ICPS

Informatique et Calcul Parallèle de Strasbourg

LSIIT

Laboratoire des Sciences de l'Image, de l'Informatique et de la Télédétection

TRAITEMENT D'IMAGES ET PARALLÉLISME

Mise en œuvre d'un algorithme de compression/décompression d'images par ondelettes sur machine parallèle

10 juin 1996

Université Louis Pasteur Strasbourg

SALOMON Michel

DEA d'Informatique
1995-1996

Remerciements

Je remercie Guy René Perrin, Professeur à l'Université Louis Pasteur et directeur de l'ICPS, ainsi que Fabrice Heitz, Professeur à l'Ecole Nationale Supérieure de Physique de Strasbourg pour avoir proposé et encadré ce stage.

Je remercie également les membres de l'ICPS pour leur accueil.

Table des matières

Table des figures

Introduction

Dans les domaines liés aux satellites tels que la défense, la télédétection, la météorologie ou la cartographie la transmission et le stockage d'images ou de séquences vidéo posent problèmes vu leur taille. On trouve ces problèmes de transmission et de stockage également dans le multimédia et les moyens de communications visuelles numériques comme le visiophone, les systèmes de vidéo-conférence, le réseau internet, le fax ou la télévision numérique. De plus les liaisons numériques sont de débit limité, tout comme la taille des supports de stockage bien que leur capacité tend à croître tout en gardant un coût raisonnable.

Il apparaît donc clairement que la réduction de la quantité d'information à transmettre ou à stocker s'impose, d'où la nécessité d'avoir des schémas de compression. Ceci d'autant plus que l'espace occupé par les images et les séquences vidéo croît suivant leur taille et leur qualité comme le montre les exemples [1, 2] ci-dessous :

- Une image numérique classique de taille 256×256 dont un pixel est codé sur 8 bits occupe $256 \times 256 \times 8$ bits si elle en niveaux de gris et $256 \times 256 \times 8 \times 3$ bits si elle est en couleur.
- Une image satellite Spot à $20m$ a environ une taille de 6000×6000 pixels.
- Une photographie standard de $35mm$ numérisée avec une résolution de $12\mu m$ par pixel nécessite 18 mégaoctets pour son stockage.
- Une séquence vidéo couleur de qualité NTSC a une taille de 23 mégaoctets pour une durée de 1 seconde.
- La numérisation de la banque de données des empreintes digitales du Federal Bureau of Investigation (FBI) des Etats-Unis nécessite sans compression approximativement $1,140$ téraoctets. La carte des empreintes d'un individu ayant environ une taille de 10 mégaoctets, sa transmission par ligne téléphonique à 9600 bauds prendrait plus de trois heures.

D'autre part les applications telles que la visiophonie, la vidéo-conférence ou encore la télévision numérique et la compression des images à bord des satellites imposent un traitement temps réel. Il faut donc des méthodes de compression très sophistiquées permettant d'assurer la production d'une image toute les $1/25^{ème}$ de seconde et une implémentation VLSI pour les systèmes embarqués.

De nombreux schémas de compression ont été proposés et standardisés, on peut citer JPEG pour les images, MEPG I et II pour la compression de vidéos avec une qualité comparable à la télévision hertzienne et haute définition pour MPEG II. L'ensemble de ces standards tout comme H.261, le standard de compression en visiophonie reposent sur la transformée discrète en cosinus (DCT).

Néanmoins il y a quelques années est apparue la transformée discrète en ondelettes (*DWT*) [3], une transformation "temps-fréquence" qui est une alternative intéressante à la transformée en cosinus. La transformée en ondelettes a été largement exploitée dans le cadre de schémas de compression flexibles et efficaces, avec ou sans pertes (*lossy ou lossless*). Cette transformation permet d'atteindre des taux de compression importants sans produire les artefacts (effets de bloc) observés sur les images compressées par la *DCT*.

Plus récemment la transformée en ondelettes fut utilisée dans des schémas de compression sophistiqués tels que le codage de contours multiéchelle [4], le codage par enchâssement (*Embedded Zerotree Wavelet*) [5] et dernièrement le codage par partitionnement en arbre hiérarchique (*Set Partitionning In Hierarchical Trees*) [6] qui est une extension de l'*EZW*. Nous nous intéresserons en particulier aux algoritmes *EZW* et *SPIHT* qui permettent de faire de la transmission progressive d'images, *SPIHT* permettant même d'obtenir de très bons taux de compression sans encodage. Cependant ces techniques de compression sont pour des raisons de complexité calculatoire peu implémentées dans des applications temps réel. Les recherches actuelles portent sur des architectures VLSI [7, 8, 18], ainsi que la parallélisation et l'implémentation de ces techniques sur machines parallèles [9].

On peut citer comme exemple d'utilisation de la transformée en ondelettes le schéma de compression défini par le FBI [2] pour compresser sa banque de données d'empreintes digitales. Les images d'empreintes sont compressées avec un taux de 20:1 et une image décompressée est tellement proche de l'originale que seul un expert peut distinguer de légères pertes.

L'objet des travaux de ce stage et de paralléliser et de mettre en œuvre sur machine parallèle un algorithme de compression/décompression d'images s'appuyant sur la transformée en ondelettes. La solution parallèle proposée sera évaluée et comparée par rapport à une implémentation séquentielle faite préalablement.

Ce mémoire de stage comprend sept chapitres et une annexe.

Le premier chapitre présente le schéma général des algorithmes classiques de compression/décompression s'appuyant sur la transformée en ondelettes et donc les diverses techniques sous-jacentes aux algorithmes en plus de la transformée en ondelettes, i.e. les techniques de quantification et d'encodage.

L'aspect théorique de la transformée en ondelettes fait l'objet du second chapitre en reprenant la présentation faite par Mallat dans [3].

Le troisième chapitre est consacré à la présentation des ondelettes choisies, à l'implémentation séquentielle de la transformée en ondelettes et au traitement des bords de l'image. Des images décomposées par la transformée en ondelettes sont aussi présentées.

Au cours du quatrième chapitre nous nous intéressons en particulier aux algorithmes *EZW* et *SPIHT*.

Le cinquième chapitre concerne l'aspect parallélisme, nous proposons une version parallèle de la transformée ondelettes et étudions son implémentation sur machine parallèle.

La machine parallèle retenue pour l'implantation, ainsi que le langage utilisé sont présentés dans le sixième chapitre.

Les résultats de l'expérimentation des versions séquentielle et parallèle sont exposés dans le chapitre sept.

Pour terminer l'annexe A présente un certain nombre de démonstrations.

Chapitre 1

Ondelettes et compression d'images

1.1 Introduction

Une image est un signal bidimensionnel. La compresser revient à trouver une représentation compacte équivalente, or comme les sinus et les cosinus dans l'analyse de Fourier les ondelettes permettent de représenter une image. On peut donc représenter une image en la décomposant à l'aide d'un ensemble d'ondelettes de base obtenues par l'intermédiaire d'une l'ondelette "mère" $\Psi(x)$. Une fois l'ondelette $\Psi(x)$ fixée, une base d'ondelettes peut être obtenue par translations et dilatations de $\Psi(x)$ $\{\Psi(\frac{x-b}{a}), (a,b) \in R \times R\}$. Un choix intéressant est $a = 2^{-j}$ et $b = k.2^{-j}$ avec $k, j \in Z$ car il permet d'établir un lien avec l'analyse multirésolution du traitement d'images comme on le verra dans le chapitre suivant.

L'intérêt des ondelettes par rapport aux sinus et cosinus se situe surtout à deux niveaux :

- Contrairement aux sinus et cosinus qui ne sont bien localisés qu'en fréquence les ondelettes le sont également en temps. Par conséquent tout changement de fréquence dans la transformée en ondelettes ne produira des changements que sur une certaine partie du domaine temporel.

- Les ondelettes permettent de représenter de manière compacte un grand nombre de fonctions : ainsi les fonctions formées de pics très prononcés nécessitent beaucoup moins d'ondelettes que de sinus/cosinus pour être représentées.

Les ondelettes sont utilisées dans les deux catégories de techniques de compression que sont la compression sans pertes ou réversible (*lossless*) et celle avec pertes ou irréversible (*lossy*). La première catégorie d'algorithmes permet d'avoir une image construite identique à l'originale mais avec des taux de compression peu élevés. La seconde catégorie donne des images reconstruites plus ou moins de bonne qualité suivant le taux de compression mais présente cependant l'avantage de donner accès à des taux de compression très important.

Avant de présenter le schéma général des algorithmes de compression par ondelettes nous allons introduire les différentes mesures qui sont utilisées pour évaluer les performances d'un algorithme.

1.2 Evaluation des performances d'un algorithme de compression d'images

On évalue les performances d'un algorithme de compression par le taux de compression, l'entropie de l'image reconstruite et le rapport signal sur bruit (PNSR) qui donne une mesure numérique de la qualité des images reconstruites.

Le taux de compression est défini par

$$\tau_c = \frac{\text{Le nombre de bits de l'image originale}}{\text{Le nombre de bits de l'image compressée}}$$

et l'entropie qui représente le nombre de bits moyen nécessaire pour chaque pixel par

$$E = -\sum_{i=0}^{n-1} p_i \ log_2 p_i \quad \text{(bits)}$$

n étant le nombre de niveaux de quantification ($n = 256$ pour une image à niveaux de gris codée sur 8 bits) et p_i la probabilité de présence du niveau de gris i. Plus l'entropie est faible moins on a besoin de bits pour représenter les niveaux de gris des pixels de l'image.

Le rapport signal sur bruit en décibels (dB) est défini par l'intermédiaire de l'erreur quadratique moyenne (MSE), pour une image originale f de taille $N \times M$ et $\hat{f}$ l'image reconstruite après compression on a :

$$PSNR = 20\left(log_{10}\frac{255}{MSE}\right) \quad \text{avec} \quad MSE = \sqrt{\frac{1}{NM}\sum_{i=1}^{N}\sum_{j=1}^{M}\left[f(i,j) - \hat{f}(i,j)\right]^2}$$

A noter que cette mesure n'est pas très adaptée pour évaluer les algorithmes qui sont basés sur les propriétés du système visuel humain car l'erreur quadratique moyenne ne rend pas correctement compte de la perception réelle de l'image par celui-ci.

Pour avoir une appréciation visuelle de la qualité de l'image reconstruite on utilise l'image "différence" entre l'image originale et l'image reconstruite qui est définie par

$$d(i,j) = 2(f(i,j) - \hat{f}(i,j)) + 128.$$

Une image parfaitement reconstruite est alors caractérisée par l'uniformité de l'image différence.

1.3 Schéma des algorithmes de compression d'images par ondelettes

Les algorithmes classiques de compression s'appuyant sur la transformée en ondelettes suivent le schéma de la *Figure (1.1)*. Dans un premier temps est appliquée la transformée en ondelettes pour décorréler les données de l'image, puis suit une phase de quantification des coefficients résultants et finalement les coefficients quantifiés sont encodés sans pertes. La décompression est faite en appliquant les opérations duales de la phase de compression dans l'ordre inverse.

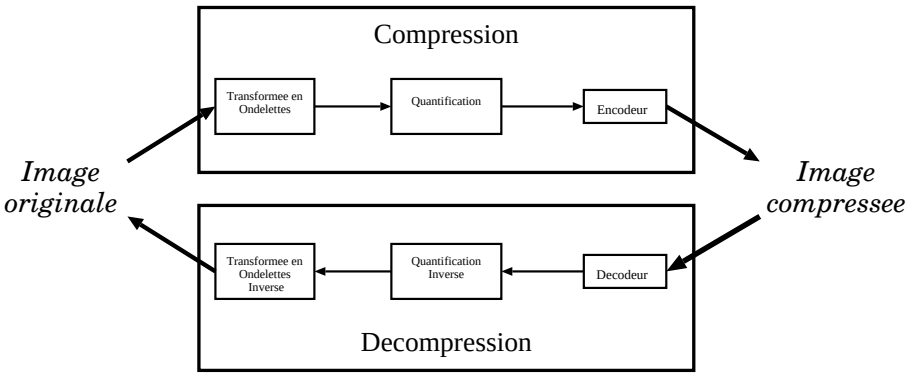


FIG. 1.1 - *Schéma général d'un algorithme de compression/décompression par ondelettes.*

La transformée en ondelettes ne fait que changer la représentation de l'image en concentrant l'information dans un nombre restreint de coefficients, la distribution des coefficients a, de plus, la forme d'un pic et donc une entropie plus faible, ce qui augmente le taux de compression sans pertes. La compression proprement dite se fait lors de la quantification et l'encodage des coefficients. D'autre part avant la quantification on peut éliminer les coefficients de faible amplitude par un seuillage sans créer de distorsion importante dans l'image reconstruite grâce à la propriété d'invariance de l'énergie contenue dans l'image originale et sa transformée.

C'est un algorithme suivant ce schéma que nous avons décidés de paralléliser.

Parmi les algorithmes de compression utilisant la transformée en ondelettes et qui ne suivent pas ce schéma il y a l'algorithme de codage de contours multiéchelle de Mallat et Zhong [4]. Cet algorithme utilise les propriétés structurelles de l'image, i.e. les arêtes et les textures ainsi que la manière dont elles sont perçues par le système visuel humain. En effet Mallat et Zhong montrent que l'évolution des maxima locaux de la transformée en ondelettes à travers les échelles caractérise la forme locale de structures irrégulières, le détecteur d'arêtes multiéchelles de Canny [10] étant équivalent à trouver le maximum local du module de la transformée en ondelettes.

1.3.1 Différentes représentations en ondelettes

Tout d'abord il y a les *ondelettes orthogonales*, celles que nous avons choisies. Elles permettent de définir une base orthogonale de l'espace des fonctions de carré intégrable de R^n. Lors de l'étude de l'aspect théorique de la transformée en ondelettes on verra qu'un algorithme de calcul rapide de la DWT a été développé par Mallat [3], et c'est cet algorithme que nous avons considérer pour implanter la transformée en ondelettes. La famille d'ondelettes orthogonales la plus intéressante est celle des ondelettes à support compact car elles induisent lors de l'implémentation des filtres à réponse impulsionnelle finie. Il y a pourtant un inconvénient de taille : les ondelettes orthogonales ne sont pas symétriques et donc introduisent des distorsions.

C'est pourquoi sont apparues *les ondelettes biorthogonales* qui, en plus des caractéristiques des ondelettes orthogonales sont symétriques. La DWT peut toujours être calculée avec l'algorithme de Mallat, mais les filtres en quadrature utilisés lors de la décomposition ne formant pas une paire orthogonale, lors de la phase de reconstruction, une autre paire de filtres orthogonale avec la première doit être utilisée.

1.3.2 Les stratégies de quantification

La quantification est une étape incontournable de l'algorithme pour réduire la quantité d'information par passage des réels aux entiers. Les deux grandes stratégies de quantification sont :

- la quantification scalaire qui associe à un réel un entier;
- la quantification vectorielle qui ne traite plus un seul réel mais un vecteur de réels, associant à un vecteur de réels un vecteur d'entiers.

Dans le cas de la quantification scalaire un quantificateur est une fonction $Q(x)$ associant a un ensemble de réels, un ensemble restreint d'entiers. En général $Q(x)$ est une fonction en escalier définie par un ensemble $\{d_i | i = 0, ..., N\}$ de points de décision et un ensemble $\{r_i | i = 0, ..., N-1\}$ de niveaux de reconstruction, N étant le nombre de niveaux de quantification. La fonction Q est alors définie par $Q : x \longmapsto r_i$ si $x \in]d_i, d_{i+1}]$. En fait l'ensemble des coefficients résultant de la transformée en ondelettes est divisé en N intervalles $]d_i, d_{i+1}]$. Le processus de quantification consiste à associer à tout réel dans l'intervalle $]d_i, d_{i+1}]$ le niveau de reconstruction r_i.

La quantification vectorielle est plus complexe à mettre en oeuvre car il faut préalablement engendrer un ensemble de vecteurs de référence appelé dictionnaire (codebook), ce qui se fait à l'aide d'un algorithme d'apprentissage que l'on applique sur un ensemble d'images. La quantification consiste alors à décomposer l'image à traiter en vecteurs de taille identique à ceux du dictionnaire, à rechercher pour chaque vecteur de l'image le plus proche dans le dictionnaire et à le remplacer par l'indice dans le dictionnaire du vecteur associé.

Il faut remarquer que la quantification vectorielle donne souvent de meilleurs résultats que la quantification scalaire.

Pourtant, dernièrement, des méthodes de quantification donnant de meilleurs résultats que les deux stratégies citées précédemment ont été introduites et notamment dans les algorithmes *EZW* de Shapiro [5] et *SPIHT* de Said et Pearlman [6]. En effet ces méthodes tirent avantage du contenu spatial de la transformée en ondelettes de l'image en quantifiant les coefficients par approximations successives à travers les sous-bandes de même orientation.

1.3.3 Les techniques d'encodage des coefficients

Les deux grandes techniques qui sont utilisées dans la plupart des algorithmes sont le *codage de Huffman* et le *codage arithmétique* [11], toutes deux reposent sur un modèle statistique de l'image à coder. La compression revenant à coder le symbole le plus probable avec moins de bits que le symbole le moins probable.

Le codeur arithmétique donne les meilleurs résultats surtout quand il utilise un modèle adaptatif, il est d'ailleurs utilisé dans les algorithmes *EZW* et *SPIHT*. Le codeur et le décodeur doivent bien entendu avoir accès au même modèle puisque c'est lui qui détermine le distribution des probabilités de présence des différents symboles. D'autre part le codeur arithmétique a l'avantage de faire clairement la distinction entre le modèle choisi pour représenter les données et le codage des informations avec ce modèle. L'implémentation proposée dans [11] est d'ailleurs faite de telle sorte que l'on puisse facilement changer de modèle.

Chapitre 2

Présentation de la théorie de la transformée en ondelettes

2.1 Introduction

De nombreux chercheurs en traitement d'images et notamment ceux travaillant à la mise au point d'algorithmes de reconnaissance de formes ont fait le constat suivant : analyser une image sur une seule échelle est inefficace. En effet les objets formant une image sont souvent de taille et de nature fréquentielle différentes, apparaissant à une échelle, disparaissant à une autre. Propos très bien illustré par Y. Meyer [12] avec la cartographie : «A des échelles différentes, les cartes contiennent des informations différentes et il est, par exemple, impossible d'organiser un circuit de visites d'églises romanes de la Charente et du Poitou à partir de l'image de la France figurant sur un globe terrestre.»

Il est clair qu'une représentation hiérarchique de l'image s'impose, consistant à structurer efficacement les données et notamment à relier le contenu spectral à la position spatiale du signal de l'image. Dans le cadre de la recherche sur ce type de représentation sont apparues les représentations multirésolutions, comprenant les techniques de décomposition en sous-bandes et les transformations pyramidales dont un panorama est donné dans [13].

Parmi les transformations pyramidales passe-bande orthogonales il y a la transformée en ondelettes. Celle-ci réorganise l'image sous forme d'une pyramide faisant apparaître l'ensemble des détails de différents niveaux de résolution. Les détails d'une image sont définis comme la différence d'information entre deux niveaux de résolution successifs. Ainsi étant donné une séquence de résolution $(r_j)_{j\in Z}$ les détails à la résolution r_j correspondent à la différence d'information entre les approximations de l'image à la résolution r_j et r_{j-1}. Approximations et détails d'une image à différentes résolutions sont obtenues par des opérations de filtrage et sous-échantillonnage successives.

Nous allons étudier l'aspect théorique en reprenant la présentation faite par S. Mallat [3] qui conduit à un algorithme pyramidal de calcul rapide de la transformée en ondelettes, le plus fréquement utilisé dans les implémentations.

2.2 Notations

– On note Z et R l'ensemble des entiers relatifs et des réels respectivement.

- $L^2(R)$ représente l'espace vectoriel des fonctions à une variable $f(x)$ mesurables et de carré intégrable.
  Dans cet espace la norme de $f(x)$ est définie par:

$$\| f \|^2 = \int_{-\infty}^{+\infty} |f(x)|^2 \ dx$$

  et sa transformée de Fourier $\hat{f}(\omega)$ par:

$$\hat{f}(\omega) = \int_{-\infty}^{+\infty} f(x) \ e^{-i\omega x} \ dx.$$

  Pour $f(x) \in L^2(R)$ et $g(x) \in L^2(R)$, le produit scalaire entre $f(x)$ et $g(x)$ s'écrit:

$$\langle g(x), f(x) \rangle = \int_{-\infty}^{+\infty} g(x) \ f(x) \ dx$$

  et leur convolution:

$$\begin{aligned} (f * g)(x) &= (f(u) * g(u))\,(x) \\ &= \int_{-\infty}^{+\infty} f(u) \ g(x-u) \ du. \end{aligned}$$

- $L^2(R^2)$ représente l'espace vectoriel des fonctions à deux variables $f(x,y)$ mesurables et de carré intégrable.
  Soit $f(x,y) \in L^2(R^2)$ et $g(x,y) \in L^2(R^2)$, le produit scalaire entre $f(x,y)$ et $g(x,y)$ est donné par:

$$\langle f(x,y), g(x,y) \rangle = \int_{-\infty}^{+\infty} f(x,y) \ g(x,y) \ dx \ dy,$$

  et la transformée de Fourier de $f(x,y)$ notée: $\hat{f}(\omega_x, \omega_y)$ est définie de la manière suivante

$$\hat{f}(\omega_x, \omega_y) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} f(x,y) \ e^{-i(\omega_x x + \omega_y y)} \ dx \ dy.$$

2.3 Transformée en ondelettes de signaux monodimensionnels

Dans cette section nous allons étudier les différentes notions introduites par la transformée en ondelettes.

Dans un premier temps cette étude ne s'intéresse qu'aux signaux monodimensionnels mesurables et d'énergie finie: $f(x) \in L^2(R)$, elle sera étendue dans la section 2.4 aux images.

2.3.1 Analyse multirésolution de $L^2(R)$

L'objet de cette section est de présenter le concept d'analyse multirésolution sous-jacent à la transformée en ondelettes tel que l'a introduit S. Mallat [3]. On se restreint à l'étude de l'analyse multirésolution dyadique, i.e. de facteur de résolution ou d'échelle égal à 2.

Soit A_{2^j} l'opérateur d'approximation du signal $f(x)$ à la résolution 2^j, on note $A_{2^j}f(x)$ le signal résultant de l'application de cet opérateur sur $f(x)$.

On commence par caractériser A_{2^j} en donnant les propriétés que l'on attend d'un tel opérateur :

1. *A_{2^j} est un opérateur de projection orthogonale, en effet c'est un opérateur de projection sur un sous-espace vectoriel particulier $V_{2^j} \in L^2(R)$ qui peut être vu comme l'espace de toutes les approximations possibles à la résolution 2^j de fonctions de $L^2(R)$, i.e*

$$A_{2^j} \circ A_{2^j} = A_{2^j} \tag{2.1}$$

De plus $A_{2^j}f(x)$ est l'approximation la plus proche de $f(x)$

$$\forall g(x) \in V_{2^j}, \quad \| g(x) - f(x) \| \quad \geq \quad \| A_{2^j} - f(x) \| \tag{2.2}$$

2. *$A_{2^{j+1}}$ contient toute l'information nécessaire au calcul du même signal à la résolution inférieure A_{2^j}, c'est le principe de causalité*

$$\forall j \in Z, \;\; V_{2^j} \subset V_{2^{j+1}} \tag{2.3}$$

d'où on en déduit

$$\forall j \in Z, \;\; f(x) \in V_{2^j} \Leftrightarrow f(2x) \in V_{2^{j+1}} \tag{2.4}$$

3. *L'approximation $A_{2^j}f(x)$ d'un signal $f(x)$ peut être caractérisée par 2^j échantillons par unité de longueur, ces échantillons correspondant aux coefficients transformés*

4. *Lorsque la résolution converge vers $+\infty$ l'approximation du signal converge vers le signal original, inversement quand la résolution converge vers $-\infty$ l'approximation du signal converge vers 0.*
Ce principe s'écrit

$$\lim_{j \to +\infty} V_{2^j} = \bigcup_{j=-\infty}^{+\infty} V_{2^j} \;\; \textit{est dense dans} \;\; L^2(R) \tag{2.5}$$

$$\lim_{j \to -\infty} V_{2^j} = \bigcap_{j=-\infty}^{+\infty} V_{2^j} = \{0\} \tag{2.6}$$

- La collection d'opérateurs A_{2^j} associée qui satisfait (2.1) et (2.2) permet d'obtenir l'approximation de toute fonction de $L^2(R)$ à la résolution 2^j.
- Une collection d'espaces vectoriels $(V_{2^j})_{j \in Z}$ satisfaisant (2.3) à (2.6) est appelée une analyse multirésolution de $L^2(R)$.

Pour caractériser numériquement l'opérateur d'approximation A_{2^j} il suffit de trouver une base orthonormale du sous-espace associé V_{2^j}.

Théorème 1 [1]

Soit $(V_{2^j})_{j\in Z}$ une analyse multirésolution de $L^2(R)$.

Il existe une unique fonction $\phi(x) \in L^2(R)$ appelée fonction d'échelle, telle que si on définit la suite de fonctions $\phi_{2^j}, j \in Z$ par $\phi_{2^j}(x) = 2^j\phi(2^j x)$ la suite $\left(\sqrt{2^{-j}}\phi_{2^j}(x - 2^{-j}n)\right)_{n\in Z}$ est une base orthonormale de V_{2^j}.

Le signal approché $A_{2^j}f(x)$ d'un signal $f(x)$ peut donc se calculer en projetant le signal $f(x)$ sur la base orthonormale de V_{2^j} définie dans le théorême précédent.

Donc $\forall f(x) \in L^2(R)$ on a

$$A_{2^j}f(x) = 2^{-j}\sum_{n=-\infty}^{+\infty}\langle f(u), \phi_{2^j}(u - 2^{-j}n)\rangle\,\phi_{2^j}(x - 2^{-j}n) \tag{2.7}$$

et la collection de produits scalaires

$$A^d_{2^j}f = \left(\langle f(u), \phi_{2^j}(u - 2^{-j}n)\rangle\right)_{n\in Z} \tag{2.8}$$

représente l'approximation discrète de $f(x)$ à la résolution 2^j, c'est-à-dire les coefficients transformés.

Mais les produits scalaires peuvent également être interprétés comme une convolution évaluée au point $2^{-j}n$, en effet

$$\begin{aligned}\langle f(u), \phi_{2^j}(u - 2^{-j}n)\rangle &= \int_{-\infty}^{+\infty} f(u)\ \ \phi_{2^j}(u - 2^{-j}n)\ \ du \\ &= (f(u) * \phi_{2^j}(-u))\,(2^{-j}n)\end{aligned}$$

on en déduit

$$A^d_{2^j}f = \left((f(u) * \phi_{2^j}(-u))\,(2^{-j}n)\right)_{n\in Z} \tag{2.9}$$

La fonction $\phi(x)$ correspond donc à un filtre passe-bas et l'approximation discrète de $f(x)$ peut être vue comme un filtrage passe-bas suivi d'un sous-échantillonnage à la cadence 2^j.

2.3.2 Formule d'implémentation du calcul du signal approché discret

En pratique on calcule les approximations successives d'un signal $f(x)$ grâce à un algorithme pyramidal comme nous allons le voir.

Soit H le filtre discret dont la réponse impulsionnelle est donnée par

$$\forall n \in Z, h(n) = \langle \phi_{2^{-1}}(u), \phi(u - n)\rangle$$

et $\tilde{H}$ le filtre miroir associé, avec $\tilde{h}(n) = h(-n)$.

Les produits scalaires de (2.7) peuvent alors s'exprimer sous la forme suivante :

$$\langle f(u), \phi_{2^j}(u - 2^{-j}n)\rangle = \sum_{k=-\infty}^{+\infty}\tilde{h}(2n - k)\,\langle f(u), \phi_{2^{j+1}}(u - 2^{-j-1}k)\rangle$$

1. Le lecteur intéressé trouvera la démonstration de ce théorême ainsi que celles des suivants dans [3].

d'où on a:

$$A_{2^j}^d f(n) = \sum_{k=-\infty}^{+\infty} \tilde{h}(2n-k) A_{2^{j+1}}^d f(k) \tag{2.10}$$

L'approximation discrète d'un signal $f(x)$ à la résolution 2^j peut donc se dériver de l'approximation discrète à la résolution 2^{j+1} par convolution avec $\tilde{H}$ et sous-échantillonnage à la cadence 2. Si le signal $A_{2^{j+1}}^d f$ est de longueur N alors le signal approché $A_{2^j}^d f$ a une longueur de $\frac{N}{2}$ suite au sous-échantillonnage.

Toute approximation $A_{2^j}^d$ avec $j < 0$ peut par conséquent être calculée à partir de A_1^d puisque l'opération d'approximation est récursive.

Le filtre H permet également d'obtenir une caractérisation de la transformée de Fourier d'une fonction d'échelle.

Théorème 2

Soit $\phi(x)$ une fonction d'échelle et H un filtre discret dont la réponse impulsionnelle est $h(n) = \langle \phi_{2^{-1}}(u), \phi(u-n) \rangle$ et ayant pour transformée de Fourier $H(\omega) = \sum_{n=-\infty}^{+\infty} h(n) e^{-in\omega}$ alors $H(\omega)$ vérifie les propriétés suivantes:

1. *$|H(0)| = 1$ et $h(n) = O(n^{-2})$ à l'infini*
2. *$|H(\omega)|^2 + |H(\omega + \pi)|^2 = 1$*

Inversement soit $H(\omega)$ une série de Fourier satisfaisant (1.), (2.) et telle que $|H(\omega)| \neq 0$ pour $\omega \in [0, \pi/2]$ alors la fonction définie par:

$$\hat{\phi}(\omega) = \prod_{p=1}^{+\infty} H(2^{-p}\omega) \tag{2.11}$$

est la transformée de Fourier de la fonction d'échelle.

Les filtres H et $\tilde{H}$ sont appelés des filtres conjugués, et grâce à (2.11) on peut à partir du filtre H déterminer la transformée de Fourier de la fonction d'échelle. On pourra donc choisir un filtre H de telle sorte que la fonction d'échelle ait de bonnes propriétés.

2.3.3 Le signal des détails

Dans cette sous-section nous allons nous intéresser à la représentation de la différence d'information entre deux niveaux de résolutions successifs 2^{j+1} et 2^j, différence que l'on appelle précisément *signal des détails* à la résolution 2^j.

Nous allons voir qu'une telle représentation peut être obtenue en décomposant le signal à l'aide d'une base orthonormée d'ondelettes.

Considérons l'analyse multirésolution d'un signal $f(x)$ de $L^2(R)$ sur une séquence dyadique de résolutions $(2^j)_{j \in Z}$; alors les approximations du signal aux résolutions 2^{j+1} et 2^j sont respectivement égale à la projection orthogonale de $f(x)$ sur $V_{2^{j+1}}$ et V_{2^j}.

Puisque $V_{2^j} \subset V_{2^{j+1}}$ il est clair que le signal des détails à la résolution 2^j est donné par la projection orthogonale du signal original sur le complémentaire orthogonal O_{2^j} de V_{2^j} dans $V_{2^{j+1}}$

$$V_{2^{j+1}} = V_{2^j} \oplus O_{2^j} \quad \text{et} \quad V_{2^j} \perp O_{2^j}$$

Pour calculer la projection orthogonale du signal $f(x)$ sur O_{2^j} on a besoin d'une base orthonormée du sous-espace vectoriel O_{2^j}.

Théorème 3

Soit $(V_{2^j})_{j\in Z}$ une séquence de sous-espaces vectoriels formant une analyse multirésolution, $\phi(x)$ la fonction d'échelle et H le filtre conjugué correspondant.

Soit $\psi(x)$ une fonction dont la transformée de Fourier est donnée par

$$\begin{aligned} \hat{\psi}(\omega) &= G(\omega/2)\hat{\phi}(\omega/2) \\ avec \quad G(\omega) &= e^{-i\omega}\overline{H(\omega+\pi)} \Leftrightarrow g(n) = (-1)^{(1-n)}h(1-n) \end{aligned} \tag{2.12}$$

Soit $\psi_{2^j}(x) = 2^j\psi(2^j x)$ la dilatation de $\psi(x)$ par 2^j, alors
$\left(\sqrt{2^{-j}}\psi_{2^j}(x-2^{-j}n)\right)_{n\in Z}$ est une base orthonormale de O_{2^j}
et $\left(\sqrt{2^{-j}}\psi_{2^j}(x-2^{-j}n)\right)_{(n,j)\in Z^2}$ est une base orthonormale de $L^2(R)$.

$\psi(x)$ est appelée une ondelette orthogonale.

On peut donc obtenir une ondelette $\psi(x)$ par (2.12) en définissant une fonction $H(\omega)$ vérifiant le **Théorème 2**, et la fonction d'échelle $\phi(x)$ correspondante grâce à (2.11). Il est à noter qu'en traitement du signal les filtres H et G sont appelés des filtres *QMF (Quadrature Mirror Filters)* [16].

En étudiant l'influence du choix de $H(\omega)$ sur les propriétés de $\phi(x)$ et $\psi(x)$, Daubechies [14] a montré que $\forall n > 0$ on peut trouver une fonction $H(\omega)$ telle que l'ondelette correspondante $\psi(x)$ ait un support fini et soit n fois continuement dérivable. Cette étude aboutit à la définition d'une classe d'ondelettes que l'on appelle *ondelettes à support compact de Daubechies.*

La projection du signal des détails dans la base introduite par le **Théorème 3** permet d'aboutir à une formule du même type que (2.7) et donc de caractériser la projection par la suite des produits scalaires :

$$D_{2^j}f = \left(\langle f(u), \psi_{2^j}(u-2^{-j}n)\rangle\right)_{n\in Z} \tag{2.13}$$

Cette suite $D_{2^j}f$ de coefficients d'ondelettes, appelée le signal des détails discret à la résolution 2^j contient les différences d'information entre $A^d_{2^{j+1}}f$ et $A^d_{2^j}f$.

En utilisant la même démarche que celle ayant permis d'aboutir à (2.9) on obtient :

$$D_{2^j}f = \left((f(u) * \psi_{2^j}(-u))\,(2^{-j}n)\right)_{n\in Z} \tag{2.14}$$

ce qui signifie que l'ondelette $\psi(x)$ peut être vue comme un filtre passe-bande et donc le signal des détails discret correspond à un filtrage passe-bande suivi d'un sous-échantillonnage à la cadence 2^j.

2.3.4 Formule d'implémentation du calcul du signal des détails discret

Comme pour le signal approché, pour calculer les signaux des détails successifs on utilise un algorithme pyramidal dont le principe est identique à celui de la section 2.3.2. On va montrer que, comme pour $A^d_{2^j}f$, $D_{2^j}f$ qui peut se calculer en convoluant $A^d_{2^{j+1}}f$ avec un filtre discret G.

Soit G le filtre discret dont la réponse impulsionnelle est donnée par

$$\forall n \in Z, g(n) = \langle \psi_{2^{-1}}(u), \psi(u-n) \rangle$$

et $\tilde{G}$ le filtre miroir associé, avec $\tilde{g}(n) = g(-n)$.

Ceci permet d'écrire les produits scalaires de (2.13) sous la forme

$$\langle f(u), \psi_{2^j}(u - 2^{-j}n) \rangle = \sum_{k=-\infty}^{+\infty} \tilde{g}(2n-k) \langle f(u), \phi_{2^{j+1}}(u - 2^{-j-1}k) \rangle$$

d'où on a

$$D_{2^j} f(n) = \sum_{k=-\infty}^{+\infty} \tilde{g}(2n-k) A^d_{2^{j+1}} f(k) \tag{2.15}$$

De même que le signal approché $A^d_{2^j} f$, le signal des détails $D_{2^j} f$ est de longueur $\frac{N}{2}$ si le signal à la résolution $j+1$ est de longueur N.

2.3.5 La représentation par ondelettes orthogonales d'un signal

Tout signal discret original $A^d_1 f$ mesuré à la résolution 0 peut être représenté pour tout nombre de niveaux de décomposition $J > 0$, par

$$\left(A^d_{2^{-J}} f, (D_{2^j} f)_{-J \leq j \leq -1} \right)$$

qui a une longueur identique au signal original.

Cet ensemble de signaux discrets se compose d'une approximation du signal à une résolution grossière $A^d_{2^{-J}} f$ et d'une séquence de signaux des détails aux résolutions 2^j pour $-J \leq j \leq -1$. On peut voir cet ensemble comme une décomposition du signal dans une base orthonormale d'ondelettes ou encore une décomposition dans un ensemble de canaux fréquentiels.

Il est à remarquer que le qualificatif d'orthogonale donné à la représentation est dû au fait que les ondelettes sont orthogonales.

D'après (2.10) et (2.15) on peut voir que la représentation par ondelettes orthogonales d'un signal discret $A^d_1 f$ se calcule par décomposition successive de $A^d_{2^{j+1}} f$ en $A^d_{2^j} f$ et $D_{2^j} f$ avec $j \in [-J, -1]$. Il faut noter cependant que les bords des images posent problème comme on le verra dans le chapitre suivant.

2.3.6 Reconstruction du signal

La représentation en ondelettes est complète car à partir du signal approché et du signal des détails à la résolution 2^j on peut reconstruire parfaitement le signal à la résolution 2^{j+1} qui a été décomposé. On verra que, comme la phase de décomposition, la reconstruction peut s'implémenter par une transformation pyramidale.

Le but est de pouvoir calculer $A^d_{2^{j+1}} f$ l'approximation du signal à la résolution 2^{j+1} à partir du signal approché $A^d_{2^j} f$ et du signal des détails $D_{2^j} f$ à la résolution 2^j.

On a vu à la section 2.3.3 que $V_{2^{j+1}} = V_{2^j} \oplus O_{2^j}$, donc d'après les **Théorèmes 1** et **3** on a $\left(\sqrt{2^{-j}} \phi_{2^j}(x - 2^{-j}n), \sqrt{2^{-j}} \psi_{2^j}(x - 2^{-j}n) \right)_{n \in Z}$ qui est une base orthonormale de $V_{2^{j+1}}$.

Pour tout $n > 0$ on peut donc décomposer $\phi_{2^{j+1}}(x - 2^{-j-1}n)$ dans cette base, le produit scalaire $(\langle f(u), \phi_{2^{j+1}}(u - 2^{-j-1}n)\rangle)$ qui permet de définir $A_{2^{j+1}}f$ d'après (2.8) peut alors être calculé de la manière suivante:

$$\begin{aligned}
&\langle f(u), \phi_{2^{j+1}}(u - 2^{-j-1}n)\rangle = \\
&\quad \left(2^{-j}\sum_{k=-\infty}^{+\infty} \langle \phi_{2^j}(u - 2^{-j}k), \phi_{2^{j+1}}(u - 2^{-j-1}n)\rangle \langle f(u), \phi_{2^j}(u - 2^{-j}k)\rangle\right) \\
&\qquad\qquad + \\
&\quad \left(2^{-j}\sum_{k=-\infty}^{+\infty} \langle \psi_{2^j}(u - 2^{-j}k), \phi_{2^{j+1}}(u - 2^{-j-1}n)\rangle \langle f(u), \psi_{2^j}(u - 2^{-j}k)\rangle\right)
\end{aligned}$$

Finalement en utilisant les filtres H et G, ainsi que (2.8) et (2.13) l'expression précédente conduit à la formule d'implémentation

$$A^d_{2^{j+1}}f(n) = 2\sum_{k=-\infty}^{+\infty} h(n-2k)A^d_{2^j}f(k) + 2\sum_{k=-\infty}^{+\infty} g(n-2k)D_{2^j}f(k) \tag{2.16}$$

$A^d_{2^{j+1}}f$ peut donc être reconstruit en additionnant l'interpolation de $A^d_{2^j}f$ et $D_{2^j}f$ par H et G respectivement.

Tout signal signal discret original $A^d_1 f$ transformé en ondelettes sous la forme $\left(A^d_{2^{-J}}f, (D_{2^j}f)_{-J\le j\le -1}\right)$ en le décomposant sur J niveaux pourra être reconstruit récursivement à l'aide la formule précédente pour $-J \le j \le -1$.

2.4 Extension de la transformée en ondelettes aux signaux bidimensionnels, i.e. les images

Cette section étend les notions introduites dans 2.3 pour aboutir à l'algorithme pyramidal de Mallat qui est utilisé dans les applications de traitement d'images.

Les signaux bidimensionnels sont supposés mesurables et d'énergie finie: $f(x,y) \in L^2(R^2)$.

L'analyse multirésolution de $L^2(R^2)$ est obtenue en la définissant comme une suite de sous-espaces vectoriels $(V^2_{2^j})$ de $L^2(R^2)$ qui satisfont une simple extension des propriétés (2.3) à (2.6).

L'approximation du signal $f(x,y)$ à la résolution 2^j s'obtient toujours en projetant orthogonalement $f(x,y)$ sur le sous-espace $V^2_{2^j}$.

Le **Théorème 1** reste également valide, i.e. que si $\Phi(x,y) \in L^2(R^2)$ est la fonction d'échelle alors

$$(2^{-j}\Phi_{2^j}(x - 2^{-j}n, y - 2^{-j}m))_{(n,m)\in Z^2} \quad \text{où} \quad \Phi_{2^j}(x,y) = 2^{2j}\Phi(2^j x, 2^j y) \tag{2.17}$$

forme une base orthonormale de $V^2_{2^j}$.

La construction de l'analyse multirésolution $(V^2_{2^j})_{j\in Z}$ de $L^2(R^2)$ peut se faire en particulier par produit tensoriel d'une analyse multirésolution $(V^1_{2^j})_{j\in Z}$ de $L^2(R)$: $V^2_{2^j} = V^1_{2^j} \otimes V^1_{2^j}$; la fonction d'échelle $\Phi(x,y)$ est alors également définissable sous forme d'un produit $\Phi(x,y) = \phi(x)\phi(y)$ où $\phi(x)$ est la fonction d'échelle de $(V^1_{2^j})_{j\in Z}$. L'analyse multirésolution de $L^2(R^2)$ est alors dite séparable et elle a la particularité de mettre en évidence les orientations de direction verticale, horizontale et diagonale. Il est à noter également que la séparabilité adoptée par Mallat permet d'avoir un calcul plus rapide.

Le signal approximé discret de $f(x,y)$ à la résolution 2^j s'obtient comme dans le cas monodimensionnel par le produit scalaire de $f(x,y)$ avec l'ensemble des vecteurs de la base de $V^2_{2^j}$ définie par (2.17)

$$A^d_{2^j} f = \left(\langle f(x,y), \Phi_{2^j}(x - 2^{-j}n, y - 2^{-j}m)\rangle\right)_{(n,m)\in Z^2} \tag{2.18}$$

De même le signal des détails à la résolution 2^j s'obtient par projection orthogonale du signal sur le complémentaire orthogonal $O^2_{2^j}$ de $V^2_{2^j}$ dans $V^2_{2^{j+1}}$, complémentaire défini par

$$O^2_{2^j} = (V^1_{2^j} \otimes O^1_{2^j}) \oplus (O^1_{2^j} \otimes V^1_{2^j}) \oplus (O^1_{2^j} \otimes O^1_{2^j})$$

où $O^1_{2^j}$ est le complémentaire orthogonal de $V^1_{2^j}$.

L'extension du **Théorème 3** au cas bidimensionnel conduit au théorème ci-dessous qui définit une base de $O^2_{2^j}$.

Théorème 4

Soit $(V^2_{2^j})_{j\in Z}$ une analyse multirésolution séparable de $L^2(R^2)$.
Soit $\Phi(x,y) = \phi(x)\phi(y)$ la fonction d'échelle bidimensionnelle associée.
Soit $\psi(x)$ l'ondelette associée à $\phi(x)$.
On peut alors construire trois ondelettes
$\Psi^1(x,y) = \phi(x)\psi(y) \quad \Psi^2(x,y) = \psi(x)\phi(y) \quad \Psi^3(x,y) = \psi(x)\psi(y)$
telles que avec $\Psi_{2^j}(x,y) = h_{2^j}(x)g_{2^j}(y)$ pour $\Psi(x,y) = h(x)g(y)$, avec $j \in Z$ on ait

$$\begin{array}{l}(\Psi^1_{2^j}(x - 2^{-j}n, y - 2^{-j}m),\\ \Psi^2_{2^j}(x - 2^{-j}n, y - 2^{-j}m),\\ \Psi^3_{2^j}(x - 2^{-j}n, y - 2^{-j}m))_{(n,m)\in Z^2}\end{array}$$

qui soit une base orthonormale de $O^2_{2^j}$, et une base orthonormale de $L^2(R^2)$ pour $(n,m,j) \in Z^3$.

Le signal des détails discret de $f(x,y)$ à la résolution 2^j est alors caractérisé par le produit scalaire de $f(x,y)$ avec l'ensemble des vecteurs de la base définie par le théorème précédent. En particulier si on considère $\Psi^1_{2^j}, \Psi^2_{2^j}$ et $\Psi^3_{2^j}$ séparément on voit que la différence d'information entre $A^d_{2^{j+1}} f$ et $A^d_{2^j} f$ est donnée par trois images de détails :

$$D^1_{2^j} f = \left(\langle f(x,y), \Psi^1_{2^j}(x - 2^{-j}n, y - 2^{-j}m)\rangle\right)_{(n,m)\in Z^2} \tag{2.19}$$

$$D^2_{2^j} f = \left(\langle f(x,y), \Psi^2_{2^j}(x - 2^{-j}n, y - 2^{-j}m)\rangle\right)_{(n,m)\in Z^2} \tag{2.20}$$

$$D^3_{2^j} f = \left(\langle f(x,y), \Psi^3_{2^j}(x - 2^{-j}n, y - 2^{-j}m)\rangle\right)_{(n,m)\in Z^2} \tag{2.21}$$

qui mettent en évidence les détails dans les directions horizontale, verticale et diagonale respectivement.

En utilisant la même méthode que celle ayant permis de passer de (2.8) à (2.9) on peut voir que les formules (2.18) à (2.21) sont équivalentes à un produit de convolution suivi d'un sous-échantillonnage

$$\begin{aligned}
A^d_{2^j} f &= \left(f(x,y) * \phi_{2^j}(-x)\phi_{2^j}(-y)\right)(2^{-j}n, 2^{-j}m)_{(n,m)\in Z^2}\\
D^1_{2^j} f &= \left(f(x,y) * \phi_{2^j}(-x)\psi_{2^j}(-y)\right)(2^{-j}n, 2^{-j}m)_{(n,m)\in Z^2}\\
D^2_{2^j} f &= \left(f(x,y) * \psi_{2^j}(-x)\phi_{2^j}(-y)\right)(2^{-j}n, 2^{-j}m)_{(n,m)\in Z^2}\\
D^3_{2^j} f &= \left(f(x,y) * \psi_{2^j}(-x)\psi_{2^j}(-y)\right)(2^{-j}n, 2^{-j}m)_{(n,m)\in Z^2}
\end{aligned}$$

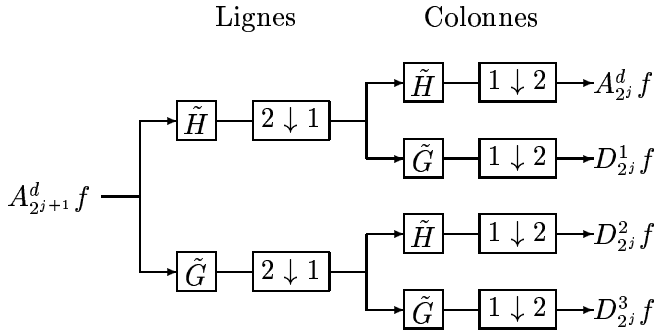


FIG. 2.1 - *Schéma de décomposition de $A^d_{2^{j+1}} f$ en $A^d_{2^j} f, D^1_{2^j} f, D^2_{2^j} f$ et $D^3_{2^j} f$.*

Il est clair que le signal approché et les différents signaux des détails à la résolution 2^j en dimension 2 se calculent par filtrages séparables du signal à la résolution 2^{j+1} suivant les deux axes. De plus leur taille est de $\frac{N}{2} \times \frac{N}{2}$ chacun si le signal $A^d_{2^{j+1}} f$ est de taille $N \times N$, suite au sous-échantillonnage.

L'introduction des filtres $\tilde{H}$ et $\tilde{G}$ permet d'aboutir à partir des produits scalaires (2.18) à (2.21) aux formules d'implémentation de l'analyse d'une image

$$A^d_{2^j} f(n,m) = \sum_{k=-\infty}^{+\infty} \sum_{l=-\infty}^{+\infty} \tilde{h}(2n-k)\tilde{h}(2m-l) A^d_{2^{j+1}} f(k,l) \tag{2.22}$$

$$D^1_{2^j} f(n,m) = \sum_{k=-\infty}^{+\infty} \sum_{l=-\infty}^{+\infty} \tilde{h}(2n-k)\tilde{g}(2m-l) A^d_{2^{j+1}} f(k,l) \tag{2.23}$$

$$D^2_{2^j} f(n,m) = \sum_{k=-\infty}^{+\infty} \sum_{l=-\infty}^{+\infty} \tilde{g}(2n-k)\tilde{h}(2m-l) A^d_{2^{j+1}} f(k,l) \tag{2.24}$$

$$D^3_{2^j} f(n,m) = \sum_{k=-\infty}^{+\infty} \sum_{l=-\infty}^{+\infty} \tilde{g}(2n-k)\tilde{g}(2m-l) A^d_{2^{j+1}} f(k,l) \tag{2.25}$$

La *Figure (2.1)* représente le schéma d'analyse de l'algorithme pyramidal de Mallat, $\tilde{H}$ et $\tilde{G}$ correspondant aux filtres miroirs de H et G respectivement.

Toute image $A^d_1 f$ est donc équivalente à $(A^d_{2^{-J}} f, (D^1_{2^j} f, D^2_{2^j} f, D^3_{2^j} f)_{-J \leq j \leq -1})$ qui correspond à sa représentation en ondelettes sur J niveaux d'analyse (décomposition) pour tout $J > 0$. Du fait de l'orthogonalité de la décomposition le nombre de coefficients résultant de la transformée en ondelettes est constant et égal à la taille de l'image originale.

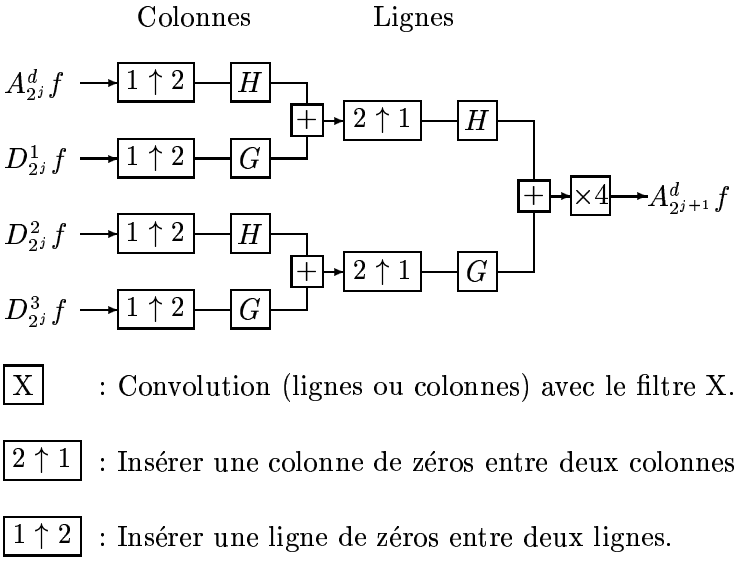


FIG. 2.2 - *Schéma de reconstruction de $A^d_{2^{j+1}} f$ à partir de l'approximation $A^d_{2^j} f$ et des signaux des détails $D^1_{2^j} f, D^2_{2^j} f$ et $D^3_{2^j} f$.*

Comme dans le cas monodimensionnel de la section 2.3.6, la reconstruction du signal se fait récursivement. On additionne l'interpolation du signal approché $A^d_{2^j} f$ à la résolution 2^j avec l'interpolation de chaque signal des détails $D^1_{2^j} f, D^2_{2^j} f$ et $D^3_{2^j} f$ à la même résolution pour obtenir le signal approché a résolution 2^{j+1}. Ce processus est ensuite répété jusqu'à arriver à $A^d_1 f$ le signal discret original.

Les filtres interpolateurs sont H et G, et on a la formule de synthèse suivante :

$$A^d_{2^{j+1}} f(n,m) = IA^d_{2^j} f(n,m) + ID^1_{2^j} f(n,m) + ID^2_{2^j} f(n,m) + ID^3_{2^j} f(n,m)$$

avec les formules d'interpolation :

$$IA^d_{2^j} f(n,m) = 4 \sum_{k=-\infty}^{+\infty} \sum_{l=-\infty}^{+\infty} h(n-2k)h(m-2l)A^d_{2^j} f(k,l) \tag{2.26}$$

$$ID^1_{2^j} f(n,m) = 4 \sum_{k=-\infty}^{+\infty} \sum_{l=-\infty}^{+\infty} h(n-2k)g(m-2l)D^1_{2^j} f(k,l) \tag{2.27}$$

$$ID^2_{2^j} f(n,m) = 4 \sum_{k=-\infty}^{+\infty} \sum_{l=-\infty}^{+\infty} g(n-2k)h(m-2l)D^2_{2^j} f(k,l) \tag{2.28}$$

$$ID^3_{2^j} f(n,m) = 4 \sum_{k=-\infty}^{+\infty} \sum_{l=-\infty}^{+\infty} g(n-2k)g(m-2l)D^3_{2^j} f(k,l) \tag{2.29}$$

La *Figure (2.2)* présente le schéma de synthèse de l'algorithme pyramidal de Mallat et l'annexe A.1 explique comment sont obtenues les formules implémentant l'analyse et l'interpolation.

Chapitre 3

Choix des ondelettes et implémentation séquentielle

3.1 Choix des filtres et propriétés

Nous avons vu dans le chapitre précédent que l'implémentation de la transformée en ondelettes se ramenait à une suite d'opérations de filtrages, sous-échantillonnages dans le cas de l'analyse et de filtrages, interpolations dans la cas de la synthèse.

Puisque les filtres sont dérivés à partir d'une base orthogonale d'ondelettes, nous allons dans un premier temps étudier les critères de choix de l'ondelette "mère" à partir de laquelle on définit la base. Nous présentons ensuite les ondelettes que nous avons choisies, puis nous donnons les propriétés des filtres *CQF* [17] qui sont dérivés de ces ondelettes.

3.1.1 Critères de choix d'une ondelette

De nombreux critères peuvent intervenir dans le choix d'une ondelette [15] tels que l'orthogonalité, son support, etc. Cependant en ce qui concerne la propriété d'orthogonalité, celle-ci est implicite puisque la projection du signal discret de l'image pour obtenir les signaux des détails ne s'est fait que sur des bases d'ondelettes orthogonales. Ce critère est rappelé car il existe des ondelettes non orthogonales.

Nous allons en particulier nous intéresser aux trois critères les plus significatifs dans le choix d'une ondelette : *support compact, oscillation et régularité.*

- *Support compact*

  Une fonction d'échelle et une ondelette à support compact permettent d'avoir des filtres H et G qui sont à réponse impulsionnelle finie. Donc d'avoir des sommes "finies" au niveau de l'implémentation avec l'algorithme pyramidal de calcul rapide de la transformée en ondelettes.

- *Oscillation*

  Ce critère est lié au nombre de moments nuls de l'ondelette, en effet le nombre de moments nuls conditionne le degré d'oscillation de l'ondelette. Plus le nombre de moments nuls est grand, plus on peut mettre de coefficients à zéro sans nuire à la qualité de l'image reconstruite. Cependant il faut faire attention à la taille des filtres choisis, en effet celle-ci

est proportionnelle au nombre de moments nuls de l'ondelette et la complexité en temps de calcul de l'algorithme augmente avec la taille des filtres.

- *Régularité*
  La régularité d'une ondelette est évaluée à partir de l'exposant de Hölder de l'ondelette ou de la fonction d'échelle puisque ces deux fonctions ont le même exposant. Ce critère est très important dans la phase de synthèse/reconstruction car un manque de régularité de l'ondelette fait apparaître des effets de blocs. A l'opposé un degré de régularité élevé aboutit à des effets de bords suite à un lissage trop important.

Il faut remarquer que l'aspect phase linéaire des filtres caractérisé par la symétrie des coefficients est important en traitement d'images. La recherche de filtres à phase linéaire ayant conduit au développement des ondelettes biorthogonales qui sont à la fois à support compact et symétriques, ce qui n'est pas le cas des *ondelettes à support compact de Daubechies.*

3.1.2 Ondelettes choisies

Plutôt que d'implémenter la transformée avec *l'ondelette de Lemarie-Battle* que Mallat a utilisé, nous avons choisi *l'ondelette de Haar* et *les ondelettes orthonormales à support compact de Daubechies* [14]. Il est à noter que *l'ondelette de Haar* est également une ondelette à support compact.

1. **L'ondelette de Haar**
   Découverte par A. Haar en 1909, cette ondelette est la plus simple et la seule à support compact dérivant des filtres à phase linéaire comme l'a montré Daubechies. Cependant cette ondelette n'est pas régulière, elle effectue donc un simple moyennage de pixels voisins d'où des effets de blocs.

2. **Les ondelettes à support compact de Daubechies**
   Ces ondelettes ont été construites par I. Daubechies dans le cadre de l'étude de la construction d'une analyse multirésolution de $L^2(R)$ à partir des coefficients d'un filtre miroir en quadrature. En effet elle remarqua que l'approche fonctionnelle de l'analyse multirésolution permet la construction d'algorithmes discrets à partir d'ondelettes, d'où l'étude pour savoir si l'inverse est possible. Le théorème fondamental auquel I. Daubechies aboutit donne les principales caractéristiques des filtres dérivés des ondelettes à support compact.

3.1.3 Propriétés des filtres CQF dérivés des ondelettes à support compact

- H est un filtre passe-bas de réponse impulsionnelle finie $h(n)$ à p coefficients, vérifiant

$$\sum_{n=0}^{p-1} h(n) = \sqrt{2}$$

$$\sum_{n=0}^{p-1} h(n-2k)h(n-2l) = \delta_{kl}$$

– G est un filtre passe-bas de réponse impulsionnelle finie $g(n)$ à p coefficients, vérifiant

$$\sum_{n=0}^{p-1} g(n) = 0$$

$$g(n) = (-1)^n h(1-n)$$

3.1.4 Les nouvelles formules d'implémentation

La réponse impulsionnelle du filtre H (CQF), $h(n)$ est définie par

$$\forall n \in Z, h(n) = \left\langle \phi(u), \sqrt{2}\phi(2u-n) \right\rangle$$

ce qui est différent de la définition de Mallat.

Par conséquent les formules d'analyse et la formule de synthèse sont différentes : en fait elles ne varient que d'un facteur multiplicatif. En effet entre la réponse impulsionnelle de H définie par Mallat h' et celle ci-dessus il y a la relation[1] $h'(n) = \frac{1}{\sqrt{2}} h(n)$.

On obtient alors à partir des formules d'implémentation du chapitre précédent les formules ci-dessous, où h et g sont des filtres CQF qui vérifient les propriétés ci-dessus. Avec $\tilde{h}$, $\tilde{g}$ leur filtres miroir respectif vérifiant toujours les relations $\tilde{h} = h(-n)$ et $\tilde{g} = g(-n)$.

Les formules d'analyse (décomposition)

$$A^d_{2^j} f(n,m) = \frac{1}{2} \sum_{k=-\infty}^{+\infty} \sum_{l=-\infty}^{+\infty} \tilde{h}(2n-k)\tilde{h}(2m-l) A^d_{2^{j+1}} f(k,l) \tag{3.1}$$

$$D^1_{2^j} f(n,m) = \frac{1}{2} \sum_{k=-\infty}^{+\infty} \sum_{l=-\infty}^{+\infty} \tilde{h}(2n-k)\tilde{g}(2m-l) A^d_{2^{j+1}} f(k,l) \tag{3.2}$$

$$D^2_{2^j} f(n,m) = \frac{1}{2} \sum_{k=-\infty}^{+\infty} \sum_{l=-\infty}^{+\infty} \tilde{g}(2n-k)\tilde{h}(2m-l) A^d_{2^{j+1}} f(k,l) \tag{3.3}$$

$$D^3_{2^j} f(n,m) = \frac{1}{2} \sum_{k=-\infty}^{+\infty} \sum_{l=-\infty}^{+\infty} \tilde{g}(2n-k)\tilde{g}(2m-l) A^d_{2^{j+1}} f(k,l) \tag{3.4}$$

La formule de synthèse (reconstruction)

On a la formule :

$A^d_{2^{j+1}} f(n,m) = IA^d_{2^j} f(n,m) + ID^1_{2^j} f(n,m) + ID^2_{2^j} f(n,m) + ID^3_{2^j} f(n,m)$

avec

$$IA^d_{2^j} f(n,m) = 2 \sum_{k=-\infty}^{+\infty} \sum_{l=-\infty}^{+\infty} h(n-2k)h(m-2l) A^d_{2^j} f(k,l) \tag{3.5}$$

$$ID^1_{2^j} f(n,m) = 2 \sum_{k=-\infty}^{+\infty} \sum_{l=-\infty}^{+\infty} h(n-2k)g(m-2l) D^1_{2^j} f(k,l) \tag{3.6}$$

1. Démonstration donnée en annexe.

$$ID^2_{2^j} f(n,m) = 2 \sum_{k=-\infty}^{+\infty} \sum_{l=-\infty}^{+\infty} g(n-2k)h(m-2l)D^2_{2^j} f(k,l) \tag{3.7}$$

$$ID^3_{2^j} f(n,m) = 2 \sum_{k=-\infty}^{+\infty} \sum_{l=-\infty}^{+\infty} g(n-2k)g(m-2l)D^3_{2^j} f(k,l) \tag{3.8}$$

3.2 Modification des formules d'analyse et d'interpolation pour avoir une implémentation efficace

Les formules d'analyse et d'interpolation que l'on a implantées ont été obtenues en appliquant la méthode proposée par Chuang et Chen [18].

L'objectif est de supprimer des formules (3.1) à (3.8) les bornes infinies au niveau des différentes sommes.

On supposera que les filtres h et g sont de longueur finie p.

Les différents facteurs $\frac{1}{2}$ et 2 seront supprimés car ils s'annulent.

3.2.1 Obtention des différentes formules d'analyse

Nous allons commencer par donner une explication approfondie de la méthode en l'appliquant à la formule donnant l'approximation de l'image à la résolution 2^j à partir de son approximation à la résolution 2^{j+1}.

Commençons par remplaçer $\tilde{h}$ par h en sachant que $\tilde{h}(n) = h(-n)$, on obtient alors

$$A^d_{2^j} f(n,m) = \sum_{k=-\infty}^{+\infty} \sum_{l=-\infty}^{+\infty} h(k-2n)h(l-2m)A^d_{2^{j+1}} f(k,l) \tag{3.9}$$

D'autre part le filtre h ayant une longueur fixe p, on notera ses coefficients $h(0), h(1), ..., h(p-1)$, on a donc

$$\begin{aligned} k - 2n \in [0, p-1] &\Leftrightarrow k \in [2n, 2n+p-1] \\ l - 2m \in [0, p-1] &\Leftrightarrow l \in [2m, 2m+p-1] \end{aligned}$$

Les bornes supérieures et inférieures des deux sommes dans (3.9) peuvent donc être modifiées pour obtenir l'expression suivante

$$A^d_{2^j} f(n,m) = \sum_{k=2n}^{2n+p-1} \sum_{l=2m}^{2m+p-1} h(k-2n)h(l-2m)A^d_{2^{j+1}} f(k,l) \tag{3.10}$$

Posons $K = k - 2n$ et $L = l - 2m$ on a alors $k = K + 2n$ et $l = L + 2m$, introduisons ensuite K et L dans (3.10) de telle manière que k et l disparaissent. L'expression (3.10) devient alors finalement

$$A^d_{2^j} f(n,m) = \sum_{K=0}^{p-1} \sum_{L=0}^{p-1} h(K)h(L)A^d_{2^{j+1}} f(K+2n, L+2m)$$

En appliquant la même méthode aux trois formules donnant les détails, en sachant que $g(n) = (-1)^n h(1-n)$ et donc que les coefficients de g seront notés $g(2-p), ..., g(1)$ on aboutit aux formules suivantes

$$\begin{aligned} D^1_{2^j} f(n,m) &= \sum_{K=0}^{p-1} \sum_{L=2-p}^{1} h(K) g(L) A^d_{2^{j+1}} f(K+2n, L+2m) \\ D^2_{2^j} f(n,m) &= \sum_{K=2-p}^{1} \sum_{L=0}^{p-1} g(K) h(L) A^d_{2^{j+1}} f(K+2n, L+2m) \\ D^3_{2^j} f(n,m) &= \sum_{K=2-p}^{1} \sum_{L=2-p}^{1} g(K) g(L) A^d_{2^{j+1}} f(K+2n, L+2m) \end{aligned}$$

Mais en fait on n'effectue pas les convolutions et sous-échantillonnages à la cadence 2 sur les lignes et les colonnes en une seule étape. En effet conformément au schéma correspondant à la phase d'analyse de la *Figure (2.1)* on éclate le processus en deux étapes:

1. Convolution et sous-échantillonnage à la cadence 2 sur les lignes;
2. Convolution et sous-échantillonnage à la cadence 2 sur les colonnes.

En effet $A^d_{2^j}$ et $D^1_{2^j}$ ainsi que $D^2_{2^j}$ et $D^3_{2^j}$ effectuent les mêmes opérations sur les lignes: filtrage par $\tilde{h}$ et par $\tilde{g}$ respectivement.

On introduit $L^1_{2^j}$ et $L^2_{2^j}$ qui correspondent au résultat de la convolution et décimation à la cadence 2 des lignes de $A^d_{2^{j+1}}$ avec les filtres $\tilde{h}$ et $\tilde{g}$ respectivement.

D'où

- $$L^1_{2^j} f(n,m) = \sum_{K=0}^{p-1} h(K) A^d_{2^{j+1}} f(K+2n, m)$$

  et

  $$\begin{aligned} A^d_{2^j} f(n,m) &= \sum_{L=0}^{p-1} h(L) L^1_{2^j} f(n, L+2m) \\ D^1_{2^j} f(n,m) &= \sum_{L=2-p}^{1} g(L) L^1_{2^j} f(n, L+2m) \end{aligned}$$

- $$L^2_{2^j} f(n,m) = \sum_{K=2-p}^{1} g(K) A^d_{2^{j+1}} f(K+2n, m)$$

  et

  $$\begin{aligned} D^2_{2^j} f(n,m) &= \sum_{L=0}^{p-1} h(L) L^2_{2^j} f(n, L+2m) \\ D^3_{2^j} f(n,m) &= \sum_{L=2-p}^{1} g(L) L^2_{2^j} f(n, L+2m) \end{aligned}$$

3.2.2 Obtention des différentes formules d'interpolation

Nous reprenons le même schéma que précédemment, on commence donc par donner le détail des manipulations qui ont permis d'aboutir à la formule interpolant l'image approchée de résolution 2^j qui a été implémentée.

On rappelle que la formule interpolant l'image approchée est la suivante

$$IA^d_{2^j} f(n,m) = \sum_{k=-\infty}^{+\infty} \sum_{l=-\infty}^{+\infty} h(n-2k)h(m-2l) A^d_{2^j} f(k,l) \tag{3.11}$$

De plus le filtre h a une longueur finie p et donc ses coefficients seront notés $h(0), h(1), ..., h(p-1)$, d'où

$$\begin{aligned} n-2k \in [0, p-1] \quad &\Leftrightarrow \quad k \in \left[\frac{n-p+1}{2}, \frac{n}{2}\right] \\ m-2l \in [0, p-1] \quad &\Leftrightarrow \quad l \in \left[\frac{m-p+1}{2}, \frac{m}{2}\right] \end{aligned}$$

Ceci permet de modifier les bornes des deux sommes dans (3.11), on pose ensuite $K = n-2k$ et $L = m-2l$ d'où $k = \frac{n-K}{2}$ et $l = \frac{m-L}{2}$. K et L sont alors introduits dans la formule de telle sorte que k et l peuvent être supprimés.

Finalement nous obtenons

$$IA^d_{2^j} f(n,m) = \sum_{K=0}^{p-1} \sum_{L=0}^{p-1} h(K)h(L) A^d_{2^j} f\left(\frac{n-K}{2}, \frac{m-L}{2}\right)$$

$$\text{avec } \tfrac{n-K}{2}, \tfrac{m-L}{2} \in Z$$

Comme pour les formules d'analyse en appliquant la même méthode aux formules interpolant les détails, on obtient les formules suivantes

$$\begin{aligned} ID^1_{2^j} f(n,m) &= \sum_{K=0}^{p-1} \sum_{L=2-p}^{1} h(K)g(L) D^1_{2^j} f\left(\frac{n-K}{2}, \frac{m-L}{2}\right) \\ ID^2_{2^j} f(n,m) &= \sum_{K=2-p}^{1} \sum_{L=0}^{p-1} g(K)h(L) D^2_{2^j} f\left(\frac{n-K}{2}, \frac{m-L}{2}\right) \\ ID^3_{2^j} f(n,m) &= \sum_{K=2-p}^{1} \sum_{L=2-p}^{1} g(K)g(L) D^3_{2^j} f\left(\frac{n-K}{2}, \frac{m-L}{2}\right) \end{aligned}$$

Comme précédemment et conformément à la *Figure (2.2)* on éclate les formules en deux étapes distinctes, pour cela on introduit $IC^d_{2^j}$, $IC^1_{2^j}$, $IC^2_{2^j}$, et $IC^3_{2^j}$ qui correspondent au résultat de la convolution et décimation à la cadence 2 des colonnes des différentes sous-images obtenues lors de l'analyse.

D'où :

$$IC^d_{2^j} f(n,m) = \sum_{L=0}^{p-1} h(L) A^d_{2^j} f\left(n, \frac{m-L}{2}\right)$$

$$
\begin{aligned}
IC^1_{2^j} f(n,m) &= \sum_{L=2-p}^{1} g(L) D^1_{2^j} f\left(n, \frac{m-L}{2}\right) \\
IC^2_{2^j} f(n,m) &= \sum_{L=0}^{p-1} h(L) D^2_{2^j} f\left(n, \frac{m-L}{2}\right) \\
IC^3_{2^j} f(n,m) &= \sum_{L=2-p}^{1} g(L) D^3_{2^j} f\left(n, \frac{m-L}{2}\right)
\end{aligned}
$$

On effectue ensuite la convolution et décimation à la cadence 2 des lignes des sommes $IC^d_{2^j} + IC^1_{2^j}$ et $IC^2_{2^j} + IC^3_{2^j}$.

L'image reconstruite correspondant alors finalement à la somme du résultat des deux opérations précédentes

$$
\begin{aligned}
&A^d_{2^{j+1}} f(n,m) \\
&= \sum_{K=0}^{p-1} h(K)(IC^d_{2^j} + IC^1_{2^j}) \left(\frac{n-K}{2}, m\right) + \sum_{K=2-p}^{1} g(K)(IC^2_{2^j} + IC^3_{2^j}) \left(\frac{n-K}{2}, m\right)
\end{aligned}
$$

$$\text{avec } \tfrac{n-K}{2} \in Z$$

3.3 Traitement des bords de l'image

Les bords de l'image posent problème du fait de la convolution. En effet l'analyse et la synthèse de l'image se font en convoluant l'image avec les filtres $\tilde{H}$, $\tilde{G}$ et H, G respectivement puis sous-échantillonnage à la cadence 2 dans le premier cas et interpolation dans le second.

Considérons un signal p monodimensionnel et les filtres dérivés de l'ondelette de Daubechies à 6 moments nuls (les filtres on un support de longueur 6), le problème posé dans la phase d'analyse peut alors être illustré par le schéma ci-dessous. Celui-ci montre le lien entre les coefficients du signal et les filtres $\tilde{H}$, $\tilde{G}$ pour obtenir $A_{2^{-1}} p(0)$ et $D_{2^{-1}} p(0)$ respectivement.

$$
\begin{array}{cccccccccc}
 & & & & \tilde{h}_{(0)} & \tilde{h}_{(-1)} & \tilde{h}_{(-2)} & \tilde{h}_{(-3)} & \tilde{h}_{(-4)} & \tilde{h}_{(-5)} \\
 & & & & p_0 & p_1 & p_2 & p_3 & p_4 & p_5 \quad p_6 \quad \cdots \\
\tilde{g}_{(4)} & \tilde{g}_{(3)} & \tilde{g}_{(2)} & \tilde{g}_{(1)} & \tilde{g}_{(0)} & \tilde{g}_{(-1)} & & & &
\end{array}
$$

Il est clair que le filtre $\tilde{H}$ nécessite une extension après la fin du signal et $\tilde{G}$ une extension avant le début du signal. Ce phénomène se produisant également dans la phase de synthèse avec les filtres H et G, il faut prolonger le support de l'image en ajoutant des données supplémentaires avant toute opération de convolution.

Diverses solutions ont été proposées [13] pour prolonger l'image :

1. Le signal est supposé nul en dehors de son support original
2. Périodisation du signal
3. Le signal est supposé continuement constant à partir de sa valeur extrème
4. Le signal est symétrique par rapport aux bords

5. Le signal est doublement symétrique par rapport à l'espace et à l'amplitude

Nous avons choisi de prolonger l'image par la méthode 3., elle est simple, assure la continuité au niveau des bords et introduit des distorsions peu visibles. Donc pour une image de taille $n \times n$, formée de pixels p_{xy} on a

			...	...	...	...	...			
			p_{00}	p_{10}	p_{20}	...	$p_{(n-1)0}$			
...	p_{00}	p_{00}	p_{00}	p_{10}	p_{20}	...	$p_{(n-1)0}$	$p_{(n-1)0}$	$p_{(n-1)0}$	...
...	p_{01}	p_{01}	p_{01}	p_{11}	p_{21}	...	$p_{(n-1)1}$	$p_{(n-1)1}$	$p_{(n-1)1}$	...

Cependant cette stratégie n'est applicable que dans la phase d'analyse, en effet si dans la phase de synthèse on prolongeait l'approximation et les signaux de détails comme précédemment on verrait apparaître des effets de bords dans l'image reconstruite. La largeur en nombre de pixels des bords qui sont modifiés sur les côtés de l'image est estimée dans [13] a $(2^j - 1)(P - 1)$ pour une image analysée sur j niveaux avec des filtres de longueur P.

Pour supprimer ces effets de bords et donc avoir une image reconstruite sans pertes par rapport à l'originale, le calcul de coefficients transformés et d'ondelettes supplémentaires s'avère nécessaire. En effet dans le cas du signal p précédent décomposé en $A_{2^{-1}}p$ et $D_{2^{-1}}p$, le schéma ci-dessous montre que pour reconstruire p les filtres H et G ont respectivement besoin de 2 coefficients transformés en plus au début du signal approché et 2 coefficients d'ondelettes supplémentaires à la fin du signal des détails d.

$Ia_0 \rightarrow$	$h_{(4)}$	$h_{(2)}$	$h_{(0)}$				$h_{(4)}$	$h_{(2)}$	$h_{(0)}$	$\rightarrow Ia_{n-2}$
	a_{-2}	a_{-1}	a_0	a_1	a_2	...	$a_{\frac{n}{2}-3}$	$a_{\frac{n}{2}-2}$	$a_{\frac{n}{2}-1}$	
$Ia_1 \rightarrow$	$h_{(5)}$	$h_{(3)}$	$h_{(1)}$				$h_{(5)}$	$h_{(3)}$	$h_{(1)}$	$\rightarrow Ia_{n-1}$

Interpolation par H des extrémités du signal approximé.

$Id_0 \rightarrow$	$g_{(0)}$	$g_{(-2)}$	$g_{(-4)}$				$g_{(0)}$	$g_{(-2)}$	$g_{(-4)}$	$\rightarrow Id_{n-2}$
	d_0	d_1	d_2	...	$d_{\frac{n}{2}-2}$	$d_{\frac{n}{2}-1}$	$d_{\frac{n}{2}}$	$d_{\frac{n}{2}+1}$	$d_{\frac{n}{2}+2}$	
$Id_1 \rightarrow$	$g_{(1)}$	$g_{(-1)}$	$g_{(-3)}$				$g_{(1)}$	$g_{(-1)}$	$g_{(-3)}$	$\rightarrow Id_{n-1}$

Interpolation par G des extrémités du signal des détails.

De manière générale pour des filtres H et G dont la réponse impulsionnelle est de taille P il faut respectivement $P - 2$ coefficients transformés et $P - 2$ coefficients d'ondelettes supplémentaires. On peut remarquer que l'ondelette de Haar ne pose pas de problèmes au niveau des bords puisque les filtres qui en sont dérivés sont de taille 2.

3.4 Exemples d'images décomposées

Les images obtenues sont présentées suivant le schéma de la *Figure (3.1)*.

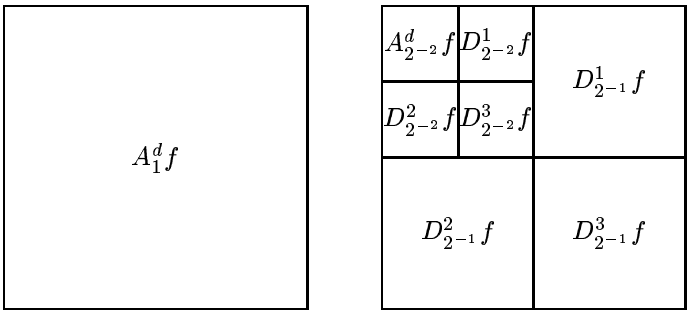


FIG. 3.1 - *Schéma de représentation en ondelettes d'une image $A_1^d f$ décomposée sur 2 niveaux de résolution.*

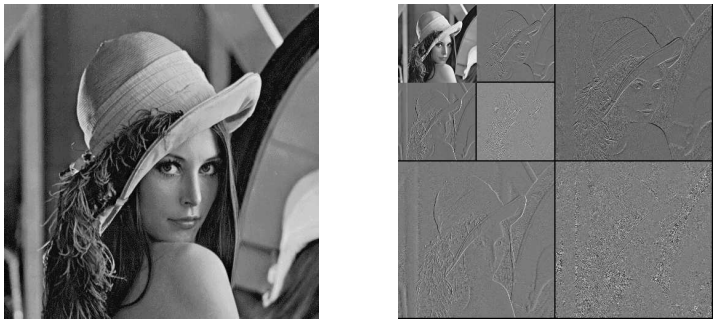


FIG. 3.2 - *Décomposition de Lena sur 2 niveaux de résolution.*

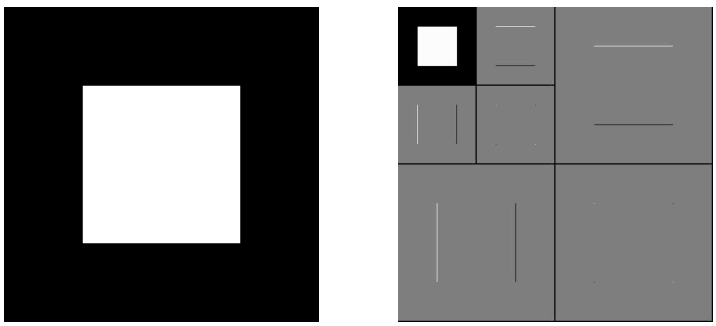


FIG. 3.3 - *Décomposition d'un carré sur 2 niveaux de résolution.*

Chapitre 4

Les algorithmes de codage EZW et SPIHT

4.1 Introduction

Dans ce chapitre nous allons voir deux algorithmes de compression d'images par ondelettes qui forment une alternative intéressante aux algorithmes classiques (quantification scalaire ou vectorielle suivi d'un codage entropique). Après avoir donné le concept sous-jacent à ces algorithmes nous verrons les grandes lignes de l'algorithme *EZW* développé par Shapiro [5]. Puis nous donnerons sommairement les modifications apportées par Said et Pearlman [6] à l'algorithme *EZW* pour aboutir à la définition de l'algorithme *SPIHT*.

L'idée à la base de ces algorithmes est de trouver le meilleur ordre de transmission des coefficients de la représentation en ondelettes. Il est clair que la transmission des coefficients dans l'ordre décroissant de leur valeur absolue est la meilleure solution, puisque les coefficients les plus significatifs sont ceux dont la valeur absolue est la plus élevée. Shapiro proposa 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. Ceci permet aux algorithmes *EZW* et *SPIHT* de faire de la transmission progressive d'image puisque le décodeur peut s'arrêter n'importe où dans la suite de bits transmise et produire la meilleure image reconstruite possible avec cette suite de bits tronquée. Ces algorithmes présentent en plus l'avantage de ne nécessiter ni phase d'apprentissage, ni dictionnaire, ni d'informations statistiques sur l'image source.

4.2 Embedded Zerotree Wavelet (*EZW*)

Après avoir calculé une transformée en ondelettes de l'image, l'algorithme code les coefficients transformés à l'aide d'une suite décroissante de seuils $T_0, ..., T_{N-1}$ avec $T_i = \frac{T_{i-1}}{2}$ et $T_0 < 2|c|$ pour tout coefficient c de la représentation en ondelettes. Pour coder les coefficients, l'algorithme effectue récursivement deux passes successives, ne traitant à chaque fois que les coefficients significatifs par rapport au seuil courant, i.e. ceux dont la valeur absolue est supérieure au seuil.

Dans la première passe, la "dominante", l'algorithme parcourt les coefficients de la transformée en ondelettes suivant l'ordre donné par la *Figure (4.1(a))* à la recherche des coefficients significatifs par rapport au seuil courant. En utilisant la hiérarchie donnée par la *Figure (4.1(b))*

l'algorithme produit alors une sorte de carte marquant la position des coefficients significatifs ainsi que leur signe. Cette carte est obtenue en associant à chaque coefficient suivant sa valeur absolue et celle de ses fils l'un des symboles suivant : *Zerotree root (ZTR), Isolated zero (IZ), Positive significant (POS)* et *Negative significant (NEG).* Un coefficient est un *Zerotree root* si lui et tous ses descendants ne sont pas significatifs, aucun symbole n'est alors associé à ses descendants. *Isolated zero* signifie que le coefficient n'est pas significatif mais a des descendants qui le sont. Enfin les coefficients significatifs sont marqués *Positive* ou *Negative significant* suivant que le coefficient est positif ou négatif. Chaque coefficient significatif est ensuite mis à zéro dans

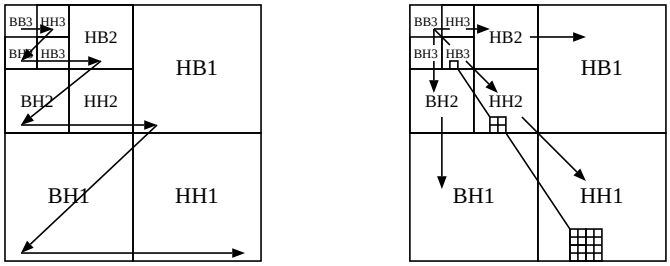


FIG. 4.1 - *(a) Ordre de parcours des coefficients et (b) organisation hiérarchique des coefficients.*

la transformée en ondelettes afin que sa position ne soit plus encodée et sa valeur absolue est placée dans une liste pour la coder par approximations successives. En effet chaque carte est suivie d'une suite de symboles "0" et "1" qui permettent au décodeur de fixer une valeur de reconstruction approximative aux coefficients significatifs. Cette valeur s'affine pour se rapprocher de plus en plus de la valeur réelle des coefficients au fur et à mesure que des suites de symboles sont encodés. Voyons comment cette suite est obtenue : si T_i est le seuil courant, alors les coefficients marqués dans la passe précédente on leur valeur absolue dans l'intervalle $[T_i, 2T_i[$ cet intervalle est alors divisé en deux intervalles $[T_i, \frac{3T_i}{2}[$ et $[\frac{3T_i}{2}, 2T_i[$. Aux coefficients dont la valeur absolue se trouve dans le premier intervalle on associe le symbole "0" alors que à ceux se trouvant dans le second intervalle on associe le symbole "1". Lorsque la seconde passe est finie l'algorithme reprend le processus et génère la carte suivante, le nouveau seuil étant T_{i+1} dans la seconde passe un nouvel intervalle s'ajoute au deux précédent : $[T_{i+1}, T_i[$. Ces trois intervalles sont alors raffinés comme dans la passe du cycle précédent pour transmettre une suite de symboles "0" ou "1", chaque symbole étant associé à un coefficient significatif. Lorsque le seuil initial T_0 est un multiple d'une puissance de deux cette stratégie peut être vue comme la transmission des bits de la représentation binaire de la valeur absolue des coefficients, en commençant par les bits les plus significatifs. Ce processus récursif s'arrête lorsque T_{N-1} est atteint ou que le nombre de bits désiré a été transmis. Pour terminer, avant transmission, l'algorithme compresse les suites de symboles produites par un codeur arithmétique adaptatif basé sur celui de Witten et al. [11]

4.3 Set Partionning In Hierarchical Trees (*SPIHT*)

Cet algorithme proposé par Said et Pearlman [6] diffère de l'*EZW* au niveau du partitionnement des coefficients de la transformée en ondelette utilisé pour déterminer ceux qui sont

significatifs, et par la manière de transmettre progressivement la valeur de ces derniers. De plus, dans l'algorithme de Shapiro, l'étape de codage du bit stream par le codeur arithmétique est essentielle pour réduire l'information sur l'ordre des coefficients significatifs, ce qui n'est pas le cas dans l'algorithme *SPIHT*. En effet l'efficacité du partitionnement et le compactage de l'information sur la valeur des coefficients significatifs font que les résultats obtenus par l'algorithme *SPIHT* sans utilisation du codeur arithmétique, sont pratiquement équivalents à ceux de l'algorithme *EZW*.

Le schéma de transmission de la valeur des coefficients significatifs repose sur les concepts suivants :

1. *Classification des coefficients dans l'ordre décroissant de leur valeur absolue.*

2. *Transmission des bits les plus significatifs en premier.*

Un aspect important est le fait que l'algorithme *SPIHT* ne transmet pas les informations sur l'ordonnancement des coefficients comme l'algorithme *EZW*. En effet la méthode de codage de l'ordre des coefficients est basée sur le fait que l'exécution d'un algorithme est complètement définie par les résultats des comparaisons au niveau des branchements de l'algorithme. Par conséquent si l'encodeur et le décodeur utilise le même algorithme de recherche des coefficients significatifs, i.e. que les coefficients transformés de position identique sont traités dans le même ordre, la transmission du chemin d'exécution de l'encodeur au décodeur permet à celui-ci de déterminer l'ordre des coefficients significatifs. Le chemin d'exécution de l'algorithme est produit lors de chaque passe de recherche des coefficients significatifs c qui sont tels que $|c| \geq 2^n$, où n est décrémenté d'une passe à une autre. Pour rechercher les coefficients significatifs l'algorithme partitionne les coefficients en ensembles, ensuite pour chaque ensemble l'encodeur transmet "1" au décodeur si il contient un coefficient significatif et "0" sinon. Lorsque le décodeur reçoit "0" il sait quel ensemble n'est pas significatif puisque l'encodeur et le décodeur partitionnent les coefficients de la même manière. Lorsqu'un ensemble est significatif il est divisé en sous-ensembles par une règle commune à l'encodeur et au décodeur, ces nouveaux ensembles sont à leur tour traités. Le processus de division successive s'arrête lorsque tous les coefficients significatifs ont été déterminés. Pour réduire le nombre de comparaison, l'algorithme *SPIHT* utilise une structure appelée *spatial orientation tree* qui définit une relation hiérarchique entre les coefficients de sous-bandes de même direction et de même position spatiale, identique à celle utilisée par Shapiro qui est donnée à la *Figure (4.1(b))*.

Chapitre 5

Parallélisation de la transformée en ondelettes

5.1 Introduction

En général les algorithmes de traitement numérique des images qui travaillent sur les pixels, et notamment l'algorithme de calcul de la transformée en ondelettes (2.4) ont une complexité calculatoire qui dépend du volume des données à traiter, i.e. du nombre de pixels de l'image. La structure de données fréquement utilisée en traitement d'images est un tableau bidimensionnel représentant la distribution des niveaux de gris ou de couleur de l'image. Les algorithmes consistent alors à appliquer une suite d'opérations sur chaque élément (pixel) du tableau. Le volume de calcul induit par ces algorithmes peut rarement être traité par une machine séquentielle dans un laps de temps restreint comme l'impose les applications temps réel. Cependant les différentes opérations peuvent être appliquées en parallèle sur chaque élément du tableau. C'est pourquoi les machines parallèles semblent bien adaptées pour implémenter de manière efficace ce type d'algorithme, et permettre de répondre aux contraintes du temps réel.

5.2 Les architectures parallèles existantes

Les machines parallèles sont souvent classées suivant leur modèle d'exécution de programmes, c'est à dire la multiplicité des flots de contrôle et de données, ainsi que suivant leur modèle de programmation.

1. **Modèle d'exécution**

   - SIMD (*Single Instruction Multiple Data*) : une seule unité de contrôle et un ensemble d'unités de calcul, i.e. chaque processeur élémentaire exécute le même programme simultanément, l'unité de contrôle s'occupant de la diffusion de l'instruction à exécuter.
   - MIMD (*Multiple Instruction Multiple Data*) : une unité de contrôle est associée à chaque unité de calcul. Ceci permet d'exécuter des programmes différents parallèlement.
   - MISD (*Multiple Instruction Single Data*) : les données sont transmises séquentiellement et traitées par une suite d'opérateurs spécialisés.

2. **Modèles de programmation**

   - *Parallélisme de contrôle*
     Ce modèle définit un programme à un ensemble de processus séquentiels communicants par l'intermédiaire de messages ou de variables partagées.
   - *Parallélisme de données*
     Ce modèle est particulièrement bien adapté aux machines SIMD : il peut être vu comme une suite de traitements séquentiels sur des données à accès parallèle. Le point essentiel dans ce modèle est la distribution des données pour réduire au minimum les communications qui sont sources de pertes de temps de calcul.

Cependant comme le souligne Charot [19], cette classification ne permet pas d'appréhender la diversité des architetures parallèles qui existent. On introduit donc des critères supplémentaires qui permettront d'affiner la classification :

- la topologie du réseau d'interconnexion (grille, tore, hyperarbre, anneau, ...),
- l'architecture du processeur utilisé comme nœud de calculs,
- l'autonomie des processeurs pour les architectures *SIMD*.

5.3 Architectures utilisées pour la parallélisation de la transformée en ondelettes

La parallélisation de la transformée en ondelettes a été étudiée sur de nombreuses architectures, et notamment :

- Les machines dont la topologie du réseau d'interconnexion est un maillage, qu'il soit bidimensionnel ou linéaire. Une version parallèle de la transformée en ondelettes sur réseau maillé est proposée par Lu [20], et dans [9] l'étude de l'implémentation de la transformée en ondelettes sur différentes architectures a montrée qu'un réseau linéaire de processeurs (la machine SYMPHONIE) permet de faire du temps réel. Il faut remarquer que ces architectures sont le plus courament utilisé dans le cadre de la parallélisation d'algorithmes de traitement d'images.

- La recherche dans le domaine des architectures VLSI est également très intense [7, 18]. L'objectif de ces architectures est de produire un composant occupant le moins de place possible et permettant de faire un traitement en parallèle optimal. Ces architectures très spécialisées donnent d'ailleurs des résultats bien meilleurs que ceux d'une machine massivement parallèle.

- La parallélisation de la transformée en ondelettes sur un réseau pyramidal est aussi très étudiée [20]. En effet cette architecture trouve son origine notamment dans les traitements multiéchelle dont fait partie la transformée en ondelettes, et devrait donc en théorie permettre d'obtenir une implémentation sur machine massivement parallèle optimale.

Pour des raisons de temps nous avons restreint l'étude au réseau maillé bidimensionnel.

5.4 Parallélisation de la transformée en ondelettes sur réseau maillé bidimensionnel

5.4.1 Introduction

On a vu lors de l'étude de l'algorithme pyramidal de Mallat que la structure algorithmique des phases d'analyse et de synthèse sont identiques, aussi ne donnerons nous qu'une version parallèle de la phase d'analyse. Nous supposerons que le réseau maillé considéré a autant de nœuds de calculs que de pixels. Nous faisons également l'hypothèse que le maillage est 4-connexe, i.e. que tout nœud de calculs est connecté à ses voisins nord, sud, est et ouest. Pour avoir une version parallèle efficace il est évident que les communications ne devront se faire qu'entre nœuds de calculs voisins. En effet les communications entre nœuds de calculs distant ralentissent l'exécution du programme car les données ne sont déplacées sur le réseau que d'une position par cycle d'horloge.

Nous avons vu au chapitre 3 que lors de l'implémentation séquentielle, la convolution et le sous-échantillonnage à la cadence 2 étaient fait en même temps, ce qui permettait de réduire de moitié le volume de calculs. Dans le cas d'une implémentation parallèle il est préférable de bien dissocier ces deux étapes, afin de diviser la complexité en calculs et en communications. La transformée en ondelettes est alors vue comme la suite de deux opérations :

1. convolution,
2. sous-échantillonnage à la cadence 2,

appliquées successivement aux lignes et aux colonnes. Ceci d'autant plus que nous utilisons des ondelettes orthogonales qui induisent des filtres séparables, car dans le cas de filtres non séparables le sous-échantillonnage ne le serait pas non plus.

5.4.2 Parallélisation de la convolution 2-D sur réseau maillé

Avant de voir en détail l'algorithme parallèle, nous allons étudier la parallélisation de la convolution 2-D sur un réseau maillé. En général les algorithmes de traitement d'images convoluent une image I de taille $N \times N$ avec un noyau K de taille L : l'idée est de voir la convolution comme la superposition de L^2 fenêtres de taille $N \times N$ et donc de paralléliser le calcul des pixels dans chaque fenêtre. Pour résoudre le problème relatif aux bords de l'image, nous supposerons que le réseau est bouclé sur lui-même de sorte que chaque ligne et colonne de processeurs du réseau forme un tore. La formule permettant de calculer la convolution 2-D étant : $C_{i,j} = \sum_{s=0}^{L-1} \sum_{t=0}^{L-1} K_{s,t} I_{i-s,j-t}$, l'exemple de la *Figure (5.1)* illustre l'idée précédente.

00	01
10	11

00	01	02	03
10	11	12	13
20	21	22	23
30	31	32	33

Noyau K *Image de taille* 4×4

De nombreux algorithmes parallèles de calcul de la convolution 2-D ont été définis suivant cette idée, ne différant qu'au niveau de la stratégie de communication des données entre processeurs. La version proposée par Maresca et Li appelé *Snake Sweeping* car les communications des données suivent le "mouvement" d'un serpent est notamment présentée par Lu dans [20].

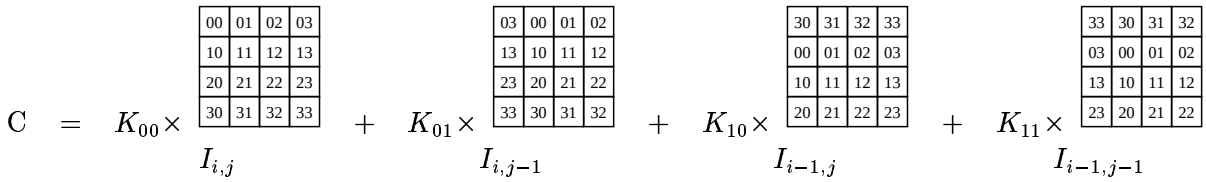


FIG. 5.1 - *Exemple de convolution 2-D entre un noyau K et une image I.*

5.4.3 Algorithme parallèle de calcul de la transformée en ondelettes

Quelques remarques relatives à l'algorithme :

1. Puisque nous utilisons des ondelettes orthogonales, les filtres qui en sont dérivés sont séparables. Ce qui signifie que si on se ramène à la convolution 2-D, le noyau de convolution est séparable en un vecteur ligne (les coefficients du filtre H) et en un vecteur colonne (les coefficients du filtre G).

2. Les bords de l'image ne posent pas de problème en raison du bouclage du réseau sur lui-même, i.e. si on décale la première ligne de l'image vers le nord, elle remplacera la dernière ligne comme on peut le voir la *Figure (5.1)*.

3. Après convolution des lignes, seuls les nœuds de calculs d'abscisse paires contiennent des données qui seront utilisées pour convoluer les colonnes. De même après convolutions des colonnes, uniquement les nœuds de calculs d'ordonnée paire contiennent des coefficients transformés. La suppression des coefficients inutiles est l'objet des phases de sous-échantillonnage à la cadence 2 qui suit immédiatement une phase de convolution. Dans le cas d'un réseau maillé la solution la plus simple consiste à désactiver les nœuds de calculs qui ne sont pas valides. Cette solution présente cependant un inconvénient de taille : la distance entre les nœuds de calculs valides est multipliée par deux d'une phase d'analyse à une autre, ce qui augmente le coût en communications de l'algorithme. Pour résoudre ce problème nous allons compacter les données après chaque phase de convolution, la *Figure (5.2)* montre le résultat du compactage après la convolution des lignes et la *Figure (5.3)* le résultat dans le cas des colonnes sur un réseau 4×4. Le résultat du compactage donne à la fin de la phase d'analyse une vision classique de la transformée. De plus cette répartition des coefficients fait que dans la phase d'analyse suivante seul le quart nord-ouest du réseau doit-être actif puisque seul le signal approché est à nouveau décomposé. Il faut remarquer que la position des signaux des détails $D^1_{2^j}$ et $D^2_{2^j}$ est inversée par rapport à la représentation en ondelettes obtenue en séquentiel.

Algorithme parallèle de la transformée en ondelettes

L'algorithme que nous allons donner est exprimé dans le modèle à parallélisme de données. En effet le data-parallélisme est la technique de parallélisation la mieux adaptée à la transformée en ondelettes. L'algorithme se différencie de celui de Lu [20] essentiellement au niveau de la stratégie de distribution des données.

On fait l'hypothèse que l'image et le réseau maillé sont de taille $N \times N$, que les filtres H et G sont ceux déduits de l'ondelette de Haar et sont donc de longueur 2. J correspond au nombre de niveaux d'analyse désiré.

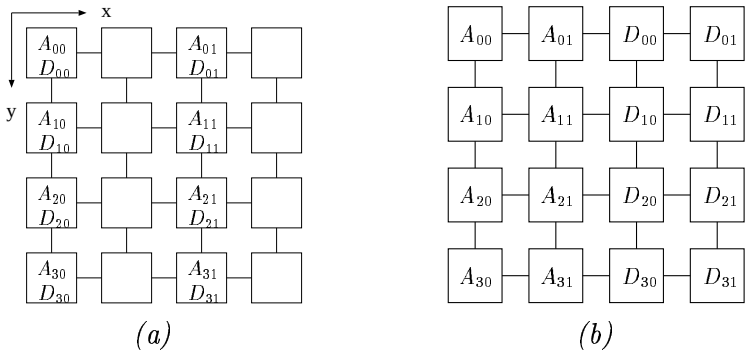


FIG. 5.2 - *(a) Position des coefficients du signal approché A et du signal des détails D, et (b) position des coefficients après compactage.*

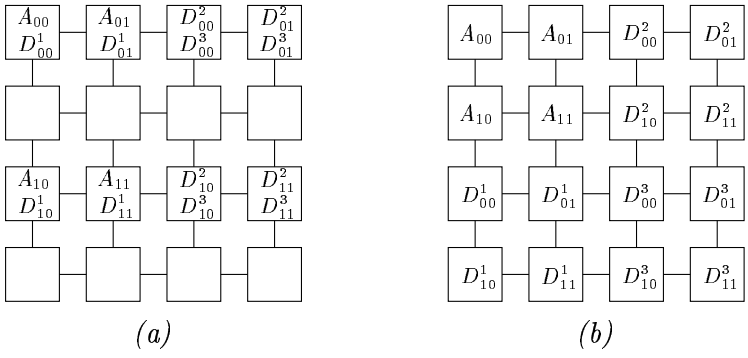


FIG. 5.3 - *(a) Position des coefficients du signal approché A et des signaux des détails D^1, D^2 et D^3, (b) position des coefficients après compactage.*

On notera les nœuds de calculs PN (*processing node*). Chacun gère les variables suivantes :

- *image* : variable contenant l'image;
- *temp1,temp2* : variables temporaires;
- a : approximation à la résolution 2^j;
- *d1,d2,d3* : images des détails à la résolution 2^j, respectivement horizontaux, verticaux et diagonaux.

On définit également les fonctions suivantes :

- **ActiverPN(n)** : active un réseau de $n \times n$ nœuds de calculs dont l'origine est dans le coin nord-ouest,
- **DataTransfert(variable,s,t)** : transfert la valeur de **variable** du nœud de calculs de position (i,j) au nœud de calculs de position (i-s modulo N,j-t modulo N).

```
Algorithme CalculDWT
% J est le nombre de niveaux de décomposition
Pour j=1 à J faire
  ActiverPN(N/2^(j-1))
  % Convolution lignes
  Pour tout PN actifs faire
    temp1 = image
    a = temp1 × h[0]
    d2 = temp1 × g[0]
    DataTransfert(temp1,1,0) % Décalages des lignes vers l'ouest
    a = a + temp1 × h[1]
    d2 = d2 + temp1 × g[1]
  finpour
  % Compactage des données
  Pour i de 0 à N/2^j faire
    Pour tous les PN d'abscisse >  i faire
      DataTransfert(a,1,0) % Décalages vers l'ouest
    finpour
    Pour tous les PN d'abscisse <  N/2^(j-1) − i faire
      DataTransfert(d2,-1,0) % Décalages vers l'est
    finpour
  finpour
  % Convolution colonnes
  Pour tout les PN actifs faire
    % Décomposition du signal approché a
    temp1 = a
    a = temp1 × h[0]
    d1 = temp1 × g[0]
    DataTransfert(temp1,0,1) % Décalages des lignes vers le nord
    a = a + temp1 × h[1]
```

```
        d1 = d1 + temp1 × g[1]
        % Décomposition du signal des détails d2
        temp2 = d2
        d2 = temp2 × h[0]
        d3 = temp2 × g[0]
        DataTransfert(temp2,0,1) % Décalages des lignes vers le nord
        d2 = d2 + temp2 × h[1]
        d3 = d3 + temp2 × g[1]
      finpour
      % Compactage des données
      Pour i de 0 à N/2^j faire
        Pour tous les PN d'ordonnée >  i faire
          DataTransfert(a,0,1) % Décalages vers le nord
          DataTRansfert(d2,0,1)
        finpour
        Pour tous les PN d'ordonnée <  N/2^(j-1) − i faire
          DataTransfert(d1,0,-1) % Décalages vers le sud
          DataTransfert(d3,0,-1)
finpour
```

Chapitre 6

Implémentation sur machine parallèle

6.1 Introduction

La version parallèle de la transformée en ondelettes proposée dans le chapitre précédent étant basée sur le data-parallélisme nous avons décidé de l'implémenter sur la *Connection Machine CM-5* [21] du Centre National de Calcul Parallèle en Sciences de la Terre à l'aide du langage data-parallèle C* ("C-star").

6.2 La Connection Machine CM-5

La CM-5 est une machine MIMD à mémoire distribuée permettant de faire du contrôle MIMD ou SIMD. Elle donne accès à plusieurs modes de programmation parmi lesquels le parallélisme de données, et le SPMD qui est un compromis entre le parallélisme de contrôle et la programmation SIMD. La CM-5 est composée d'un ensemble de nœuds de calculs (*processing nodes*) qui sont des processeurs RISC SPARC-2 munis d'unités vectorielles, chaque nœud peut en théorie atteindre une puissance de calcul de 128 MFlops. Cet ensemble de nœuds de calculs est divisé en groupes plus petits de taille variable appelés *partitions*. Chaque partition est contrôlée par un processeur de contrôle appelé le *partition manager*, généralement il s'agit d'une SPARCstation Sun, globalement on peut donc voir la CM-5 également comme une machine multi-SIMD. La machine du CNCPST qui est composée de 128 nœuds de calculs est ainsi divisée les jours de la semaine en trois partitions 64 - 32 - 32 et le week-end en une partition unique de 128 nœuds. Le réseau d'interconnexion des processeurs de calculs à la topologie d'un hyperarbre de processeurs comme le montre la *Figure (6.1)*, mais peut simuler d'autres topologies telles que grille, tore, réseau linéaire, pyramide.

Dans le cadre de la programmation data-parallèle le parallélisme s'obtient en projetant chaque élément à traiter sur un nœud de calculs. Cependant le nombre de processeurs de la machine est souvent inférieur au nombre de données à traiter, c'est pourquoi chaque processeur physique simule en fait un certain nombre de processeurs virtuels. L'objectif majeur dans les programmes data-parallèle et donc de bien contrôler la distribution des données sur les nœuds de calculs afin de réduire au maximum les communications. La programmation data-parallèle sur la CM-5 est facilitée par le fait que la distribution des données et les communications inter-

processeurs sont gérées directement par le compilateur data-parallèle. D'autre part les nœuds de calculs de la partition active ne sont utilisés que pour des calculs parallèles puisque les parties séquentielles du programme sont exécutées par le processeur du partition manager.

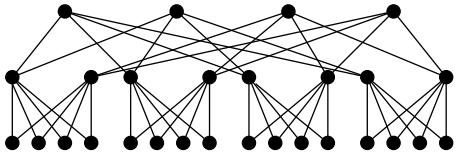


FIG. 6.1 - *Réseau d'interconnexion de la Connection Machine CM-5.*

6.3 Le langage data-parallèle C*

Le langage C* est une extension de la version ANSI du langage C permettant de faire efficacement de la programmation data-parallèle sur la CM-5. C* fournit un modèle de programmation global car les données sont réparties sur l'ensemble des processeurs de la partition et les instructions C* agissent sur la totalité des données. La version data-parallèle du langage C présente outre les caractéristiques classiques du C un certain nombre d'éléments nouveaux tant au plan opérateurs que structures de contrôle.

Tout d'abord nous allons voir comment sont définies les variables parallèles. En C* la création d'un objet parallèle s'obtient en lui attribuant un *shape* en plus de son type classique, il est ainsi possible d'avoir un shape d'entiers, un tableau de shape, un shape de structures, etc. Lors de la définition d'un shape un constructeur appelé *left indexing* permet de spécifier son nombre de dimensions et de positions dans chacune d'entre elles. Par exemple pour paralléliser le tableau bidimensionnel correspondant à une image de taille 256×256 on commence par déclarer un shape à deux dimensions de 256 positions chacune : `shape [256][256]ShapeImage;`. Ensuite on déclare l'image proprement dite : `double:ShapeImage image;`. Lorsqu'une variable parallèle intervient dans une opération de calcul (addition par exemple) celle-ci est faite en parallèle, dans le cas ou un scalaire apparaît dans l'opération celui-ci est parallélisé par le compilateur. Parmi les nouveaux opérateurs introduit par C* il y a `<?` et `>?` grâce auxquels on peut déterminer respectivement le minimum et le maximum entre deux variables. Les nouvelles structures de contrôle qui sont définies permettent de manipuler des variables parallèles : **`with`** pour sélectionner le type de shape que l'on va traiter, **`where`** pour restreindre les processeurs virtuels qui sont "actifs" et **`everywhere`** pour les activer tous.
Exemple : on met à un les pixels nuls de l'image.

```
with(ShapeImage) {
  where(image==0)
    image=1;
}
```

Enfin C* permet deux types de communications : *régulières et irrégulières.* Dans la première catégorie les processeurs transfèrent leur données du même nombre de positions dans la même direction, alors que dans la seconde catégorie le nombre de positions et la direction peuvent être différentes d'un processeur à un autre.

Chapitre 7

Résultats expérimentaux

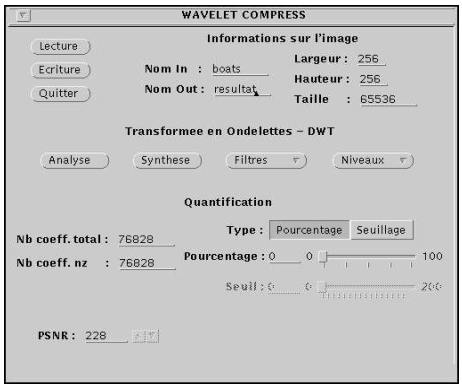
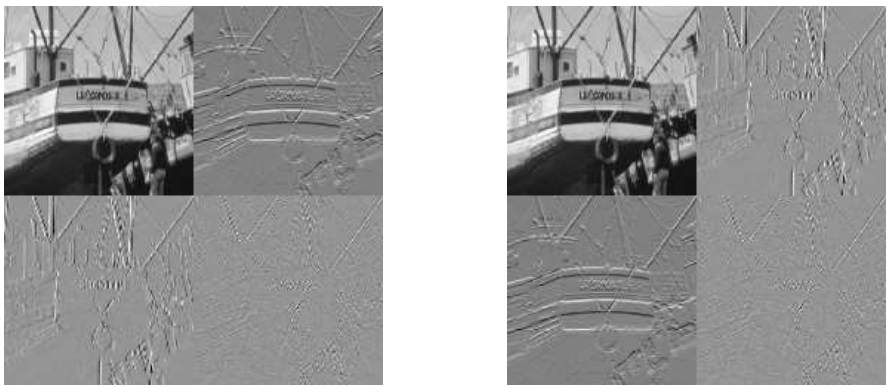
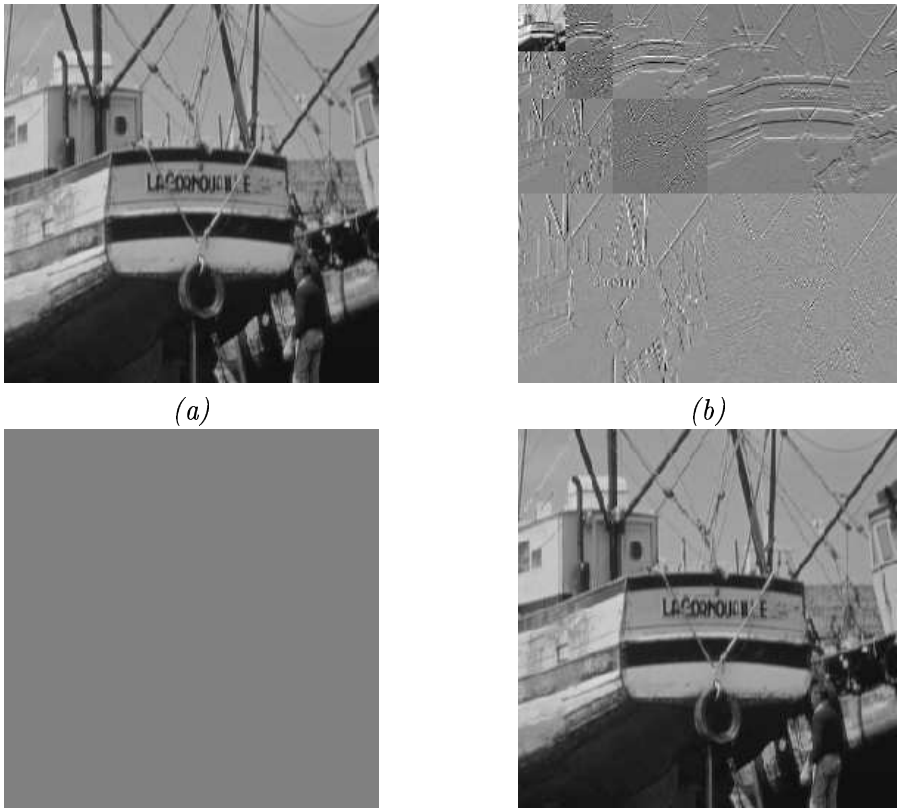


FIG. 7.1 - *Interface de l'implantation séquentielle de la transformée en ondelettes.*



(a) *(b)*

FIG. 7.2 - *(a) Résultat de la transformée en ondelettes séquentielle et (b) parallèle, la décomposition est sur 1 niveau et les filtres utilisés sont ceux dérivés de l'ondelette de Haar.*



(a) *(b)* *(d)* *(c)*

FIG. 7.3 - *(a) Image originale, (b) représentation en ondelettes sur 3 niveaux, (c) image reconstruite à partir de (b), et (d) image différence entre l'image originale et l'image reconstruite.*

L'uniformité de l'image différence montre que l'image reconstruite est de très bonne qualité, d'autant plus que le rapport signal sur bruit (PSNR) est de 228 dB.

Conclusion

Dans un premier temps nous avons étudié le schéma des algorithmes classiques (transformation en ondelettes pour décorréler les pixels, puis quantification et codage entropique des coefficients), ainsi que deux algorithmes plus évolués : l'*EZW* [5] et le *SPIHT* [6], qui utilisent une stratégie de quantification par approximations successives et permettent notamment de faire de la transmission progressive d'images.

L'étude de la représentation en ondelettes orthogonales proposée par Mallat [3] montre clairement comment il a abouti à l'algorithme pyramidal de calcul rapide de la transformée en ondelettes, celui que nous avons implanté. La version séquentielle que nous avons développée montre de manière évidente qu'une application temps réel utilisant la transformée en ondelettes ne peut être implantée sur une machine monoprocesseur.

La version data-parallèle de l'algorithme pyramidal de calcul rapide de Mallat de la transformée en ondelettes que nous proposons est basée sur celui de J. Lu [20]. La principale différence par rapport à ce dernier se situe au niveau de la distribution des données, et de la stratégie de compactage des coefficients. L'algorithme proposé tient compte du fait que chacun des deux signaux résultant de la convolution et du sous-échantillonnage des lignes de l'image est utilisé pour dériver deux signaux lors de la phase de convolution des colonnes. Pour le moment nous avons implanté sur la Connection Machine CM-5 uniquement la phase d'analyse de l'algorithme.

La version séquentielle de l'algorithme pyramidal permettra de procéder ultérieurement à une étude comparative entre les versions séquentielle et parallèle. Cependant l'implantation parallèle actuelle n'est pas optimale et devra être préalablement retravaillée avant toute étude comparative.

Les prochains travaux concernent l'implantation de la phase de synthèse et l'étude de la parallélisation des étapes suivantes de l'algorithme de compression d'images, i.e la phase de quantification et le codage des coefficients.

Annexe A

Démonstrations

A.1 Formules d'implémentation bidimensionnelle

On donne les différentes étapes permettant d'aboutir à l'expression de $D^1_{2^j} f$ à partir des filtres miroirs $\tilde{\mathrm{H}}$ et $\tilde{\mathrm{G}}$, ainsi qu'à celle de $ID^1_{2^j} f$ à partir des filtres H et G. L'idée étant la même pour aboutir à $A^d_{2^j} f, D^2_{2^j} f$ et $D^3_{2^j} f$ dans un cas et $IA^d_{2^j} f, ID^2_{2^j} f$ et $ID^3_{2^j} f$ dans l'autre.

A.1.1 Expression de $D^1_{2^j} f$

On sait que $D^1_{2^j} f = (\langle f(u,v), \Psi^1_{2^j}(u - 2^{-j}n, v - 2^{-j}m)\rangle)_{(n,m)\in Z^2}$

Or $\Psi^1_{2^j} \in V_{2^{j+1}}$, d'où l'expression de $\Psi^1_{2^j}$ dans la base de $V_{2^{j+1}}$

$$\begin{aligned}
&\Psi^1_{2^j}(u - 2^{-j}n, v - 2^{-j}m) \\
&= 2^{-2(j+1)} \sum_{k=-\infty}^{+\infty} \sum_{l=-\infty}^{+\infty} (\langle \Psi^1_{2^j}(u - 2^{-j}n, v - 2^{-j}m), \Phi_{2^{j+1}}(u - 2^{-j-1}k, v - 2^{-j-1}l)\rangle \\
&\quad \Phi_{2^{j+1}}(u - 2^{-j-1}k, v - 2^{-j-1}l))
\end{aligned}$$

On rappelle que les formules de décomposition pour $\Phi_{2^{j+1}}$ et $\Psi^1_{2^j}$ sont

$$\Phi_{2^{j+1}}(u - 2^{-j-1}k, v - 2^{-j-1}l) = \phi_{2^{j+1}}(u - 2^{-j-1}k)\phi_{2^{j+1}}(v - 2^{-j-1}l) \quad (A.1)$$

$$\Psi^1_{2^j}(u - 2^{-j}n, v - 2^{-j}m) = \phi_{2^j}(u - 2^{-j}n)\psi_{2^j}(v - 2^{-j}m) \quad (A.2)$$

On en déduit

$$\begin{aligned}
&2^{-2(j+1)} \langle \Psi^1_{2^j}(u - 2^{-j}n, v - 2^{-j}m), \Phi_{2^{j+1}}(u - 2^{-j-1}k, v - 2^{-j-1}l)\rangle \\
&= 2^{-2(j+1)} \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} \Psi^1_{2^j}(u - 2^{-j}n, v - 2^{-j}m)\ \Phi_{2^{j+1}}(u - 2^{-j-1}k, v - 2^{-j-1}l)\ dudv \\
&= 2^{-2(j+1)} \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} (\phi_{2^j}(u - 2^{-j}n)\psi_{2^j}(v - 2^{-j}m))\ (\phi_{2^{j+1}}(u - 2^{-j-1}k)\phi_{2^{j+1}}(v - 2^{-j-1}l))\ dudv \\
&= \left(\int_{-\infty}^{+\infty} 2^{-j-1}\phi_{2^j}(u - 2^{-j}n)\phi_{2^{j+1}}(u - 2^{-j-1}k)du\right) \left(\int_{-\infty}^{+\infty} 2^{-j-1}\psi_{2^j}(v - 2^{-j}m)\phi_{2^{j+1}}(v - 2^{-j-1}l)dv\right) \\
&= (2^{-j-1} \langle \phi_{2^j}(u - 2^{-j}n), \phi_{2^{j+1}}(u - 2^{-j-1}k)\rangle) (2^{-j-1} \langle \psi_{2^j}(v - 2^{-j}m), \phi_{2^{j+1}}(v - 2^{-j-1}l)\rangle) \\
&= \tilde{h}(2n - k)\tilde{g}(2m - l)
\end{aligned}$$

En effet on a

$$2^{-j-1}\left\langle \phi_{2^j}(u-2^{-j}n), \phi_{2^{j+1}}(u-2^{-j-1}k)\right\rangle = \langle \phi_{2^{-1}}(u), \phi(u-(k-2n))\rangle = \tilde{h}(2n-k)$$
$$2^{-j-1}\left\langle \psi_{2^j}(v-2^{-j}m), \phi_{2^{j+1}}(v-2^{-j-1}l)\right\rangle = \langle \psi_{2^{-1}}(v), \phi(v-(l-2m))\rangle = \tilde{g}(2m-l)$$

Donc

$$\Psi^1_{2^j}(u-2^{-j}n, v-2^{-j}m) = \sum_{k=-\infty}^{+\infty}\sum_{l=-\infty}^{+\infty} \tilde{h}(2n-k)\tilde{g}(2m-l)\ \ \Phi_{2^{j+1}}(u-2^{-j-1}k, v-2^{-j-1}l)$$

Finalement on obtient

$$\begin{aligned} D^1_{2^j} f(n,m) &= \left\langle f(u,v), \Psi^1_{2^j}(u-2^{-j}n, v-2^{-j}m)\right\rangle \\ &= \left\langle f(u,v), \sum_{k=-\infty}^{+\infty}\sum_{l=-\infty}^{+\infty} \tilde{h}(2n-k)\tilde{g}(2m-l)\ \ \Phi_{2^{j+1}}(u-2^{-j-1}k, v-2^{-j-1}l)\right\rangle \\ &= \sum_{k=-\infty}^{+\infty}\sum_{l=-\infty}^{+\infty} \tilde{h}(2n-k)\tilde{g}(2m-l)\ \ \left\langle f(x,y), \Phi_{2^{j+1}}(u-2^{-j-1}k, v-2^{-j-1}l)\right\rangle \end{aligned}$$

Or $A^d_{2^{j+1}} f(k,l) = \langle f(u,v), \Phi_{2^{j+1}}(u-2^{-j-1}k, v-2^{-j-1}l)\rangle$

Donc

$$D^1_{2^j} f(n,m) = \sum_{k=-\infty}^{+\infty}\sum_{l=-\infty}^{+\infty} \tilde{h}(2n-k)\tilde{g}(2m-l)\ \ A^d_{2^{j+1}} f(k,l)$$

A.1.2 Expression de $ID^1_{2^j} f$

On sait que $ID^1_{2^j} f = \left(\left\langle f(u,v), P_{V^1_{2^j}\otimes O^1_{2^j}} \Phi_{2^{j+1}}(u-2^{-j-1}n, v-2^{-j-1}m)\right\rangle\right)_{(n,m)\in Z^2}$

$P_{V^1_{2^j}\otimes O^1_{2^j}}$ correspondant à l'opérateur de projection de f sur le sous-espace vectoriel $V^1_{2^j}\otimes O^1_{2^j}$

Or $\Psi^1_{2^j}$ est une base de $V^1_{2^j}\otimes O^1_{2^j}$, d'où

$$\begin{aligned} &P_{V^1_{2^j}\otimes O^1_{2^j}} \Phi_{2^{j+1}}(u-2^{-j-1}n, v-2^{-j-1}m) \\ &= 2^{-2j}\sum_{k=-\infty}^{+\infty}\sum_{l=-\infty}^{+\infty} \left(\langle \Phi_{2^{j+1}}(u-2^{-j-1}n, v-2^{-j-1}m), \Psi^1_{2^j}(u-2^{-j}k, v-2^{-j}l)\rangle\right. \\ &\quad \left.\Psi^1_{2^j}(u-2^{-j}k, v-2^{-j}l)\right) \end{aligned}$$

A partir des formules de décomposition (A.1) et (A.2) on en déduit

$$\begin{aligned} &2^{-2j}\left\langle \Phi_{2^{j+1}}(u-2^{-j-1}n, v-2^{-j-1}m), \Psi^1_{2^j}(u-2^{-j}k, v-2^{-j}l)\right\rangle \\ &= 2^{-2j}\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} \Phi_{2^{j+1}}(u-2^{-j-1}n, v-2^{-j-1}m)\ \ \Psi^1_{2^j}(u-2^{-j}k, v-2^{-j}l)\ \ dudv \\ &= 2^{-2j}\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} \left(\phi_{2^{j+1}}(u-2^{-j-1}n)\phi_{2^{j+1}}(v-2^{-j-1}m)\right)\left(\phi_{2^j}(u-2^{-j}k)\psi_{2^j}(v-2^{-j}l)\right) dudv \end{aligned}$$

$$= \left(2\int_{-\infty}^{+\infty} 2^{-j-1}\phi_{2^j}(u-2^{-j}k)\phi_{2^{j+1}}(u-2^{-j-1}n)du\right)$$
$$\left(2\int_{-\infty}^{+\infty} 2^{-j-1}\psi_{2^j}(v-2^{-j}l)\phi_{2^{j+1}}(v-2^{-j-1}m)dv\right)$$
$$= 4\left(2^{-j-1}\left\langle\phi_{2^j}(u-2^{-j}k),\phi_{2^{j+1}}(u-2^{-j-1}n)\right\rangle\right)\left(2^{-j-1}\left\langle\psi_{2^j}(v-2^{-j}l),\phi_{2^{j+1}}(v-2^{-j-1}m)\right\rangle\right)$$
$$= 4\ h(n-2k)g(m-2l)$$

En effet on a

$$2^{-j-1}\left\langle\phi_{2^j}(u-2^{-j}k),\phi_{2^{j+1}}(u-2^{-j-1}n)\right\rangle = \langle\phi_{2^{-1}}(u),\phi(u-(n-2k))\rangle = h(n-2k)$$
$$2^{-j-1}\left\langle\psi_{2^j}(v-2^{-j}l),\phi_{2^{j+1}}(v-2^{-j-1}m)\right\rangle = \langle\psi_{2^{-1}}(v),\phi(v-(m-2l))\rangle = g(m-2l)$$

Donc

$$P_{V_{2^j}^1\otimes O_{2^j}^1}\Phi_{2^{j+1}}(u-2^{-j-1}n,v-2^{-j-1}m) = \sum_{k=-\infty}^{+\infty}\sum_{l=-\infty}^{+\infty} 4\ h(n-2k)g(m-2l)\ \ \Psi_{2^j}^1(u-2^{-j}k,v-2^{-j}l)$$

Et finalement on obtient

$$\begin{aligned} ID_{2^j}^1 f(n,m) &= \left\langle f(u,v), 4\sum_{k=-\infty}^{+\infty}\sum_{l=-\infty}^{+\infty} h(n-2k)g(m-2l)\ \ \Psi_{2^j}^1(u-2^{-j}k,v-2^{-j}l)\right\rangle \\ &= 4\sum_{k=-\infty}^{+\infty}\sum_{l=-\infty}^{+\infty} h(n-2k)g(m-2l)\ \ \langle f(u,v),\Psi_{2^j}^1(u-2^{-j}k,v-2^{-j}l)\rangle \end{aligned}$$

Or $D_{2^j}^1 f(k,l) = (\langle f(u,v),\Psi_{2^j}^1(u-2^{-j}k,v-2^{-j}l)\rangle)$

Donc

$$ID_{2^j}^1 f(n,m) = 4\sum_{k=-\infty}^{+\infty}\sum_{l=-\infty}^{+\infty} h(n-2k)g(m-2l)\ \ D_{2^j}^1 f(k,l)$$

A.2 Relation entre les coefficients du filtre H défini par Mallat et ceux du filtre H dérivé d'une ondelette à support compact

Soit $h'(n)$ la réponse impulsionnelle du filtre H définie par Mallat et $h(n)$ celle du filtre dérivé d'une ondelette à support compact.

On sait que $\forall n \in Z, h'(n) = \langle\phi_{2^{-1}}(u),\phi(u-n)\rangle$ et que $h(n) = \left\langle\phi(u),\sqrt{2}\phi(2u-n)\right\rangle$.

Donc

$$h'(n) = \frac{1}{2}\int_{-\infty}^{+\infty}\phi\left(\frac{u}{2}\right)\overline{\phi(u-n)}du$$

On pose $U = \frac{u}{2}$, d'où $dU = \frac{1}{2}du$.

Par conséquent on a

$$h'(n) = \int_{-\infty}^{+\infty}\phi(U)\overline{\phi(2U-n)}dU$$

$$
\begin{aligned}
h'(n) &= \frac{1}{\sqrt{2}} \left(\sqrt{2} \int_{-\infty}^{+\infty} \phi(u) \overline{\phi(2u-n)} du \right) \\
&= \frac{1}{\sqrt{2}} \left\langle \phi(u), \sqrt{2}\phi(2u-n) \right\rangle \\
&= \frac{1}{\sqrt{2}} h(n)
\end{aligned}
$$

Bibliographie

[1] M.L. Hilton, B. Jawerth & A. Sengupta *"Compressing Still and Moving Images with Wavelets"*, Manuscrit, 1994. http://www.mathsoft.com/wavelet.html

[2] J.N. Bradley & C.M. Brislawn *"The Wavelet/Scalar Quantization Compression Standard for Digital Fingerprint Images"*,
Tech. Rep. LA-UR-94-827, Proc. IEEE ISCAS-94, London. http://www.mathsoft.com/wavelet.html

[3] S.G. Mallat *"A Theory for Multiresolution Signal Decomposition : The Wavelet Representation"*, IEEE Trans. Pattern Anal. and Machine Intell., vol. 11, pp. 674-693, Juillet 1989.

[4] S.G. Mallat & S. Zhong, *"Characterization of Signals from Multiscale Edges"*, IEEE Trans. Pattern Anal. and Machine Intell., vol. 14, pp. 710-732, Juillet 1992.

[5] J.M. Shapiro, *"Embedded Image Coding Using Zerotrees of Wavelet Coefficients"*, IEEE Trans. on Signal Proc., vol. 41, pp. 3445-3462, Décembre 1993.

[6] A. Said & W.A. Pearlman, *"A New Fast and Efficient Image Codec Based on Set Partionning in Hierarchical Trees"*, Manuscrit soumis à IEEE Trans. on Circuits and Syst. for Video Tech., presenté au IEEE Int Symp. on Circuits and Syst., Chicago, Mai 1993. http://ipl.rpi.edu/SPIHT

[7] J. Bae & V.K. Prasanna, *"Synthesis of VLSI Architectures for Two-Dimensionnal Discrete Wavelet Transforms"*, Int. Conf. ASAP'95, Strasbourg, actes publiés par IEEE Comp. Soc. Press, pp. 174-181, Août 1995.

[8] J. Bae & V.K. Prasanna, *"A Fast and Area-Efficient VLSI Architecture for Embedded Image Coding"*, Manuscrit, Department of Electrical Engineering Systems, University of Southern California. http://www-scf.usc.edu/ jbae/icip95.ps

[9] P. Moravie, H. Essafi, D. Juvin, C. Lambert-Nebout & J-L. Basille, *"Algorithme Data-Parallèle de Compression d'Images"*, RenPar'7, Actes des $7^{èmes}$ Rencontres Francophones du Parallélisme, Mons, Belgique, 1995.

[10] J. Canny, *"A Computationnal Approach to Edge Detection"*, IEEE Trans. Pattern Anal. and Machine Intell., vol. 8, pp. 679-698, 1986.

[11] I.H. Witten, R.M. Neal & J.G. Cleary, *"Arithmetic coding for data compression"*, Commun. ACM, vol. 30, pp. 520-540, Juin 1987.

[12] Y. Meyer, *"Les Ondelettes - Algorithmes et Applications"*, Armand Colin, 1994.

[13] N. Baaziz & C. Labit, *"Transformations Pyramidales d'Images Numériques"*, Publication Interne $N°$ 526, IRISA, Mars 1990.

[14] I. Daubechies, *"Orthonormal bases of compactly supported wavelet"*, Commun. Pure Appl. Math., vol. 41, pp. 909-996, Novembre 1988.

[15] B. Jawerth & W. Sweldens, *"An Overview of Wavelet Based Multiresolution Analyses"*, Manuscrit, Departement of Mathematics, University of South Carolina, 1992. http://www.mathsoft.com/wavelet.html

[16] D. Esteban & C. Galand, *"Application of quadrature mirror filters to split band voice coding schemes"*, Proc. Int. Conf. Accous. Speech, Signal Processing, pp. 191-195, Mai 1977.

[17] M. Smith & T. Barnwell, *"Extract reconstruction techniques for three structural subband coders"*, IEEE Trans. Acous. Speech, Signal Processing, vol. ASSP-34, $N°$ 3, pp. 434-441, Juin 1986.

[18] H.Y.H. Chuang & L. Chen, *"VLSI Architecture for Fast 2D Discrete Orthonormal Wavelet Transform"*, Journal of VLSI Signal Processing, vol. 10, pp. 225-236, 1995.

[19] F. Charot, *"Architectures parallèles spécialisées pour le traitement d'image"*, Publication Interne $N°$ 722, IRISA, Avril 1993.

[20] J. Lu, *"Computation of 2-D Wavelet Transform on the Massively Parallel Computer for Image Processing"*, Technical Report, Thayer School of Engineering, Dartmouth College, Février 1991.

[21] Thinking Machine Corporation, *"Connection Machine CM-5 Technical Summary"*, The Connection Machine System, Novembre 1993.