z)>=0;
   out(index) = sgn; bitctr = bitctr + 1;
   index = index +1; Bits_LIS = Bits_LIS + 1;
else
    out(index) = 0; bitctr = bitctr + 1;
    index = index +1; Bits_LIS = Bits_LIS + 1;
    LIP = [LIP; 2*x-1 2*y-1 2*z];
end
if (bitctr + 1) >= max_bits
   if (bitctr < max_bits)
   out(length(out))=[];

    end
    return
end
if abs(M(2*x-1,2*y,2*z)) >= 2^n
    LSP = [LSP; 2*x-1 2*y 2*z];
    out(index) = 1; bitctr = bitctr + 1;
    index = index +1; Bits_LIS = Bits_LIS + 1;
    sgn = M(2*x-1,2*y,2*z)>=0;
    out(index) = sgn; bitctr = bitctr + 1;
    index = index +1; Bits_LIS = Bits_LIS + 1;
else
    out(index) = 0; bitctr = bitctr + 1;
    index = index +1; Bits_LIS = Bits_LIS + 1;
    LIP = [LIP; 2*x-1 2*y 2*z];
end
if (bitctr + 1) >= max_bits
   if (bitctr < max_bits)
   out(length(out))=[];
   end
   return
end
if abs(M(2*x,2*y-1,2*z)) >= 2^n
   LSP = [LSP; 2*x 2*y-1 2*z];
out(index) = 1; bitctr = bitctr + 1;
   index = index +1; Bits_LIS = Bits_LIS + 1;
   sgn = M(2*x,2*y-1,2*z)>=0;
   out(index) = sgn; bitctr = bitctr + 1;
   index = index +1; Bits_LIS = Bits_LIS + 1;
else
   out(index) = 0; bitctr = bitctr + 1;
   index = index +1; Bits_LIS = Bits_LIS + 1;
   LIP = [LIP; 2*x 2*y-1 2*z];
end
if (bitctr + 1) >= max_bits
   if (bitctr < max_bits)
   out(length(out))=[];
   end
   return
end
if abs(M(2*x,2*y,2*z)) >= 2^n
   LSP = [LSP; 2*x 2*y 2*z];
   out(index) = 1; bitctr = bitctr + 1;
   index = index +1; Bits_LIS = Bits_LIS + 1;
   sgn = M(2*x,2*y,2*z)>=0;
   out(index) = sgn; bitctr = bitctr + 1;
```

```
                index = index +1; Bits_LIS = Bits_LIS + 1;
            else
                out(index) = 0; bitctr = bitctr + 1;
                index = index +1; Bits_LIS = Bits_LIS + 1;
                LIP = [LIP; 2*x 2*y 2*z];
            End
```

Si type A :

Si significatif, sortie 1 et coder ses enfants (8 enfants):

* Si un enfant est significatif, sortie 1, puis un bit de signe et l'ajouter à la LCS.

*Si un enfant est non signifiant, sortie 0 et ajouter l'enfant à la fin de LCN.

Si non signifiant, sortie 0.

```
            if ((2*(2*x)-1) < size(M,1) & (2*(2*y)-1) < size(M,2) &
                                    (2*(2*z)-1) < size(M,3))
               LIS = [LIS; LIStemp(i,1) LIStemp(i,2) LIStemp(i,3) 1];
               LIStemp = [LIStemp; LIStemp(i,1) LIStemp(i,2)
                                                  LIStemp(i,3) 1];

            end
                LIS(temp,:) = []; temp = temp-1;
```

Si les enfants ont des descendants, déplacer (i,j,λ) à la fin de la LEN en tant que type B et le retirer de LEN.

```
        else
                out(index) = 0; bitctr = bitctr + 1;
                index = index +1; Bits_LIS = Bits_LIS + 1;
        end
    else
        if bitctr >= max_bits
           return
        end
    max_d = func_Descendant(LIStemp(i,1),LIStemp(i,2),LIStemp(i,3),
                                        LIStemp(i,4),M);
        if max_d >= 2^n
            out(index) = 1; bitctr = bitctr + 1;
            index = index +1;
            x = LIStemp(i,1); y = LIStemp(i,2); z = LIStemp(i,3);

            LIS = [LIS; 2*x-1 2*y-1 2*z-1 0; 2*x-1 2*y 2*z-1 0;
                 2*x 2*y-1 2*z-1 0; 2*x 2*y 2*z-1 0;2*x-1 2*y-1 2*z 0;
                      2*x-1 2*y 2*z 0; 2*x 2*y-1 2*z 0; 2*x 2*y 2*z 0];

            LIStemp = [LIStemp; 2*x-1 2*y-1 2*z-1 0; 2*x-1 2*y 2*z-1 0;
```

```
                    2*x 2*y-1 2*z-1 0; 2*x 2*y 2*z-1 0;2*x-1 2*y-1 2*z 0;
                        2*x-1 2*y 2*z 0; 2*x 2*y-1 2*z 0; 2*x 2*y 2*z 0];

                LIS(temp,:) = []; temp = temp - 1;

            else
                 out(index) = 0; bitctr = bitctr + 1;
                 index = index +1; Bits_LIS = Bits_LIS + 1;
            end
```

Si type B:

Si signifiant, sortie 1, ajouter chacun des descendant (8 enfants) à la fin de LEN comme entrée de type A et enlever le nœud parent (i,j,λ) de la LEN.

Si non signifiant, sortie 0

```
          end
              i = i+1;
      end

      % Refinement Pass

      temp = 1;
      value = floor(abs(2^(n_max-n+1)*M(LSP(temp,1),LSP(temp,2),
                                                    LSP(temp,3))));
      while (value >= 2^(n_max+2) & (temp <= size(LSP,1)))
            if bitctr >= max_bits
            return
            end
            s = bitget(value,n_max+2);
            out(index) = s; bitctr = bitctr + 1;
            index = index +1; Bits_LSP = Bits_LSP + 1;
            temp = temp + 1;
            if temp <= size(LSP,1)
               value = floor(abs(2^(n_max-n+1)*M(LSP(temp,1),LSP(temp,2),
                                                    LSP(temp,3))));
            end
      end
      n = n - 1;
end
```

Exécuter la passe de raffinement

Décrémenter n et retourne à la passe de signification.

Pour finir nous allons voir maintenant la fonction associée au code qui vient d'être présenté et qui permet de trouver le descendant ayant la plus grande valeur absolue du pixel (i,j,λ) .

```
function value = func_Descendant(i, j, k, type, M)
%
% trouvez le descendant ayant la plus grande valeur  absolue de la racine
% (i,j,λ)  .
%
% input:  i : lignes.
%         j : colonnes.
%         k : fréquences.
%         type : type de  descendant ( A ou B ).
%         M : Donnée  3D .
%
% output: value : descendant ayant la plus grande valeur.

[s1,s2,s3] = size(M);
S = [];
index = 0; a = 0; b = 0; c = 0;
while ((2*i-1)<s1 & (2*j-1)<s2 & (2*k-1)<s3)
      a = i-1; b = j-1; c = k-1;

      mind = [2*(a+1)-1:2*(a+2^index)];
      nind = [2*(b+1)-1:2*(b+2^index)];
      zind = [2*(c+1)-1:2*(c+2^index)];

      chk = mind <= s1;
      len = sum(chk);
      if len < length(mind)
          mind(len+1:length(mind)) = [];
      end

      chk = nind <= s2;
      len = sum(chk);
      if len < length(nind)
         nind(len+1:length(nind)) = [];
      end

      chk = zind <= s3;
      len = sum(chk);
      if len < length(zind)
         zind(len+1:length(zind)) = [];
      end

      S = [S reshape(M(mind,nind,zind),1,[])];
      index = index + 1;
      i = 2*a+1; j = 2*b+1; k = 2*c+1;
end
if type == 1
S(:,1:4) = [];;
end

value = max(abs(S));
```

Tests & Résultats

CHAPITRE 4

- Préambule.
- Présentation de l'interface graphique.
- Critères d'évaluation de la compression.
- Tests et résultats de la compression.
- Discussion.

1. Préambule :

Ce chapitre présente l'expérimentation et les résultats des codes introduits aux chapitres précèdent. Rappelons que notre travail consiste à développer un algorithme de compression d'images hyperspectrales utilisant une transformée en Ondelettes 3D (TOD 3D) et une quantification par arbre de zéros (SPIHT 3D) afin d'atteindre de bons taux de compression pour ce type d'images sans trop altérer la qualité. Nous évaluerons ici les résultats obtenus pour différents paramètres de précision et nous verrons l'influence de ces paramètres sur la qualité de compression obtenue.

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. La notion de critère qualité en adéquation avec les applications en aval, est donc particulièrement intéressante pour évaluer l'impact d'un procède de compression sur l'exploitation de l'image finale. Pour ce faire, nous allons définir certains critères tels que le PSNR, ainsi que le MSE ou le SNR, pour permettre l'évaluation de la qualité de compression de l'algorithme élaborée pour les images hyperspectrales.

Afin de faciliter l'utilisation des algorithmes élabores pour la compression des images hyperspectrales et de manipuler les différents paramètres nous avons regroupé ces algorithmes dans un GUI (Graphical User Interface) sous MATLAB. Cette interface graphique permettra de sélectionner l'image 3D ou chaque canal de l'image 3D séparément à l'aide d'un explorateur et d'appliquer soit la chaine de compression 2D ou 3D étant donné que les deux chaines de compression ont été implémentées afin de voir l'amélioration apportée par la version 3D de l'algorithme.

Cette interface permettra de tester les performances de compression de plusieurs filtres pour différents niveaux de décomposition ; elle permettra aussi de calculer le PSNR et d'afficher l'image compressée et non compressée afin de faire une comparaison visuelle des résultats. Nous allons présenter dans ce qui suit cette interface plus en détail et décrire son utilisation.

2. Présentation de l'interface graphique :

Le GUI (Graphical User Interface) de MATLAB permet de créer des interfaces où le programmeur choisit plusieurs types d'objets (boutons, edit box, listbox.....) appelés handles. Ensuite, il réalise la programmation pour obtenir l'interaction qu'il souhaite entre ces différents objets.

L'interface réalisée permet la compression des images hyperspctrales à l'aide d'un large choix de filtres ondelettes et pour différents niveaux de décomposition. La compression peut être faite sur chaque canal séparément ou sur le cube hyperspctral 3D en entier. Une capture de l'interface est donnée dans la figure suivante.

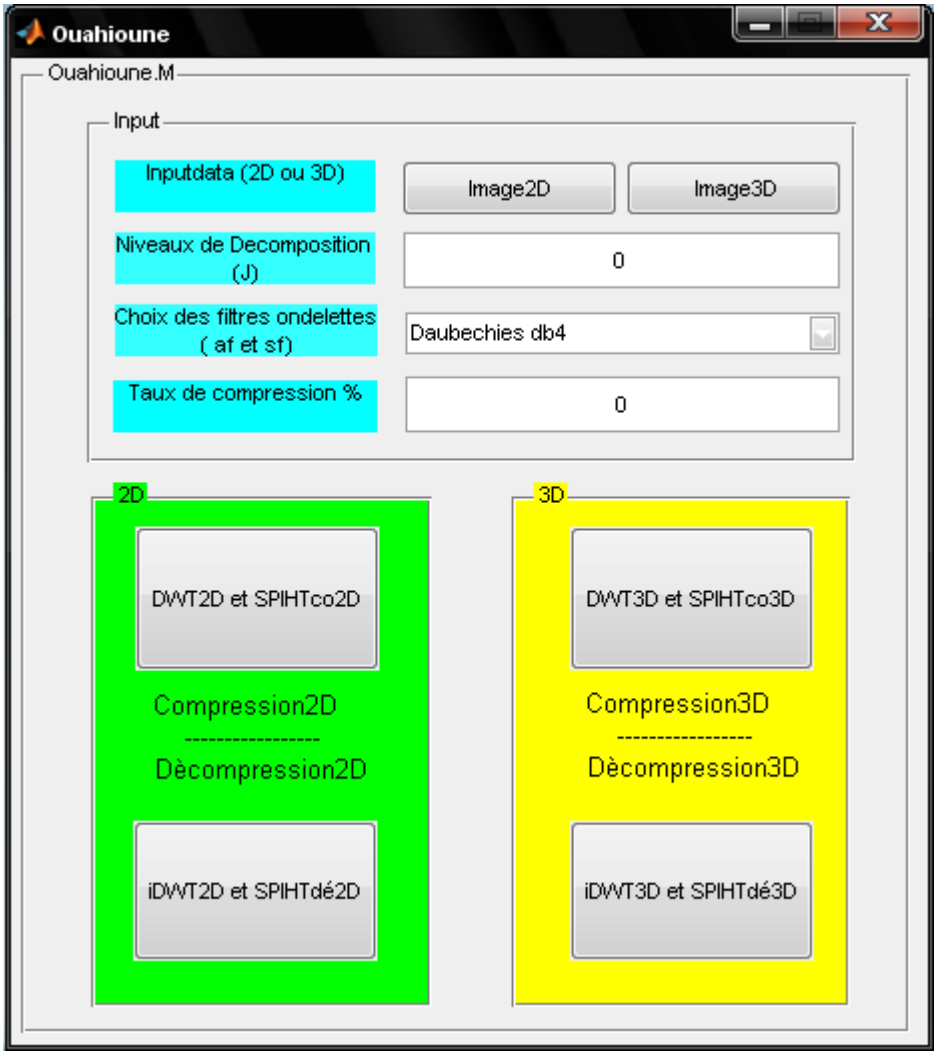


Fig. IV.1 – L'interface graphique de compression des images hyperspectrales.

2.1 Cette interface contient :

– Deux **‘Pushbutton’** pour sélectionner une image 2D ou 3D.

– Deux **‘Edit boxes’** pour définir le niveau de décomposition (J) et le rapport de compression (RC).

– Un **‘Pop-up Menu’** dans lequel se trouvent les différents filtres ondelettes (filtres d’analyses (af) et filtres de synthèses (sf)).

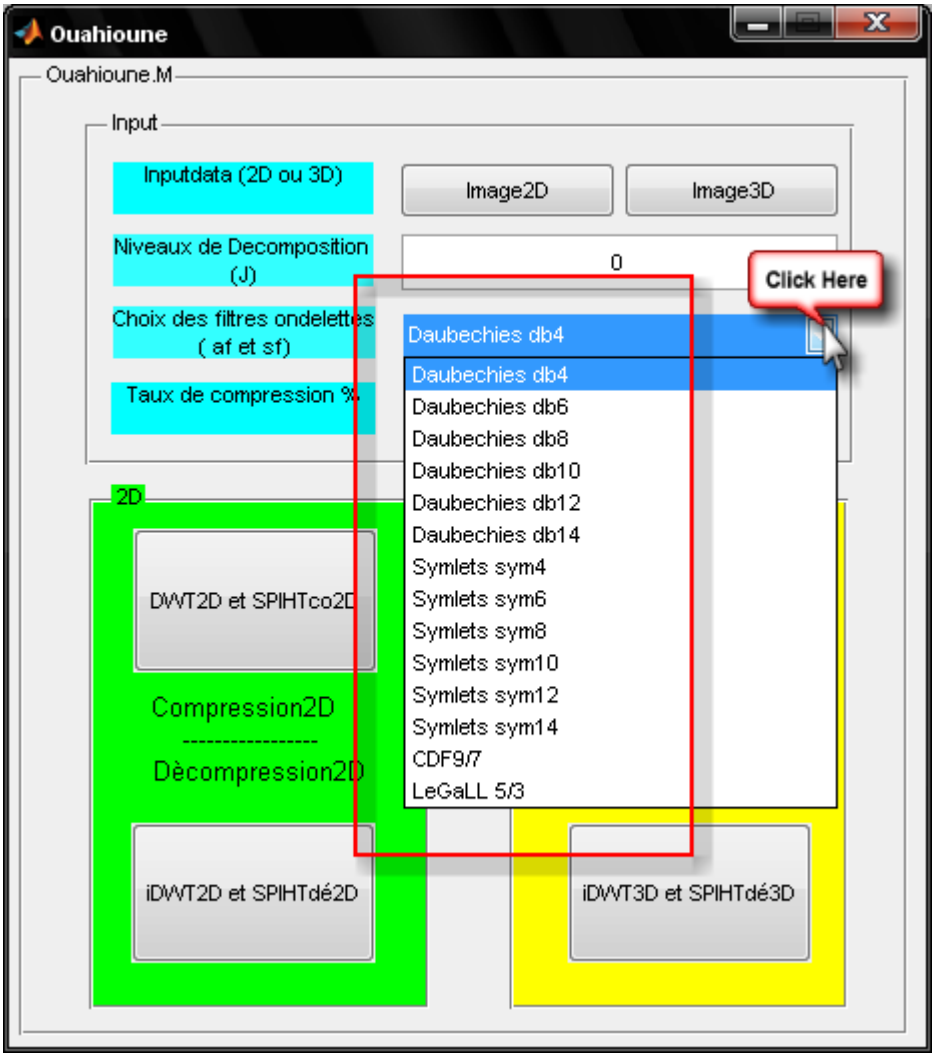


Fig. IV.2 – Les différents filtres ondelettes disponibles dans le GUI.

– Deux autres panels (un 2D (en vert) et l’autre 3D (en jaune)) regroupant chacun deux **‘Pushbutton’** permettent d’exécuter la compression et la décompression pour les deux cas 2D et 3D.

– Un dernier panel ‘Output’ nous permet d’afficher l’image avant et après compression et nous permet de calculer le PSNR pour les deux cas 2D et 3D (voir Fig. IV.5).

2.2 Démarche à suivre pour la compression des images hyperspectrales :

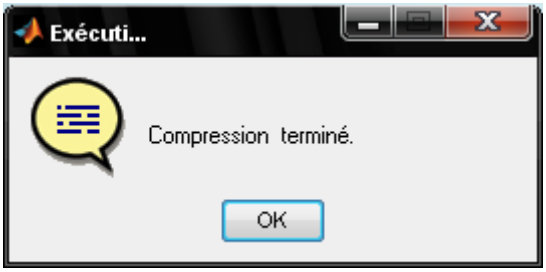
1. Pour sélectionner une image, il faut cliquer sur le bouton sélectionner une image 3D, qui ouvre une fenêtre permettant de parcourir le disque et de choisir n'importe quelle image hyperspectrale.

Une fenêtre apparait pour nous informer que l'image a été chargée et le nombre de canaux de l'image 3D chargés (par défaut nous avons choisi de charger 32 canaux).

2. Puis, par action d'un clic sur 'Pop-up Menu', permet de dérouler le menu contenant les différents filtres ondelettes et de choisir un par un clic dessus.
3. Ensuite à l'aide du clavier numérique nous saisissons le nombre de décomposition (J) et le rapport de compression (RC) désiré.
4. Une fois que tous ces paramètres sont définis il ne reste plus qu'à compresser l'image 3D par un clic sur le Botton **'DWT3D et SPIHTco3D'**; la compression 3D s'exécute.

Fig. IV.3 – Compression 3D par clic sur le Botton **' DWT3D et SPIHTco3D'**.

Une fois la compression 3D terminée la fenêtre suivante apparait :



5. En suite un clic sur le Botton **‘ iDWT3D et SPIHTdé3D’** permet de décompresser l’image 3D. Lorsque l’opération se termine, une fenêtre avec le message ‘décompression terminée’ apparait.

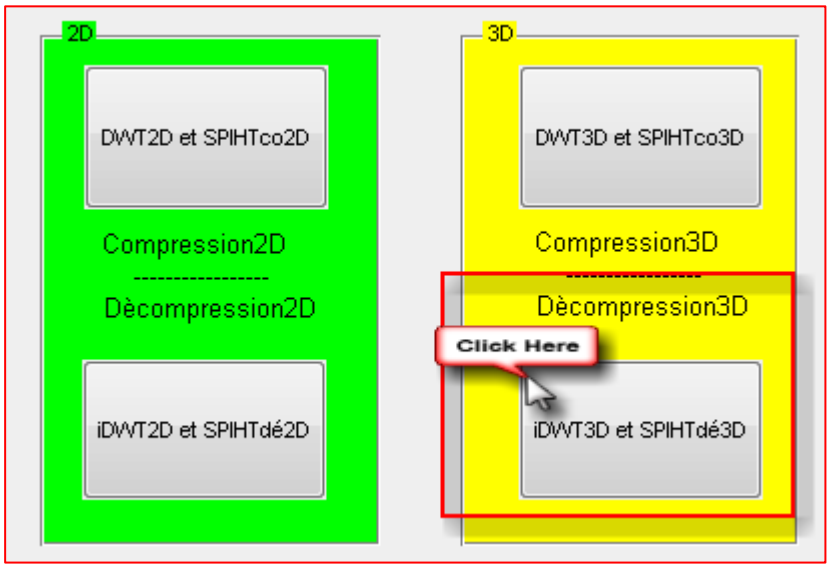
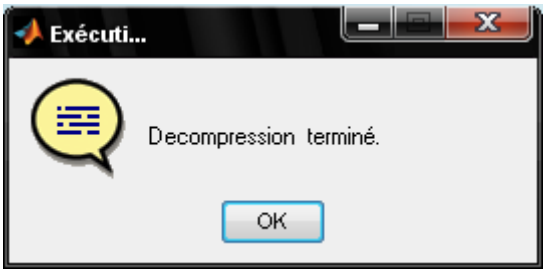


Fig. IV.4 – Décompression 3D par clic sur le Botton **‘ iDWT3D et SPIHTdé3D’**.

Une fois la décompression 3D terminée on aura la fenêtre suivante :



L'ajout du panel **'output'** dans notre interface graphique (GUI de MATLAB) nous permet de récupérer les deux images (pour les deux cas 2D et 3D) l'une avant compression (originale) et l'autre après compression (compressée) et de les visualiser dans une fenêtre sur notre écran.

Fig. IV.5 – Panel **'output'** dans l'interface graphique permet d'afficher l'image compressée et de calculer le PSNR.

Deux **'Pushbutton'** (PSNR 2D et PSNR 3D) nous permettent de calculer le PSNR de l'image compressée. La figure suivante nous donne un exemple des résultats obtenus avec notre algorithme.

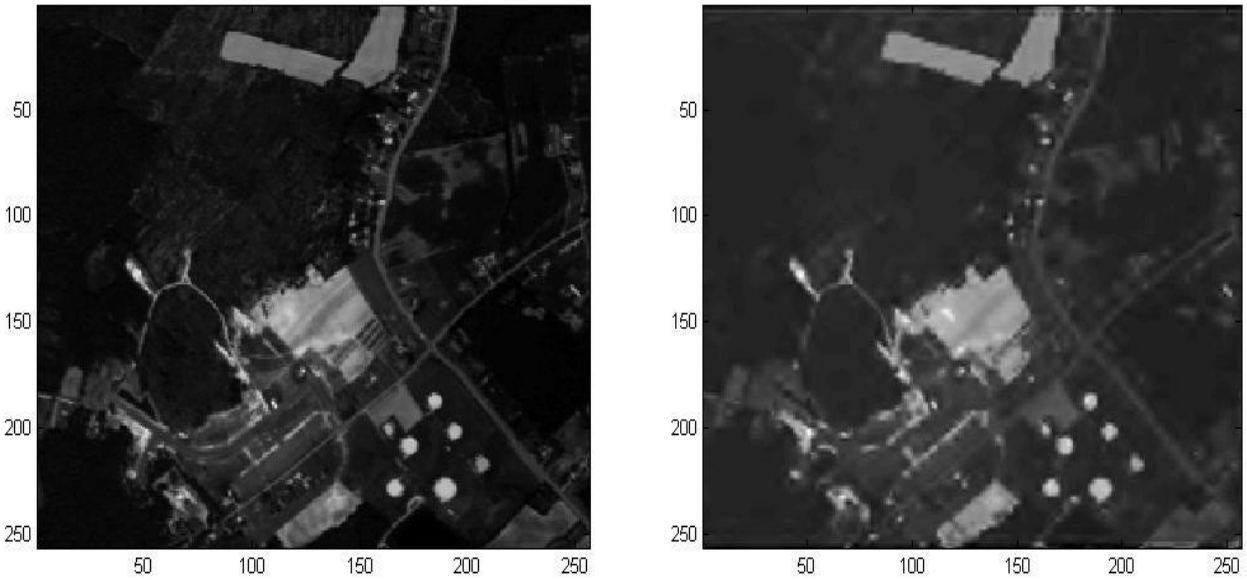


Fig. IV.6 – Image originale (gauche) (canal numéro 32 du cube 3D) et image compressée par un RC = 60 (droite).

Afin de permettre l'évaluation de la qualité de compression de l'algorithme élaboré, nous allons définir les critères de qualité suivants :

3. Critères d'évaluation de la compression :

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 $\tilde{I}$.

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. On notera également $\sum_x^{nx} \sum_y^{ny} \sum_\lambda^{n\lambda} I(x,y,\lambda)$ par $\sum_{x,y,\lambda} I(x,y,\lambda)$. Ces notations seront conservées par la suite.

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 **[2],** nous utiliserons ici des outils tels que le PSNR ou le SNR, ainsi que le MSE et le MAE, que nous implémenterons sous Matlab et intégrerons dans l'interface graphique élaborée.

- **Mean Absolute Error** (**MAE**) : C'est la moyenne de la différence en valeurs absolues entre les pixels de l'image originale et compressée.

$$MAE = \frac{1}{n_x\, n_y\, n_\lambda} \sum_{x,y,\lambda} (I(x,y,\lambda) - \tilde{I}(x,y,\lambda) \quad \text{----------(1)}$$

- **Erreur Quadratique Moyenne** (**EQM**) ou **Mean Square Error** (**MSE**):
  Elle représente l'erreur quadratique moyenne entre l'image compressée et l'image originale. Cette mesure va permettre d'évaluer l'influence des dégradations de la compression sur l'image compressée. Elle est définit comme suit :

$$MSE = \frac{1}{n_x\, n_y\, n_\lambda} \sum_{x,y,\lambda} (I(x,y,\lambda) - \tilde{I}(x,y,\lambda))^2 \qquad \text{----------(2)}$$

Pour une meilleure interprétation, cette mesure peut être reliée à des notions de moyenne et variance. En notant µ la moyenne et σ^2 sa variance.

$$MSE = \sigma^2_{I-\tilde{I}} + \mu^2_{I-\tilde{I}} \qquad \text{----------(3)}$$

- **Rapport Signal sur Bruit** (**RSB**) ou **Signal to Noise Ratio** (**SNR**). C'est le rapport entre la puissance de l'erreur avec la puissance du signal original (PS).

$$SNR_{(db)} = 10\, log_{10}\, \frac{PS}{MSE} \qquad ------(4)$$

Lorsque le signal est de moyenne nulle, on a $PS = \sigma_I^2$. En revanche, quand ce n'est pas le cas, une ambiguïté existe sur la définition de la puissance du signal, certains gardent la variance pour exprimer la puissance du signal et d'autres choisissent la puissance réelle du signal.

- 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.

$$PSNR_{(db)} = 10 log_{10} \frac{PeakSignal^2}{MSE} \qquad \text{----------(5)}$$

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.

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 (RC) :

$$RC = \frac{Nombre\ de\ bits\ de\ l'image\ originale}{Nombre\ de\ bits\ de\ l'image\ comprimée} = \frac{Dc}{Do} \quad \text{---------(6)}$$

Lorsque l'on souhaite préserver intégralement la qualité de l'image (compression sans perte), le calcul du débit entropique De, permet de connaître le débit minimal que l'on peut atteindre sans dégrader l'information. Il est défini par :

$$De = H_0 = -\sum_{m=1}^{M-1} p(x_m) \log_2(p(x_m)) \quad \text{----------(7)}$$

Où : $p(x_m)$ correspond à la probabilité d'apparition du niveau de gris x_m dans l'image.

Une manière de donner un ordre de grandeur de l'entropie (ou au moins une borne supérieure) des images hyperspectrales est de tester les méthodes de codage sans pertes sans tenir compte de la nature de l'image. On a utilisé les utilitaires suivant :

– gzip : algorithme LZ77;

– bzip2 : algorithme de Burrows-Wheeler, suivi de Huffman.

Original		**gzip**		**bzip2**	
Taille (Mb)	**bpp**	**Taille (Mb)**	**bpp**	**Taille (Mb)**	**bpp**
134.4Mb	16	82.9Mb	9.86	69.6 Mb	8.23
		RC obtenu =1.61		RC obtenu =1.92	

Tableau.IV.1 Estimation de l'entropie des images hyperspectrales.

Ceci nous permet seulement de comprendre que l'entropie des images hyperspectrales est inférieure à 8.23 bits (débit minimum en bits/pixel obtenu à l'aide d'un des algorithmes de compression sans perte standard).

4. Testes et Résultats de la compression :

Les algorithmes mentionnés ont été mis en œuvre sous Matlab R2009a 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), Symlettes (symN) 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 :

Tableau.IV.2 Table des filtres utilisés dans les tests.

Famille : Abréviation : Ordre du filtre utilisé:	Daubechies. DbN. N = 4, 6, 8, 10, 12, 14.
Famille : Abréviation : Ordre du filtre utilisé:	Symlettes. SymN. N = 4, 6, 8, 10, 12, 14.
Famille : Abréviation : Ordre du filtre utilisé:	Biorthogonales. BiorN. N = 2.2 (LeGaLL5/3) et 4.4 (CDF9/7).

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 32 images qui sont prises du canal 1 jusqu'au canal 32 mais nous pouvons utiliser n'importe quelle séquence, 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 256 x 256 x 32 ce qui correspond à 4 Mb ; L'application développée peut être appliquée sur toute la séquence, nous avons choisi de réduire la taille de la séquence utilisée afin de réduire le temps de calcul nécessaire à l'exécution de l'algorithme. La séquence choisie est illustrée dans la figure (IV.7).

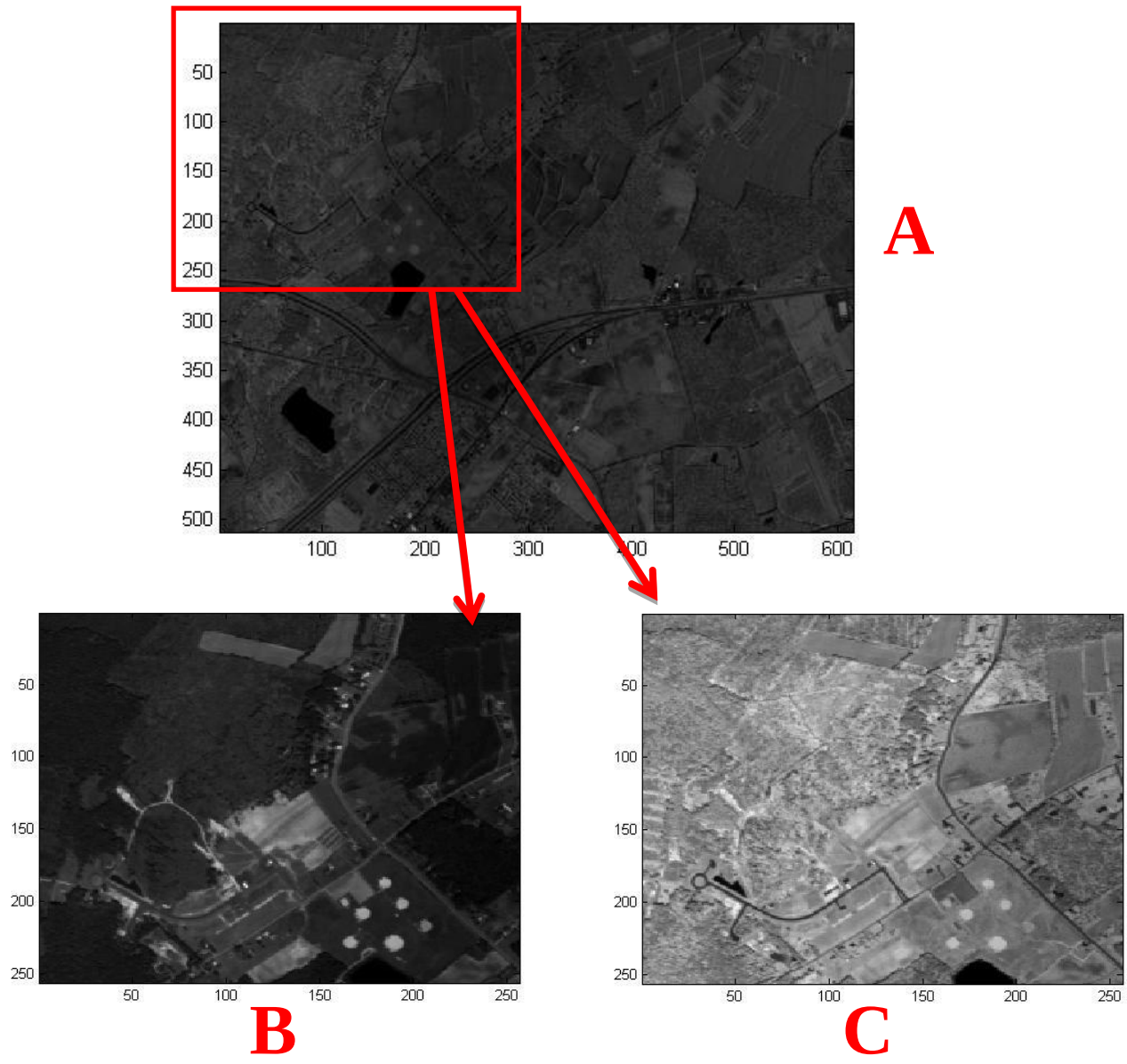


Fig. IV.7 – (A) Canal 30 de l'image AVRIS Yellowstone ; (B) et (C) canal 35 et canal 50 de la séquence utilisée pour les tests.

4.1 Influence du type de filtre :

Les tableaux (IV.3, IV.4 et IV.5) 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.

Rapport de compression (20---100)

Filtres	20	30	40	50	60	70	80	90	100
DB4	47.4115	44.3444	42.4607	40.6042	39.4439	38.3668	37.1447	35.7136	35.1316
DB6	47.6617	44.1923	42.4199	40.5623	39.0322	37.4338	36.3803	34.9966	34.7703
DB8	48.0391	45.3517	43.3142	41.4310	40.2712	38.8902	37.8840	36.1289	35.3609
DB10	42.1822	41.2011	40.1793	39.0097	38.0987	36.7174	35.5967	34.2446	33.6912
DB12	25.5736	25.5438	25.5059	25.4316	25.3666	25.2371	25.1204	24.9833	24.8649
DB14	23.4893	23.4517	23.3092	23.3678	23.4009	23.3678	23.3092	23.4517	23.3837

Tableau.IV.3 PSNR (dB) en fonction du rapport de compression pour différentes ondelettes (DbN).

Rapport de compression (20---100)

Filtres	20	30	40	50	60	70	80	90	100
Sym4	47.3189	43.7494	41.8817	40.2533	39.1672	37.7722	37.1442	35.9901	34.8440
Sym6	47.6616	44.1923	42.4199	40.5623	39.0322	37.4337	36.3802	34.9965	34.7703
Sym8	47.9784	45.3517	43.3142	41.4310	40.2712	38.8902	37.8840	36.1289	34.9093
Sym10	48.1206	45.7080	43.7246	42.1809	41.2420	39.5591	38.4852	37.2597	36.0977
Sym12	44.9920	43.1220	41.8191	40.5554	39.6314	38.1708	36.9220	36.0670	34.9825
Sym14	37.1891	36.9051	36.7235	36.2135	35.7982	35.2958	34.8484	34.3721	33.1832

Tableau.IV.4 PSNR (dB) en fonction du rapport de compression pour différentes ondelettes (SymN).

Rapport de compression (10---100)

Filtres	10	20	30	40	50	60	70	80	90	100
CDF 9/7	51.7183	47.9307	45.4117	43.3666	41.4060	40.0950	38.5667	37.2918	36.4086	35.3807
LeGaLL 5/3	52.8186	48.1745	45.3076	42.8423	41.1709	39.3908	38.0484	36.5127	35.5526	34.7816

Tableau.IV.5 PSNR (dB) en fonction du rapport de compression pour différentes ondelettes (biorN).

Nous rappelons que pour des images de bonne qualité, le PSNR varie entre 30 et 40 dB. Comme nous le voyons sur les tableaux, les PSNR obtenus dépassent les 30 dB même pour des rapports de compression élevés (RC = 100→ 0.16 bpp) sauf pour l'ordre N=12 et N=14 pour la famille d'ondelette de Daubechies; Pour des rapports de compression peu élevés, nous obtenons un PSNR très bon, dépassant les 40 dB allant jusqu'au-delà des 50 dB (RC = 10→ 1.6 bpp) pour certain ordre de filtre.

La solution optimale obtenue pour chaque famille (solution optimale par rapport à l'ordre du filtre dans chaque famille) est mentionnée par le rouge dans les tableaux et sont les suivantes :

- Pour les ondelettes de Daubechies ; la DB8 donne les meilleurs résultats avec un PSNR allant de 35.35→ 48.03dB.
- Pour les ondelettes Symlettes ; la sym10 avec un PSNR allant de 36.09→48.12dB.
- Pour les ondelettes Biorthogonales ; les deux ondelettes testées obtiennent de bons résultats, avec la Bior2.2 (LeGaLL5/3) allant de 34.78→48.17 dB et la Bior4.4 (CDF9/7) allant de 35.38→47.93 dB.

Les figures (IV.8.a, IV.8.b et IV.8.c) montrent l'évolution du PSNR en fonctions du rapport de compression pour les trois familles d'ondelettes (Symlettes, Daubechies et biorthogonales) et pour différents ordres.

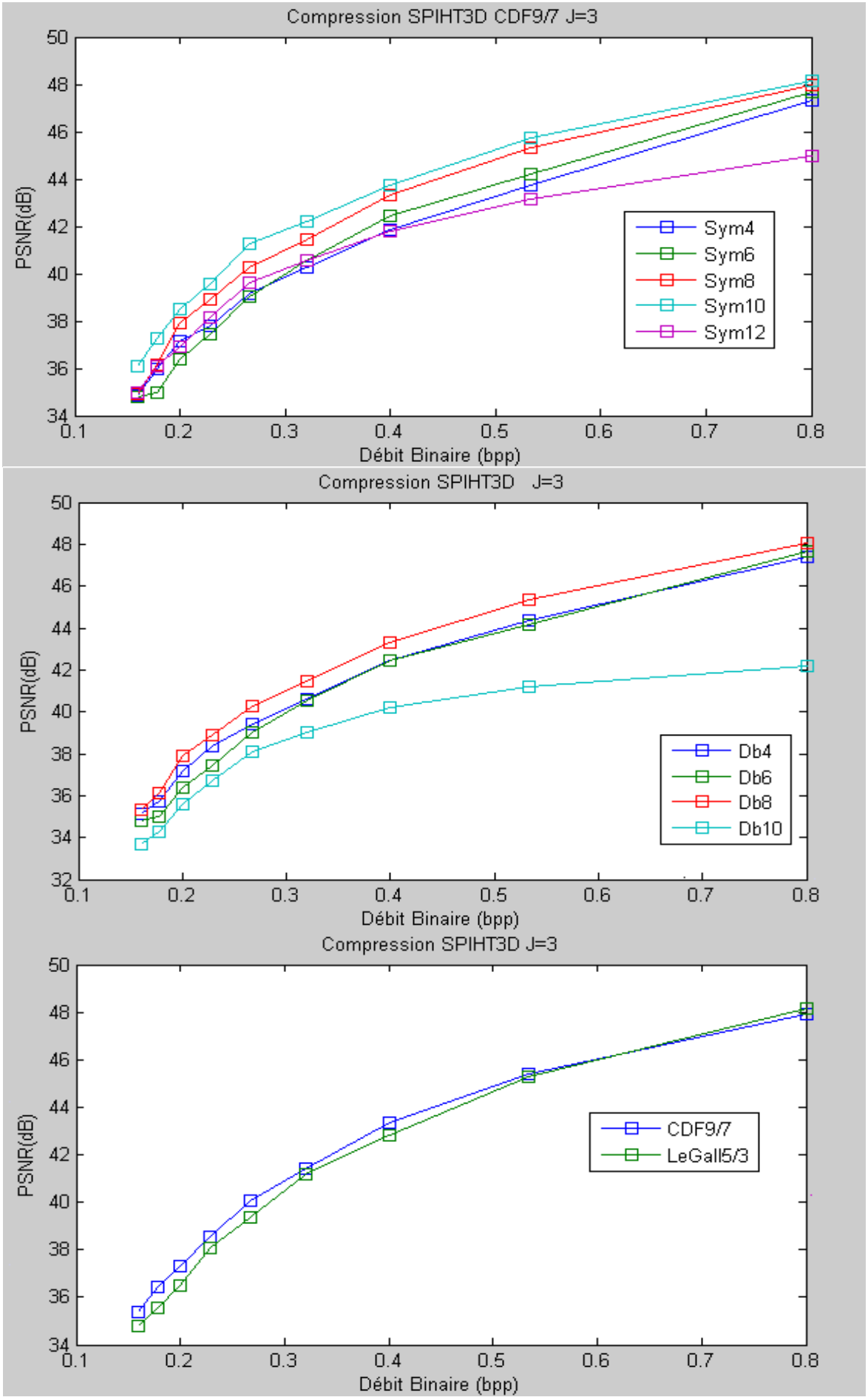


Fig.IV.8 (a, b et c) PSNR en fonctions du RC pour (Symlettes, Daubechies et biorthogonales).

Tel que mentionné précédemment, l'un des avantages du SPIHT est qu'il fait un codage progressif en commençant par les bits les plus significatifs vers les moins significatifs. Ceci permet à l'algorithme de faire de la transmission progressive d'images, puisque le décodeur peut s'arrêter n'importe où dans la suite de bits transmise et produire la meilleure image reconstruite. Au niveau du récepteur, on peut choisir la qualité de l'image ; quand on augmente le débit, nous obtenons des informations plus détaillées et bien sûr la qualité de l'image est meilleure. La figure (IV.9) montre la qualité de compression obtenue pour différents rapports de compression. Vous pouvez observer clairement un effet de bloc dans les rapports de compression élevés, cela est dû à la faible quantité d'information contenue dans les images (grand rapport de compression), puis quand on ajoute de l'information, la qualité de l'image augmente (faible rapport de compression).

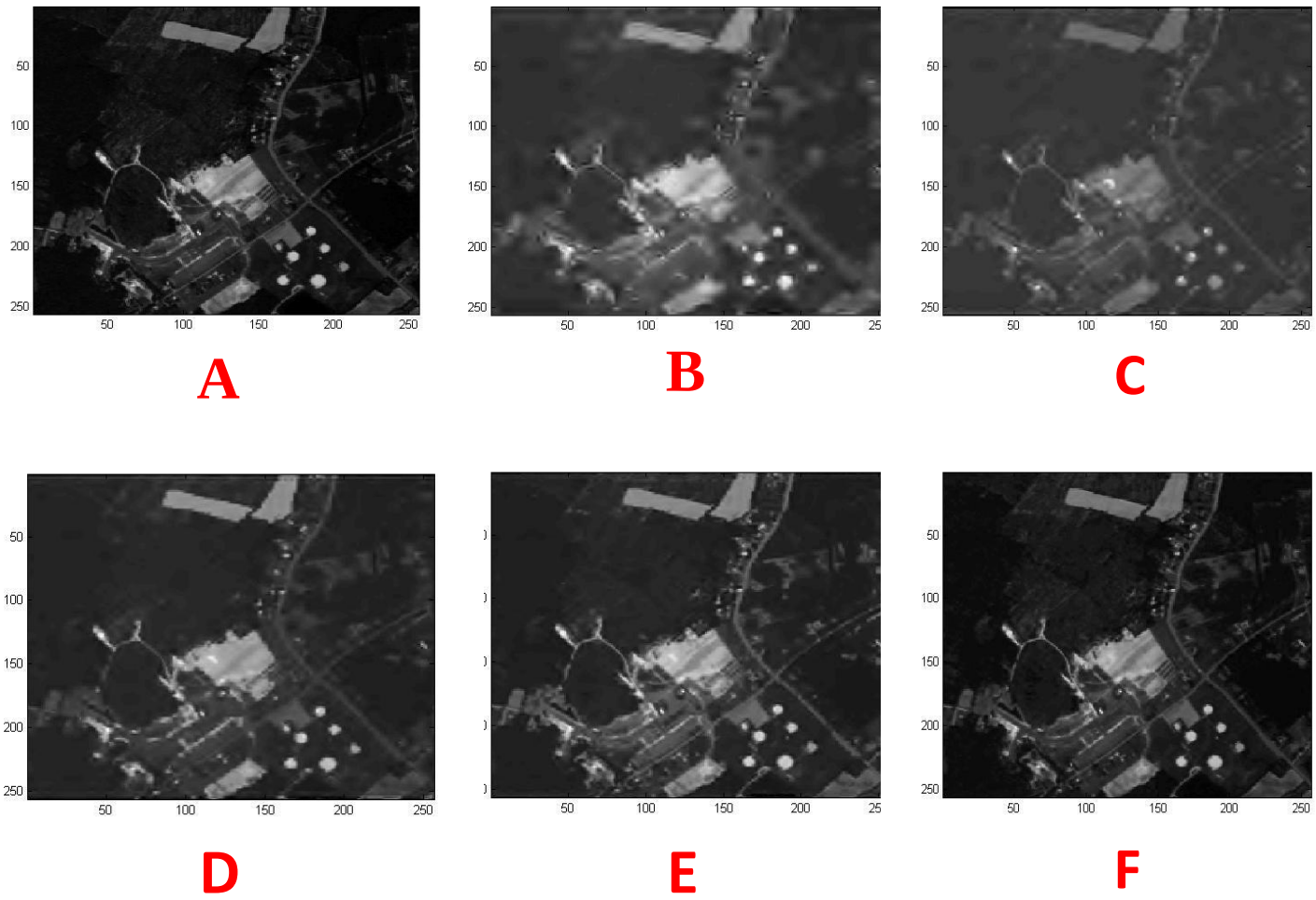


Fig. IV.9 – (A) : Image originale ; (B) : RC=100 ; (C) : RC=80 ; (D) : RC=60 ; (E) RC=40 ; (F) RC=20.

4.2 Influence du nombre de décomposition :

Les tableaux (IV.6, IV.7, IV.8 et IV.9) montrent l'effet du niveau de décomposition du filtre pour différentes familles d'ondelettes. Nous observons une nette amélioration du PSNR, lorsque le niveau de décomposition augmente pour tous les filtres et pour toutes les méthodes, que ce soit les méthodes 2D ou 3D. Les valeurs max de PSNR sont obtenues pour le niveau de décomposition J = 3 (Note : Le niveau de décomposition max utilisable pour notre séquence est J = 4 parce que l'axe z = 32). A partir du niveau de décomposition J = 3, il n'y a pas d'amélioration du PSNR obtenu. Ceci s'explique par le fait que, quand le niveau de décomposition augmente de plus amples détails sont ajoutés, mais plus d'énergie est perdue et les erreurs de reconstruction augmentent. C'est ce qui cause un effet négatif après un certain niveau et c'est le cas pour le niveau J = 4.

PSNR (dB) pour différents niveaux de la DB8

RC	20	30	40	50	60	70	80	90	100
J=4	22.2376	22.2308	22.2243	22.2176	22.2176	22.2055	22.1981	22.1882	22.1847
J=3	48.0391	45.3517	43.3142	41.4310	40.2712	38.8902	37.8840	36.1289	35.3609
J=2	33.0147	27.9174	22.5751	22.5228	20.7357	19.8972	19.8972	19.8972	19.8972

Tableau.IV.6 PSNR (dB) pour différents fonction du niveau de décomposition.

PSNR (dB) pour différents niveaux de la Sym10

RC	20	30	40	50	60	70	80	90	100
J=4	25.3664	25.3541	25.3439	25.3326	25.3251	25.3154	25.3057	25.2945	25.2755
J=3	48.1206	45.7080	43.7246	42.1809	41.2420	39.5591	38.4852	37.2597	36.0977
J=2	34.6828	28.9583	23.1423	23.0407	21.4840	19.9059	19.9059	19.9059	19.9059

Tableau.IV.7 PSNR (dB) pour différents fonction du niveau de décomposition.

PSNR (dB) pour différents niveaux de LeGaLL 5/3

RC	20	30	40	50	60	70	80	90	100
J=4	41.1019	40.7454	40.263	40.055	39.591	39.1542	38.7594	38.6001	38.404
J=3	48.1745	45.3076	42.8423	41.1709	39.3908	38.0484	36.5127	35.5526	34.7816
J=2	34.5619	30.3925	24.0802	24.0758	21.8547	20.335	20.335	20.335	20.335

Tableau.IV.8 PSNR (dB) pour différents fonction du niveau de décomposition.

PSNR (dB) pour différents niveaux de la CDF9/7

RC	20	30	40	50	60	70	80	90	100
J=4	27.6249	27.6099	27.5906	27.5608	27.5466	27.5216	27.4955	27.4674	27.4429
J=3	47.9307	45.4117	43.3666	41.4060	40.0950	38.5667	37.2918	36.4086	35.3807
J=2	33.0145	28.1296	22.1988	22.1988	20.7078	19.9099	19.9099	19.9099	19.9099

Tableau.IV.9 PSNR (dB) pour différents fonction du niveau de décomposition.

Les résultats obtenus montrent qu'en augmentant le nombre de niveaux de décomposition, on obtient une nette amélioration des résultats et cela est dû à la notion des arbres de zéro sur laquelle se basent les méthodes EZW et SPIHT avec leurs deux versions 2D et 3D. En fait, en augmentant le nombre de décompositions, cela permet une meilleure concentration de l'information utile dans un nombre réduit de coefficients dans les sous-bandes basses fréquences. Ainsi, le nombre de coefficients dans l'arbre de zéro sera plus grand, but de la notion des arbres de zéro, qui est de regrouper le plus de coefficients dans un seul arbre. Ce qui permet un meilleur codage de la carte de signification.

4.3 Influence du contenu de l'image :

Une image satellitaire se compose d'un tableau de pixels contenant les valeurs de l'intensité de la réflectance. Cette intensité détermine les coefficients bas fréquences (approximation) et les coefficients hauts fréquences (détails), ce qui affecte la façon dont est compacté l'énergie et donc la compression. La taille de l'image AVIRIS est de 512x614x224. Nous avons utilisé pour ce test, quatre séquences d'images issues de notre image Yallowstone. Au fait, nous avons divisé notre scène AVIRIS en quatre parties, chaque partie contient des informations spécifiques de l'image et chacune correspond à une taille de 256x256x32. Les séquences choisies sont illustrées dans la figure suivante et le tableau donne les résultats montrant l'effet du contenu de chaque séquence en utilisant l'ondelette CDF9/7.

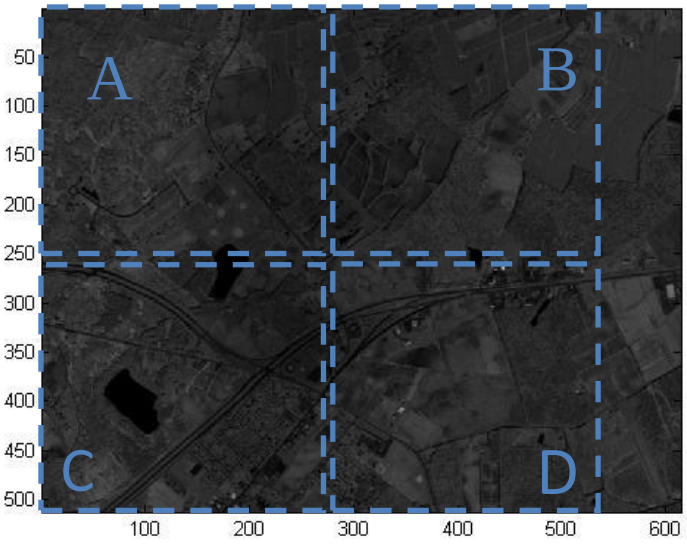


Fig. IV.10 – Effet du contenu de chaqu'une des quatre séquences en utilisant l'ondelette CDF9/7.

PSNR (dB) pour différents séquences.	
Séquence A RC = 50 : 41.4061 dB RC = 100 : 35.3807 dB	Séquence B RC = 50 : 36.0969 dB RC = 100 : 30.8403 dB
Séquence C RC = 50 : 42.2167 dB RC = 100 : 35.3834 dB	Séquence D RC = 50 : 43.7122 dB RC = 100 : 36.7033 dB

Tableau.IV.10 PSNR (dB) obtenu pour les quatre séquences avec l'ondelette CDF9/7 et J=3.

Nous constatons bien qu'avec la même ondelette et le même niveau de décomposition appliqué pour les quatre séquences, nous avons obtenu des résultats différents, notamment pour la séquence B, ou on a obtenu l'écart le plus important par rapport aux autres séquences.

4.4 Influence de la taille de l'image :

L'algorithme peut être mis en œuvre pour toute taille d'image, mais quand la taille de l'image augmente le temps nécessaire à la compression et la reconstruction de l'image augmente aussi. Le tableau (IV.11) montre l'effet de la taille de l'image sur le temps de calcul et sur qualité de compression en utilisant l'ondelette CDF9/7 pour un RC =100.

Influence de la taille de l'image.

Scène spatiale

Scène spectrale

128x128x32	256x256x32
J : 3 PSNR : **35.4339** T=5 sec	J : 3 PSNR : **35.3807** T=1 min
128x128x64	**256x256x64**
J : 3 PSNR : 32.5385 T=10 sec	J : 3 PSNR : 32.8579 T=3 min
J : 4 PSNR : **36.9398** T=30 sec	J : 4 PSNR : **36.0967** T=10 min
128x128x128	**256x256x128**
J : 3 PSNR : 34.9114 T=20 sec	J : 3 PSNR : 33.1755 T=3 min
J : 4 PSNR : **39.5542** T= 2min,30sec	J : 4 PSNR : **39.3307** T=30 min
J : 5 PSNR : **39.3759** T= 2min,40sec	J : 5 PSNR : **39.6357** T=40 min

Tableau.IV.11 PSNR (dB) obtenu pour des séquences de différentes tailles ; RC=100.

Nous pouvons constater effectivement que plus la taille de la séquence augmente que ce soit celle de la scène spatiale (x,y) ou/et celle de la scène spectrale (z) le temps d'exécution de l'algorithme augmente. Deuxièmement nous constatons que, quand la taille de la scène augmente, nous obtenons de meilleurs résultats, en particulier quand la scène spectrale augmente, les résultats sont très bons du fait de la forte corrélation spectrale, ainsi les arbres de zéros regroupent plus de coefficients dans un seul arbre, ce qui permet un meilleur codage.

L'amélioration des résultats obtenus en augmentant la taille de la scène spectrale est illustrée dans la figure suivante :

Fig. IV.11 – (A) : Scène 256x256x32 PSNR=35.38 ; (B) : Scène 256x256x64 PSNR=36.09; (C) : Scène 256x256x128 PSNR=39.33 ; compression par CDF9/7 et pour un RC =100.

Nous pouvons déduire d'après ces résultats, qu'il est plus intéressant de concevoir notre algorithme de façon à traiter les images à compresser par petit bloc, en réduisant la taille de la scène spatiale et en privilégiant celle de la scène spectrale. Ceci permettra de tirer profit pleinement de la forte corrélation spectrale sans sacrifier le cout calculatoire de l'exécution de l'algorithme (en réduisant la taille de la scène spatiale, nous réduisons le temps d'exécution sans sacrifier la qualité, car sur la scène spatiale, la corrélation est à faible distance).

4.5 Performance SPIHT 2D vs. SPIHT 3D :

Pour réaliser cette comparaison, nous avons **compressé chacun des 32 canaux de notre séquence d'image séparément** et calculer pour chaque canal le PSNR. La figure (IV.12) montre ces résultats pour l'ondelette CDF9/7 et pour un débit binaire allant de 0.1 à 1 bpp. Pour faire la comparaison, nous avons calculé le PSNR3D pour cette séquence, c-à-d une fois que tous les canaux ont été compressés séparément par la TOD 2D et SPIHT 2D, nous les avons regroupés pour former le cube 3D compressé et nous avons calculé son PSNR3D. Ensuite, nous avons comparé les résultats à ceux (PSNR3D) trouvés avec la

compression utilisant la TOD 3D et SPIHT 3D. La figure (IV.13) montre le PSNR3D pour les deux cas de compression, celle impliquant une TOD 2D, et celle impliquant une TOD 3D.

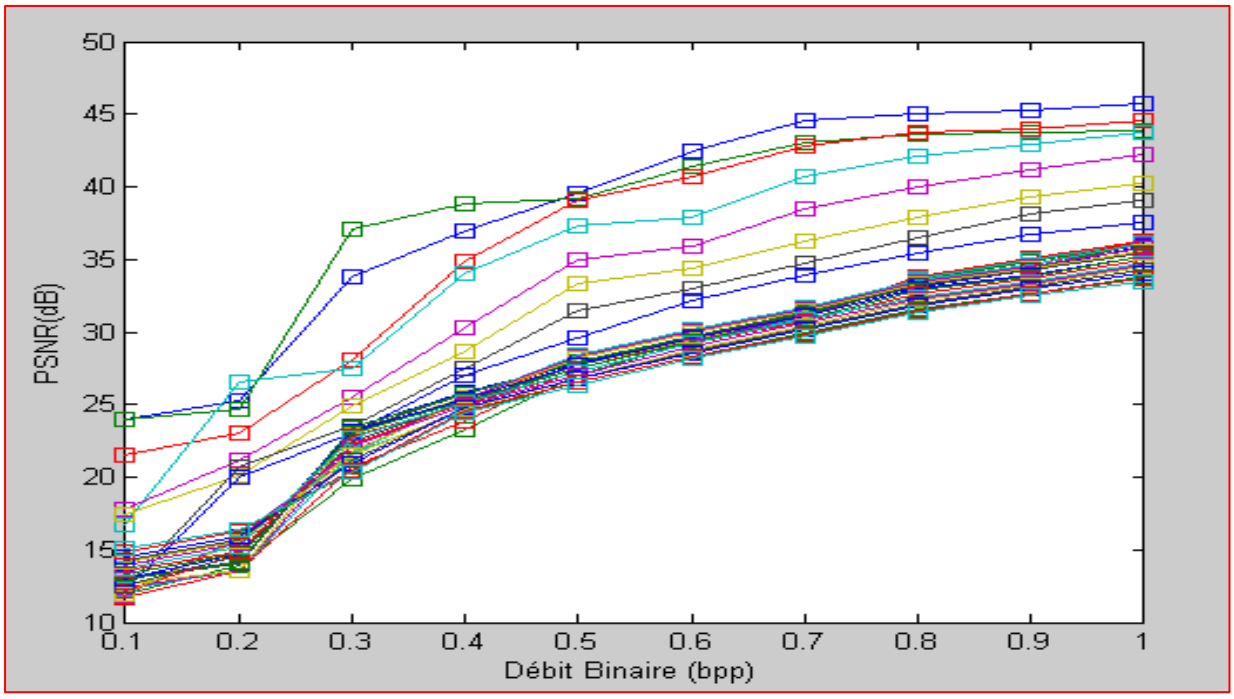


Fig. IV.12 – PSNR pour chacun'un des 32 canaux de la séquence compressé.

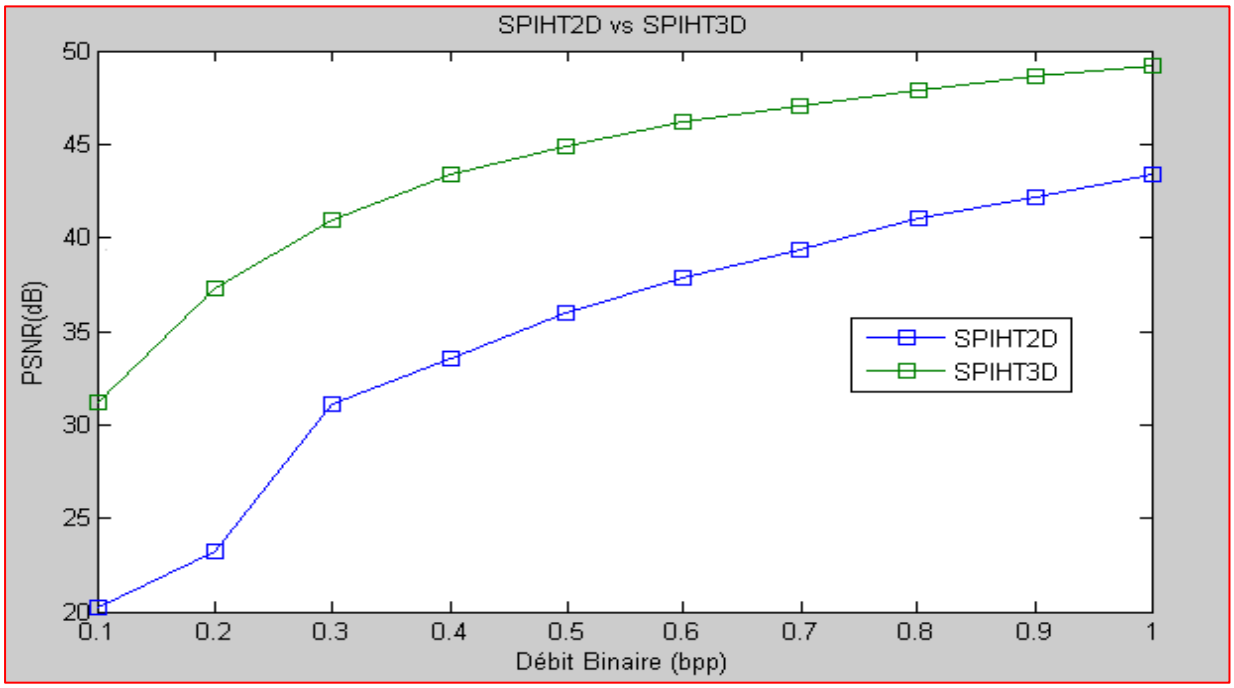


Fig. IV.13 – PSNR3D résultants de la compression par SPIHT 2D et par SPIHT 3D.

En comparant les courbes de distorsion du SPIHT 2D et du SPIHT 3D, nous pouvons constater que la version 3D, garantie plus de **10 dB** d'amélioration à tous les rapports de compression. Cela implique que les images hyperspectrales sont très corrélées et en utilisant une TOD 3D, on peut exploiter pleinement ces corrélations inter-bandes et ainsi atteindre de bien meilleures performances que les versions 2D qui codent les images séparément.

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, de l'ordre du filtre ondelettes et des images elles-mêmes qui ont un effet drastique sur la compression, 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. 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 32 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. Le codeur SPIHT 3D tire profit de la forte corrélation spectrale et offre un gain de codage bien supérieur à celui offert par le SPIHT 2D, les résultats ont montré un gain allant jusqu'à 10 dB pour tous les rapports de compression (sur notre séquence de 32 canaux).

Conclusion

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 hyperspectrale 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. C'est ce qu'on appelle le zoom multiéchelle. Il existe donc des dépendances entre les représentations de l'image à différentes échelles. Cette propriété est à la base des méthodes de codage de type arbres de zéros, comme l'algorithme EZW et SPIHT.

Ensuite nous avons proposé un schéma de codage inter-bandes pour caractériser les dépendances des coefficients d'ondelettes localisés dans les sous-bandes ayant la même orientation. Ce schéma de codage est basé sur l'hypothèse des arbres de zéros. Nous avons choisi d'implémenter l'approche inter-bandes : SPIHT avec ses deux variantes, celles impliquant une TOD 2D (SPIHT 2D) et celles impliquant une TOD 3D (SPIHT 3D). Ces approches utilisent les arbres de zéros pour le codage des coefficients d'ondelettes et exploitent les redondances inter-échelles dans les différentes dimensions. De plus cette approche de type intra-bandes, nous permet une quantification progressive des coefficients en commençant par les coefficients les plus significatifs (importants) vers les moins significatifs. Ceci permet à l'algorithme SPIHT de faire de la transmission progressive d'images.

Nos résultats expérimentaux montrent que notre technique permet d'avoir un bon compromis entre la qualité des images reconstruites et le taux de compression. En effet, nous avons obtenu un PSNR qui varie entre 35-55 dB, pour des rapports de compression allant de 10 à 100 en fonction de l'ondelette utilisée. Notre Algorithme basé sur la transformée en ondelettes TOD 3D et le codeur SPIHT 3D, tire profit de la forte corrélation spectrale et

permet d'atteindre un gain de codage bien supérieur à celui offert par le codeur SPIHT 2D. En effet, nous avons pu obtenir un gain supérieur de plus de 10 dB par rapport au standard à deux dimensions (SPIHT 2D) pour tous les rapports de compression sur notre séquence de 32 canaux. Notons qu'en augmentant le nombre de canaux de la séquence ces résultats seraient encore meilleur).

L'ensemble de ces résultats, permettent de faire les observations suivantes :

- L'utilisation d'une transformée en ondelettes 3D en présence d'une pile d'images augmente sensiblement les performances de compression. Les méthodes 3D sont toujours meilleures que les méthodes 2D.
- L'approche inter-bande 3D pour le codage avec pertes, présente de meilleurs résultats que pour le codage sans pertes. L'utilisation des méthodes 3D ne permet pas d'atteindre de très forts taux de compression lorsqu'il s'agit de compression sans pertes. La compression avec pertes maîtrisée est donc une réponse à cette limitation.

A travers nos résultats et ceux publiés dans la littérature, nous pouvons avancer qu'une technique performante de compression pour 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. Ces perspectives peuvent s'articuler sur deux thèmes majeurs qui sont :

Amélioration de l'algorithme de 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.

- Utiliser un schéma appelé lifting pour calculer la TO discrète. Celui-ci permet de réduire le coût calculatoire par rapport à l'implantation par convolution, car il utilise les redondances mathématiques entre le filtre passe-bas et le filtre passe-haut. Son second avantage est sa capacité à produire des coefficients d'ondelettes entiers. Cette fonctionnalité est fondamentale pour pouvoir envisager des méthodes de compression sans perte ou allant d'un schéma avec pertes vers du sans perte.

- Utilisation d'autres codeurs inter-bandes ou intra-bandes afin de trouver le codeur le plus approprié, qui permet en effet d'exploiter de manière efficace à la fois les corrélations existantes au sein d'un canal, mais également entre les canaux.

Définition de critères d'évaluation de qualité de compression d'images hyperspectrales :

- Le PSNR, ainsi que le MSE ou SNR, ne permettent pas de rendre compte de la nature de la dégradation et donc ne permet pas de prévoir l'impact sur les applications des images hyperspectrales. Nous proposons donc de définir et de chercher des critères de qualité complémentaires, qui permettent de qualifier le type de dégradation appliquée à l'image et de donner un ordre d'idée des performances de l'application sur les images après compression **[2].**

Annexe A :

AVIRIS

Le capteur AVIRIS :

AVIRIS (Airborne Visible / Infrared Imaging Spectrometer) est l'un des tout premier imageur à spectromètre, il a été mis au point en 1987 par la NASA. Ce capteur 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 4m-20m). Le tableau suivant résume les caractéristiques du capteur :

Caractéristiques d'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

Fig. A1 Caractéristiques du capteur AVIRIS.

Les images AVIRIS :

Le "cube image" montre les données obtenues par l'instrument AVIRIS monté sur un avion U-2 de la NASA à une altitude variant entre 4.000 mètres à 20.000 mètres au-dessus de la scène à photographier (voir Fig.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 (2.500 nanomètres). Les côtés sont des pseudo-couleurs, allant du noir au bleu (faible réponse) et rouge (réponse élevée).

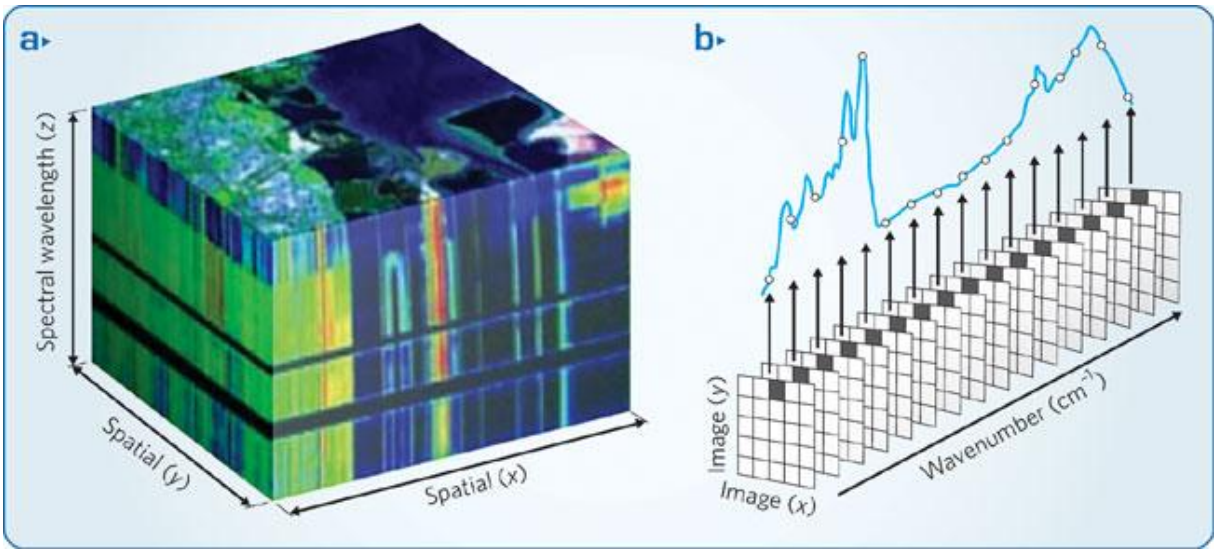


Fig. A2 : (a) Le **cube image** des données AVIRIS ; (b) Bandes spectrales contiguës et continues.

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 Fig.A2.b).

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, avec des bandes spectrales contiguës permettant ainsi des applications fines comme :

1. Détection de cibles,
2. Identification des matières,
3. La cartographie des matériaux,……

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.

L'image AVIRIS « Yellowstone »:

Pour notre étude, nous avons utilisé des séquences hyperspectrales de la base de données AVIRIS. En effet, nous avons utilisé les données de la scène Yellowstone, acquise en 2006. Cette image AVIRIS calibrée peut être téléchargée à partir de : http://aviris.jpl.nasa.gov/html.

Les canaux 2D des images hyperspectrales que nous avons utilisés pour la compression sont de taille 256 × 256 × 1 chacun et nous avons utilisé une séquence de 32 images. 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 notre séquence est de 256 x 256 x 32 ce qui correspond à 4 Mb. La séquence choisie est illustrée dans la figure suivante :

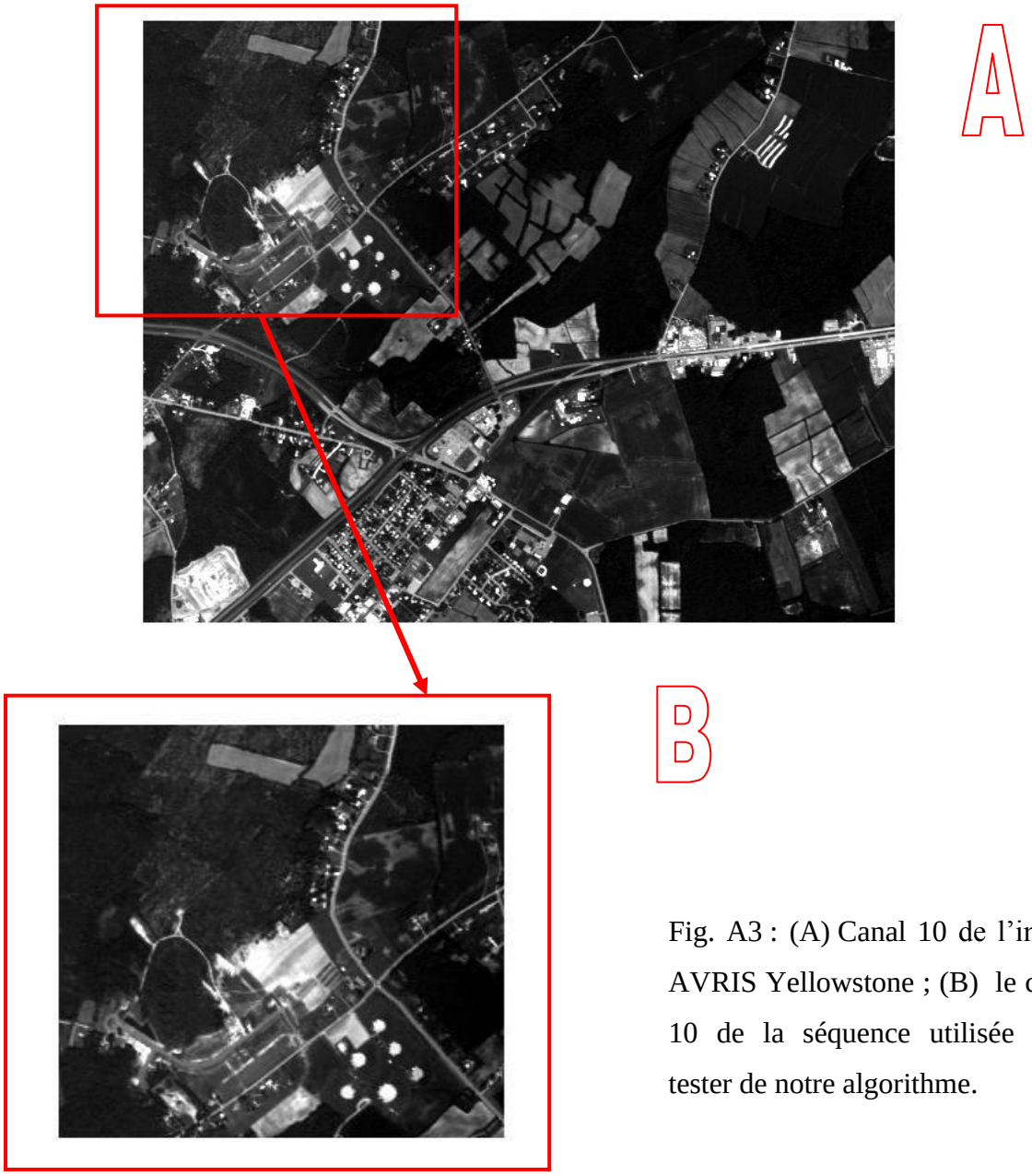


Fig. A3 : (A) Canal 10 de l'image AVRIS Yellowstone ; (B) le canal 10 de la séquence utilisée pour tester de notre algorithme.

Annexe B :

ARBRE DE ZEROS

ARBRES DE ZEROS :

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.Shapiro** avec 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. Cho** a 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 la racine sont égaux à zeros 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).

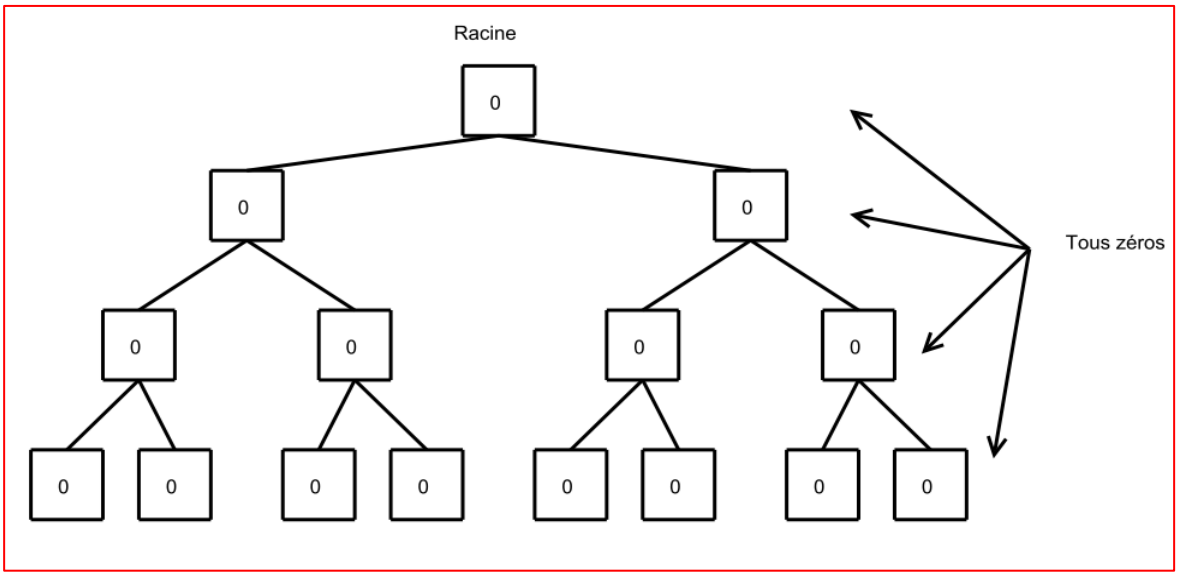


Fig. B1 Arbres de zéros de degré 0.

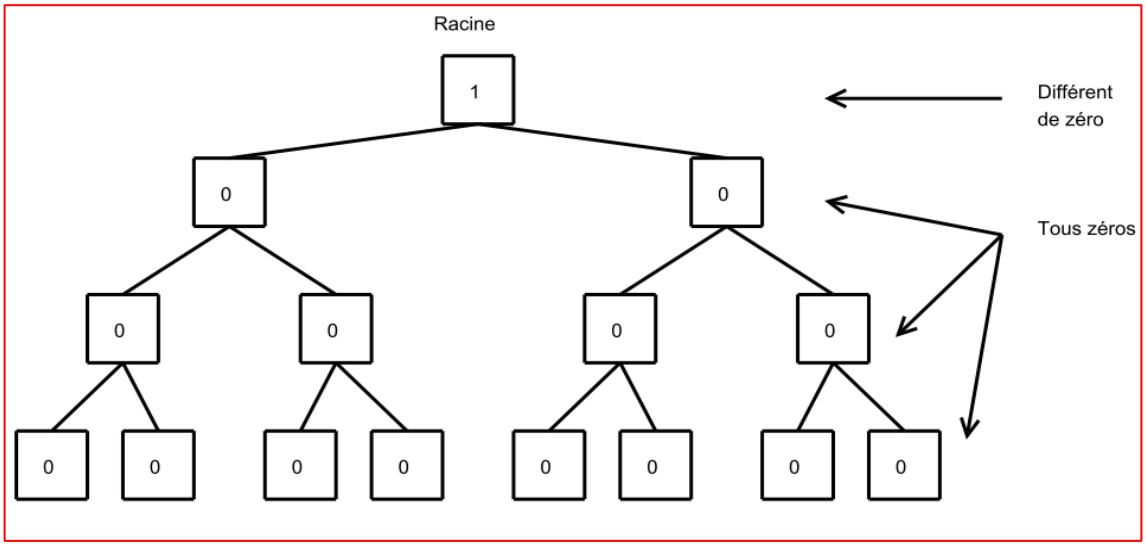


Fig. B2 Arbres de zéros de degré 1.

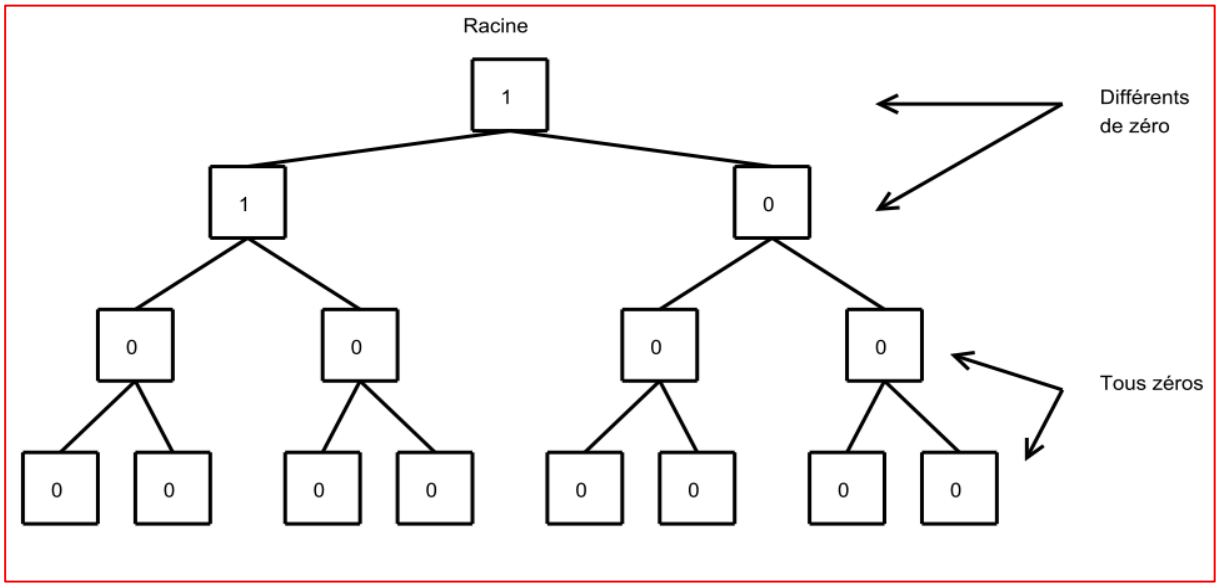


Fig. B3 Arbres de zéros de degré 2.

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]. Said Homayouni (2005). **Caractérisation des Scènes Urbaines par Analyse des Images Hyperspectrales.** Thèse de Doctorat, INFORMATIQUE, TELECOMMUNICATIONS ET ELECTRONIQUE, 175p.

[2]. Christophe Emmanuel (2006). **Compression des Images Hyperspectrales et son Impact sur la Qualité des Données.** Thèse de Doctorat, Spécialité: Signal et image, université de Toulouse, France. 210p.

[3]. David Salomon (2004). **Data Compression. The Complete Reference.** Springer, 920p.

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

[5]. Giovanni Motta, Francesco Rizzo, James A. Storer (2005). **Hyperspectral Data Compression.** Springer, 420p.

[6]. S.G.Mallat. (1989). **A Theory for Multiresolution Signal Decomposition : the Wavelet Representation**", IEEE PAMI, Vol.11(7), pp.674−693.

[7]. Stephane.G Mallat**. A Wavelet Tour of Signal Processing - The Sparse Way.** Academic Press, 810p.

[8]. Shapiro, J. (1993). **Embedded image coding using Zerotree of Wavelet coefficients,** IEEE transactions on signal processing, vol. 41, no. 12, p. 3445-3462.

[9]. A. Said and W. A. Pearlman (1996). **A new and efficient image codec based on set partitioning in hierarchical trees,** IEEE Transactions on Circuits and Systems for Video Tech., vol. 6, pp. 243–250.

[10]. Pearlman, W. A et B.J. Kim (1997). **An Embedded Wavelet Video Coder Using Three Dimensional Set Partitioning in Hierarchical Trees (SPIHT).** In IEEE Data Compression Conference DCC'97, p 221-260.

[11]. David F. Walnut (2001). **An Introduction to Wavelet Analysis.** Springer, 450p.

[12]. Frédéric Truchetet (1998) **Ondelettes pour le signal numérique.** Editions Hermes.

[13]. Gaudeau, Y. (2006). **Contributions en compression d'images médicales 3D et d'images naturelles 2D.** Thèse de doctorat de l'Université Henri Poincaré, Nancy 1, Spécialité Automatique, Traitement du Signal et Génie Informatique, France.160p.

[14]. Demaeyer Jonathan, Bebronne Michael et Forthomme Sebastien. **Les Ondelettes.** Université libre de Bruxelles.

[15].C.Sidney Burrus, Ramesh A. Gopinath, Haitao Guo **Introduction to Wavelets and Wavelet Transforms.** Prentice hall, 270p.

[16]. Ingrid Daubechies (1992). **Ten Lectures on Wavelets.** Society for Industrial and Applied Mathematics. 340p.

[17]. Vivien Chappelier (2005). **Codage progressif d'images par ondelettes orientées.** Thèse de Doctorat. Université de Rennes 1.

[18]. Chui, C. 1992, **Wavelets: a tutorial in theory and applications**, Academic press inc.

[19]. Cohen, A. 1992, **Ondelettes et traitement numérique du signal**. Masson, Paris, 207p.

[20]. Cohen, A. Daubechies, I., Feauveau, J. C. (1992). **Biorthogonal Bases of Compactly Supported Wavelets**, Comm. On Pure and Applied Mathematics, vol.XLV, p. 485-560.

[21].C. Valens, (1999). **Embedded Zerotree Wavelet Encoding.** (rapport) 12p.

[22]. Asad Islam and William A. Pearlman (1998). **An embedded and efficient low-complexity hierarchical image coder,"** in Visual Communications and Image Processing '99, San Jose, CA, USA, pp. 294-305.

[23]. Christophe Emmanuel, Corinne Mailhes and Pierre Duhamel, (2008). **Hyperspectral Image Compression: Adapting SPIHT and EZW to Anisotropic 3D Wavelet Coding,** IEEE TRANSACTIONS ON IMAGE PROCESSING, VOL. 17, NO. 12.

[24]. Hans F. Grahn and Paul Geladi (2007). **Techniques and Applications of Hyperspectral Image Analysis.** Wiley, 400p.

[25]. Xiaoli Tang, William A. Pearlman and James W. Modestino (2004). **Hyperspectral Image Compression Using Three-Dimensional Wavelet Coding**, Rensselaer Polytechnic Institute Troy, NY.

[26]. Xiaoli Tang, William A. Pearlman and James W. Modestino. (2004). **Hyperspectral Image Compression Using Three-Dimensional Wavelet Coding: A Lossy-to-Lossless Solution,** Rensselaer Polytechnic Institute Troy, NY, 22p.

[27]. Cziho, A. (1999). **Quantification vectorielle et compression d'image, application à l'imagerie médicale**. Thèse de doctorat, école doctorale en informatique, traitement du signal et télécommunications, université de Rennes 1, France, 160 p.

[28]. Lahdir, M. (2008). **Nouvelle approche de compression d'images basée sur les ondelettes et les fractales : Application aux images Météosat**, Thèse de doctotat de l'Université de Mouloud Mammeri de Tizi-Ouzou, Spécialité Electronique, Option Télédétection, 128 p.

[29]. Lahdir, M., Ameur, S. et Akrour, L (2005). **Compression d'images numériques par fractal dans le domaine DCT.** Internationale Conférence: Sciences of Electronic, Technologies of Information and Telecommunication SETIT2005, 27-31 Mars 2005, Sousse, Tunisie.

[30]. Lahdir, M., Ameur, S. et Akrour, L. (2006). **Compression d'images par un codage fractal non itératif Implémenté dans le domaine DCT.** 4ème Conférence Internationale JTEA 2006, 12-14 Mai 2006, Tunisie.

[31]. Savaton, G. (1999). **Compression réversible et irréversible d'images médicales à base d'ondelettes entières.** Mémoire DEA. Université d'Angers.

[32]. Ziv, J. and Lempel, A. (1977). **A universal algorithm for sequential data compression,** IEEE transactions on information theory, vol. 23, no. 3, p 337-343.

[33]. Rabbani, M. and Santa Cruz, D. (2001). **JPEG2000 image coding standard. In Proc**. IEEE International Conference on Image Processing. ICIP '01, Thessaloniki, Greece.

[34]. Delaunay, X. (2008). **Compression d'images satellite par post-transformées dans le domaine ondelettes.** Thèse de doctorat, de l'université de Toulouse, Discipline : Signal, Image, Acoustique et Optimisation, France.