Implémentation de la Transformée en Ondelettes par l'approche Lifting Scheme sur GPU

DJOFANG Jannesquin Royer
dirigé par
Dr. Mvogo Ngono Joseph

14 mars 2015

Résumé

Les supercalculateurs massivement parallèles sont de plus en plus performants ces dernières années, et permettent des gains de temps énormes dans les opérations arithmétiques. Les traitements de flux tels que les images, les vidéos, sonores et leur transmission en tant que signaux sont cependant sujets aux lenteurs extrêmes, selon la qualité du canal de transmission et du stockage. Cette lenteur influence aussi leur utilisation par diverses applications qui elles mêmes ont des contraintes spécifiques d'utilisation. Les innovations pour le calcul haute performance, dans les modèles de parallélisme et les architectures multi-coeurs, sont un domaine en pleine expansion d'un intérêt d'exploration significatif. Nous avons focalisé notre travail dans la mise en œuvre sur GPU, la transformée en ondelettes discrètes. Essentiellement, nous avons implémenté l'approche lifting scheme de certaines ondelettes particulières et les avons mises en application dans le cadre de la compression. Nous avons proposé pour la quantification, une version pseudo-parallèle de la quantification vectorielle, rendue efficace sur les architectures parallèles GPU mises à notre disposition, avec CUDA. Les résultats obtenus ont permis de noter des gains de temps énormes sur la version de quantification proposée en rapport avec les versions séquentielles existantes ou envisageables. Ces résultats pourraient ainsi permettre d'apporter de plus grandes performances à la chaîne de codage vidéo ou à la fusion d'images de télédétection qui est le domaine d'intérêt principal de nos recherches futures.

Mots clés : Image, signal, parallélisme, GPU, ondelettes, lifting scheme, compression, quantification vectorielle.

Abstract

The massively parallel supercomputers are increasingly powerful in recent years, and allow huge speedup in arithmetic operations. streams processing

such as images, videos and sound transmission as signals are however subject to extreme delays, depending on the quality of the transmission channel and storage. This slowness influence consequently, medias use by various applications which themselves are subject to use case constraints. Recent achievements in digital signal processing theory, coupled with innovations in parallelism and multi-core architectures are therefore a very interesting field to explore. We focused our work in the graphics processor unit GPU computing, which can withstand massive parallelism, well known techniques of these two areas as application of the results from the signal processing theory, namely the discrete wavelet transform .

Essentially, we have implemented Approaches Version facelift scheme of particular wavelet for later touch the most concrete application, namely compression. This process has given us in the quantification of phase to provide a pseudo-parallel version of vector quantization, which for us draw strength from parallel GPU architectures available to us, with CUDA. In this work, experiments were conducted and documented and discussed results. However, we recognize that the range of applications of our work ranges from simple signal processing to compression of video coding chain, through image fusion are essential tools in the field of remote sensing, in which we we record in our journey. **Key words**: Signal, image, wavelets, filters, lifting scheme, GPU, vector quantization, parallel programming, graphic hardware, compression.

Il n'est pas de vent favorable pour celui qui ne sait pas où il va. (Sénèque)

Le scientifique lira des centaines de livres au cours de sa vie et restera persuadé qu'il lui reste beaucoup à apprendre. Un religieux n'en lira qu'un et sera persuadé d'avoir tout compris.

Contents

List of Figures

List of Tables

Remerciements

L'ordre, toujours l'ordre, encore l'ordre.
Partant du yotta[1] *au yocto*[2]*, les inclusions vont jusqu'à l'infini.*
Mais qu'est-ce donc cet infini?
Chaque chose qui existe est sous un ordre.
Toute chose émane d'une autre, et rien n'est d'origine spontanée.
Beaucoup nous sont même encore incompréhensibles, malgré nos prouesses techniques.
Je reconnais que je ne suis qu'un rien dans un univers infini.

Jannesquin D.

Je tiens tout d'abord à remercier l'Éternel Dieu vivant, par qui j'ai eu le souffle nécessaire pour la rédaction de ce document, après moult phases d'alitement.

Je remercie tous les membres du staff administratif de l'académie internet, pour les bons moments passés en leur présence, accompagné de leurs conseils au sein du laboratoire d'informatique appliqué (LIA) de l'université de Douala. Merci pour la confiance qu'ils m'ont témoigné à tous égards.

Merci à tous les enseignants pour leur temps accordé, et les immenses potentialités qu'ils m'ont laissé voir être possible avec tous les enseignements reçus.

Un merci à ma famille qui par le soutien réconfortant de la chaleur familiale, n'a sans cesse procuré les résultats escomptés du fait de savoir que je suis aimé. Aussi, ce fut d'un secours incommensurable face à tous les maux et difficultés rencontrés dès le début de cette formation.

Je pense à tous mes camarades de la 5^e promotion du master IASIG pour l'empreinte multiculturelle sans pareille. Ce fut une toute première pour moi. Les blagues des uns et des autres à tout moment par skype, les conseils des aînés, le soutien que je bénéficiais de tous, sans compter l'ambiance émouvante de la riche phase de regroupement de notre formation. Je crois honnêtement que je ne serai pas le seul à laisser dessiner quelques traits de sourire à chaque fois que j'y repenserai.

A tous mes amis, chacun selon son rang, se reconnaissant dans ce travail, je dis encore merci.

1. 10^{24}
2. 10^{-24} en unités de mesures internationales

Liste des acronymes et définitions

DWT, **TOD**: Discrete wavelet transform: c'est l'abréviation anglaise de transformée en ondelettes discrète. en dimension 2, on écrit *2D-DWT TO2D*.
CUDA: compute unified Device Architecture
TFD: Transformée de Fourrier Discrète
GPU: General Purpose Unit
IASIG: Informatique appliqué aux systèmes d'information géographiques
cloud-computing:
data-mining:
CODEC: compression et décompression.
OTB [3]: Orfeo ToolBox. C'est un logiciel d'imagerie avancée, en sa version 4.2 pendant que nous rédigeons ce document (fevrier 2015).
Filtre: système servant à séparer des éléments dans un flux, suivant une logique particulière. En électronique, c'est un circuit qui réalise une opération de traitement du signal. En informatique, c'est un programme capable de traiter un ensemble d'informations pour en extraire un sous-ensemble d'informations pertinentes. Il peut être passif ou actif.
RADAR: Radio Detection And Ranging.

3. http://www.orfeo-toolbox.org

A propos du Master IASIG

En partenariat avec l'Agence Universitaire de la Francophonie (AUF), l'Université Paris Est Marne-la-vallée(UPEMLV), l'École Nationale des Sciences Géographiques(ENSG) de France, le Master IASIG(Informatique Appliquée aux Systèmes d'information Géographiques) est un parcours de formation mis sur pied par l'Université de Douala.

La formation IASIG comporte entre autres trois modules principaux que sont:

- Un aspect juridique sur les systèmes d'information en général et les SIG en particulier.
- Le second étant celui du génie logiciel et des WEB-SIG, permettant de posséder les compétences techniques nécessaires pour la conception, et la mise en œuvre des systèmes d'informations géographiques, et leur intégration dans toute sorte d'applications Web.
- Le troisième aspect quant-à lui, est celui du traitement d'image, de la télédétection optique et RADAR. Il a pour but de nous donner les compétences de faire prévaloir les outils informatiques dans les applications.

Introduction générale

La miniaturisation et l'avenue de dispositifs performants, ouvre la voie à de nombreux domaines d'application aux rangs desquels l'analyse du signal. Les applications dans la conception de systèmes d'imagerie sophistiqués de télédétection [4] allant de la simple surveillance, à la détection des cibles pour applications militaires, à la fusion d'images avec comme contrainte de performance la fourniture de résultats en temps réel. En médecine, l'imagerie scanner est un domaine d'application de ces systèmes où les contraintes de performances peuvent directement sauver des vies. Le besoin est alors crucial de fournir des moyens rapides, évolués, fiables et efficaces pour une amélioration nette de la qualité des résultats, bien évidement avec un compromis toujours d'actualité, à savoir celui du gain espace/temps.

Apparu pour faciliter le traitement massif et délocalisé de gigantesques volumes de données, le big data [5] (une évolution du data-mining) s'intègre désormais au cloud-computing, qui exige la production de résultats en temps réel lui aussi. Des algorithmes robustes sont proposés de manière fréquente et sans cesse évolutive, à la communauté scientifique, tirant parti des concepts du parallélisme. A ce titre, le FBI a normalisé l'utilisation des résultats sur les ondelettes de la théorie du signal, afin de faciliter la compression et l'accès en temps réel, aux images numérisées qui sont stockées dans des bases de données réparties.

La mise en place de systèmes fournissant l'information en temps réel implique la nécessité d'une analyse efficace de ces données. C'est le cas de l'analyse fonctionnelle qui est un domaine d'étude très applicatif des mathématiques, dont les résultats ont comme conséquence des prouesses dans les moyens de communication, et l'imagerie dans son ensemble. Dans ces aspects de la vie courante, toute information est modélisée sous forme de signal, et nécessite des moyens performants pour la transformation et le traitement de ces données, qui peuvent varier du simple bit à plusieurs Giga-octets, et devenir de ce pas très contraignantes.

Afin de pallier au problème de représentation et du stockage de l'information, les évolutions dans les recherches ont permis la mise sur pieds de techniques révolutionnaires de traitement numérique, aux rangs desquels les ondelettes (I. Daubechies en 1992, S. Mallat en 1998), pour l'approximation des signaux.

Une ondelette ψ_1 n'est rien d'autre qu'une fonction de $L^2(\mathbb{R})$ [6] qui est intégrable, avec une moyenne nulle qui oscille localement (on note donc $\int_{-\infty}^{+\infty} \psi_1(t)dt = 0$). Celle-ci peut être *dilatée*(ou contractée), et *translatée*. On utilisera pour ces deux grands cas de transformation, respectivement les symboles $e \in \mathbb{R}_+^*$ pour facteur

4. dans son acception la plus large, désigne l'acquisition d'informations sur un objet ou un phénomène, par l'intermédiaire d'un instrument de mesure n'ayant pas de contact avec l'objet étudié.

5. fouille de données sur les gigantesques et complexes volumes de données de source multiples, autonomes et sans cesse croissante. Cette science s'étend dans tous les domaines de l'ingénierie, tant les sciences physiques que biologiques

6. Espace des fonctions de carré sommable

d'échelle, et $u \in \mathbb{R}$. Une telle opération est généralement notée

$$\psi_{u,e}(t) = \frac{1}{\sqrt{e}}\psi_1\left(\frac{t-u}{e}\right) \tag{1}$$

C'est l'ensemble des fonctions $\psi_{u,e}$ qui composent la base d'ondelettes $\{\psi_{u,e}(t)\}_{u,e}$

Par conséquence, la transformée en ondelettes continue d'un signal f avec une ondelettes dilatée ou contractée et translatée, se note:

$$W_f(u,e) = \int_{-\infty}^{+\infty} f(t)\frac{1}{\sqrt{e}}\psi_1 * \left(\frac{t-u}{e}\right) dt = \langle f, \psi_{u,e}\rangle \tag{2}$$

Suite à une opération de TOD, il est possible de reconstruire le signal original f depuis la transformée $W_f(u,e)$ comme suit:

$$f(t) = \frac{1}{C_{\psi_1}}\int_0^{+\infty}\int_{-\infty}^{+\infty} W_f(u,e)\frac{1}{\sqrt{e}}\psi_1\left(\frac{t-u}{e}\right) du\frac{de}{e^2} \tag{3}$$

où

$$C_{\psi_1} = \int_0^{+\infty} \frac{|\psi_1(w)|^2}{w} dw \tag{4}$$

C_{ψ_1} doit être une constante finie, impliquant pour l'ondelette qu'elle doit être de moment nul. La formule de reconstruction de f donnée par l'équation (3), fait appel à des ondelettes correspondant à toutes les localisations temporelles et à tous les facteurs d'échelle. On parle de *transformée inverse*.

Des techniques telles l'analyse multirésolution, les bancs de filtres ou encore l'approche lifting scheme, permettent chacune de développer les ondelettes. Ainsi, pour un signal donné, il est souvent très intéressant de le décomposer suivant un ensemble fini d'ondelettes (en faisant par exemple varier la fonction d'échelle sous forme de puissance de 2 comme c'est souvent le cas). Dans ce cas, l'analyse multirésolution se définit comme une suite vectorielle de sous-espaces fermés de $L^2(\mathbb{R})$.

Les boîtes à outils ont été développées pour les ondelettes, dont certains bien incorporés dans les logiciels puissants de calcul tels *Matlab*, ou encore des bibliothèques évoluées telles OTB. Cependant, ces dernières ne tirent pas encore partie des architectures hétérogènes et environnements parallèles pour une meilleure implantation de leurs robustes algorithmes très approuvés par des experts à l'échelle internationales. Nous reprendrons donc particulièrement des algorithmes connus pour la plupart, en occurrence la transformée en ondelette, mais cette fois en procédant à l'usage de la version lifting scheme, en l'adaptant à une architecture parallèle pour implémentation. Nous faisons par la suite une adaptation pour l'intégration dans le processus de compression d'image. A cet effet, nous proposons une version parallélisable d'un algorithme très connu, à savoir celui de Lloyd, Buzo et Gray, dans sa version généralisée, pour la quantification vectorielle.

Nous organisons la présentation de ce travail comme suit:
Premièrement, la section 1 présentera les procédés d'acquisition, de modélisation et

de traitements d'images numérique. Ensuite la section 2 présentera un état de l'art généralisé, allant de la théorie des ondelettes, à celle de la compression, en passant par les architectures parallèles. Elle sera suivie de la section 3 consacrée à la description détaillée de notre procédé expérimental, de la présentation et de l'évaluation des performances de la qualité des résultats obtenus après implémentation. Enfin viendra une conclusion associée d'éventuelles perspectives.

1 Acquisition, représentation et transformation d'images

La pertinence de l'utilisation des images a été décrite à l'introduction dans la section précédente. Seulement, il existe une panoplie de types d'images, dépendant de ce qu'on voudrait en faire, c'est-à-dire du type d'applications visées. De la même façon, le type d'application donne une orientation sur comment procéder pour l'acquisition de ces images; Ce qui répond au dispositif à mettre en place et sous quelles contraintes. On distinguera ainsi les images radar, satellites, de scènes naturelles, de scanners médicaux, et ainsi de suite. Pour notre étude, nous nous limitons sans nuire à la généralité, aux images fixes, qui constituent une bonne base pour les constructions plus complexes à l'instar des vidéos (succession d'images dans le temps à une certaine fréquence), dans des dimensions k variées. Les images fixes (k-dimension) en niveaux de gris peuvent être modélisées comme une fonction k-dimensionnelle d'intensité lumineuse, ou d'amplitude d'un signal rétro-diffusé vers un capteur.

1.1 Acquisition

De façon générale, les images sont acquises par le procédé que nous décrivons ci-dessous. Une source d'énergie, illumine un objet d'une scène dans un plan. L'objet illuminé reflète une partie de cette énergie, qui peut être mesurée par un capteur, et dont les caractéristiques sont propres à l'objet, en fonction du type d'onde (car la lumière n'est en fait qu'une onde) émise par la source. La source d'énergie peut être électromagnétique telle le RADAR, l'infrarouge, ou les rayons X. L'énergie ainsi émise peut être mesurée par des capteurs, et transformée en image numérique. Les trois types de capteurs les plus connus: uniques, sous forme de barrettes vecteur, ou sous forme matricielles, tels que présentés sur la figure 1 ci-dessous de la section 1.1fig:capteur. L'énergie entrante est transformée en tension par une combinaison d'énergie électrique et d'un matériau sensible, suite au passage d'un filtre. Cela caractérise la réponse du capteur, qui émet une valeur numérique correspondante à la quantité d'énergie détectée.

La figure 2 ci-dessous de la section 1.1 représente une acquisition d'image depuis une scène naturelle. Elle traduit la transformation d'énergie depuis une source d'illumination, reflétée par un élément de la scène. Cependant, cette énergie peut aussi provenir d'autres éléments de la scène, autre que l'objet principal (reflectance). La première opération effectuée par le système d'imagerie en (c) est la collecte de l'énergie entrante, afin de la projeter dans le plan image. En général, les images issues de satellites sont sous forme de bandes, sur plusieurs canaux (R, V, B, IR) et nécessitent des prétraitements particuliers afin de faciliter l'usage par les logiciels courants. Nous supposerons donc que ce pré-traitement a été effectuée par la suite, pour une généralisation à tout type d'image.

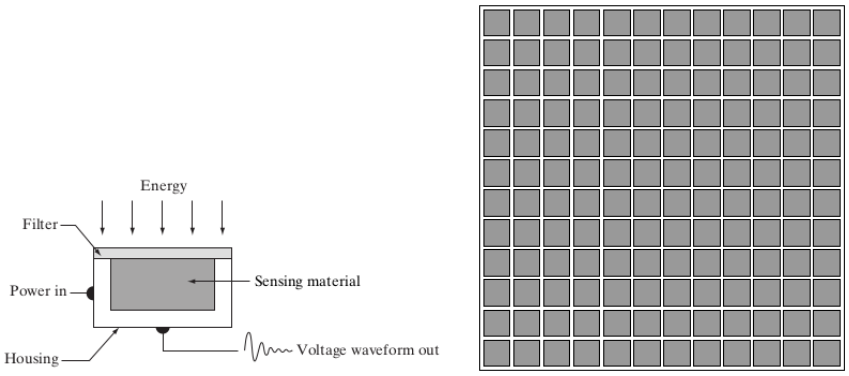

Figure 1: Un capteur unique et une matrice de capteurs ([3])

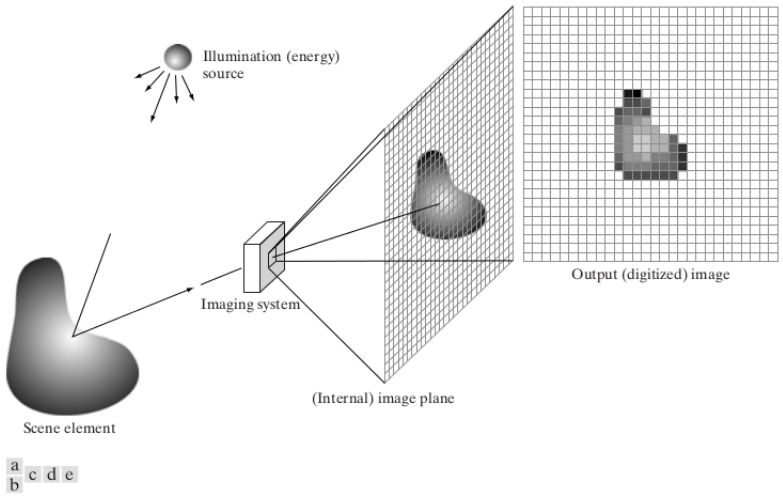

Figure 2: Un exemple d'acquisition d'image numérique. (a) source d'énergie (de lumière). (b) un élément de la scène. (c) système de prise d'image, capteur 2D. (d) projection de la scène dans le plan image. (e) image numérisée. ([3])

1.2 Représentation d'image numérique

Les processus d'acquisition d'image peuvent varier, selon qu'il s'agisse d'une photographie aérienne, d'une image radar à Synthèse d'ouverture, ou encore d'une composante d'images multispectrales. Dans cette section nous décrivons quelques traitements possibles sur les images acquises. Pour atteindre ce but, une bonne compréhension de la représentation et du stockage des images est importante, afin de les modéliser, pour répondre efficacement aux besoins pouvant se poser. C'est l'objet de cette section, à savoir donner les repères de modélisation d'une image quelconque.

Modélisation d'image

Une image afin d'être numérisée pour des traitements automatisés, passe par un processus d'échantillonnage et un procédé de quantification. On peut pour les représenter, utiliser plusieurs formalismes. Par exemple, les images obtenus de certains satellites, sont prises sous forme de trames (trame carrée, trame hexagonale, etc...) comme le présente la figure 3b et 3c. Supposons qu'une image $f(x, y)$ soit échantillonnée. Alors l'image résultante le plus souvent possède M lignes et N colonnes. L'image numérique est définie sur une grille à deux dimensions, dont es éléments sont appelés des pixels. Une image en k dimension est définie sur un ensemble $\{1, ..., M\} \times \{1, ..., N\}$. A chaque pixel s'attribue une couleur, définie le plus souvent dans le système RVB, ou en niveau de gris. Afin de simplifier les notations, nous définirons de manière générale par le formalisme mathématique qui s'y rapporte, une image comme une application f spécifiée comme suit:

$$\begin{array}{rcccl} f : \{1, ..., M\} & \times & \{1, ..., N\} & \longrightarrow & couleur \in \{0, ..., 2^l\}^k \\ & & (x, y) & \longmapsto & f(x, y) = (p_0, ..., p_k) \end{array}$$

Si le pixel est représenté sur l bits en machine, et que l'image est donnée sur k canaux (cas des images multispectrales, avec les canaux R,V,B,IR par exemple). Par convention sur $l = 8bitsaveck = 1canal$, si $f(x, y) = 0$, on a un pixel noir, et dans le cas où il vaut 255 on a un pixel blanc, et les valeurs intermédiaires donnent différentes teintes de gris. Nous prenons donc comme origine la coordonnée $(0, 0)$. Par analogie aux notions mathématiques acquises, la notion de repère est ainsi posée. Nous remarquons cependant, que ce repère représenté par cette figure de la section 1.2, est pris de telle façon que nous n'aurons que des coordonnées positives. On peut donc voir que la notation $(0, 1)$ représentera le deuxième échantillon de la première ligne. Pour des raisons de simplifications, l'image peut être vue en 2D comme une matrice sous la forme ci-dessous:

$$I = \begin{bmatrix} a_{0,0} & a_{0,1} & ... & a_{0,N-1} \\ a_{1,0} & a_{1,1} & ... & a_{1,N-1} \\ ... & ... & & ... \\ a_{M-1,0} & a_{M-1,1} & ... & a_{M-1,N-1} \end{bmatrix} \tag{5}$$

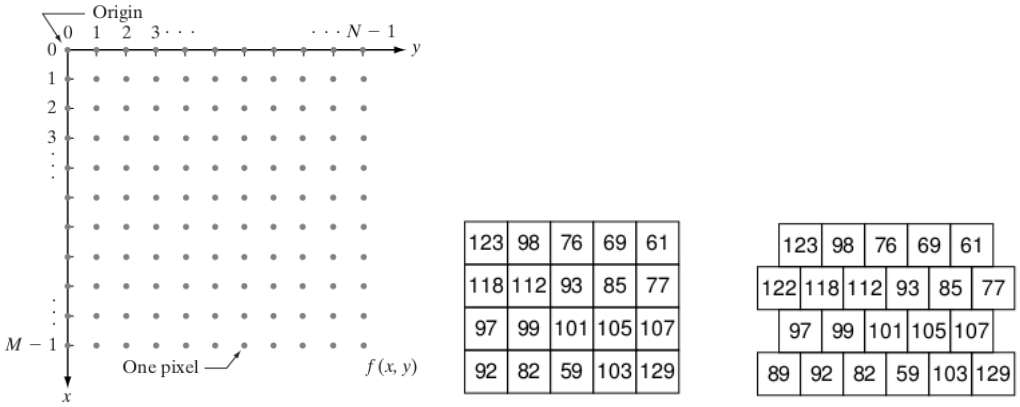

Figure 3: (a)convention du système de coordonnées pour la représentation d'une image([3]). (b) trame carrée. (c)trame hexagonale.

Il est cependant bon de remarquer que pour raisons de manipulation, toute matrice peut être mise sous forme d'un seul vecteur. On dira qu'on passe d'un signal 2D pour un signal 1D, et vice versa, et ces passages seront motivés par les techniques de traitement que l'on veut appliquer au dit signal.

Pour des raisons particulières, le processus de numérisation requière souvent des prises de décisions sur les valeurs de M et N, qui sont généralement préférées en puissance de 2. De même, en fonction du traitement à effectuer, la luminance d'un pixel (valeur réelle ou entière du pixel) peut avoir des exigences en terme de bornes. Ces préférences sont motivées par les opérations de traitement, le stockage, la quantification, les considérations matérielles, ou l'adaptation logicielle.

1.3 Transformations sur les images

Les applications en imagerie sont soumises à différentes contraintes, auxquelles doivent répondre les systèmes et logiciels mis en place. Un certain nombre d'opérations sont rendues possibles sur les images, certaines pouvant dégrader l'image originale, d'autres pas.

1.3.1 Filtrage

C'est un type d'opération de la classe des traitements radiométriques. En général, il s'agit de mécanismes au sens large d'opérations mathématiques, applicables sur les images, afin de produire des opérations par exemple de sélection, ou plus complexes encore, sur les images ou blocs d'images fournies en entrées. On distingue le filtrage global du filtrage local, en ce sens que chaque pixel de l'image résultante est calculé en prenant en compte la totalité ou le voisinage des pixels de l'image de départ. Nous distinguons de manière générale, les filtres convolutifs, les filtres dits morphologiques, les filtres heuristiques, et les filtres adaptatifs. Chaque filtre a pour but d'atténuer certains défauts précis, aucun d'eux n'étant capable à lui seul

de corriger tous les défauts. Il faut le plus souvent une combinaison de filtres pour un bon traitement.

Filtrage convolutif Cette technique fait usage des filtres de lissage, qui appliqués aux images, permettent de répartir de manière uniforme, la luminance de chaque pixel, et d'équilibrer en quelque sorte l'histogramme de l'image. On y retrouve le filtre moyenne et le filtre gradient.

Filtrage heuristique Cette méthode est appliquée pour des traitements spécifiques non linéaires. On y retrouve le filtre médian. Soit H un opérateur prenant en entrée deux images f et g. H est dit opérateur linéaire si, pour n'importe quelles images f et g, et pour deux scalaires a et b,

$$H(af + bg) = aH(f) + bH(g) \tag{6}$$

En d'autres mots, le résultat de l'application d'un opérateur linéaire à une somme de deux images (qui au préalable ont été multipliées par des constantes), est identique à l'application séparée des multiplications des scalaires aux images individuellement, suivies par l'application de l'opérateur. Par exemple, l'opérateur calculant la somme de K images est un opérateur linéaire. Celui qui produit la valeur absolue de la différence de deux images ne l'est pas. Tout opérateur qui ne vérifie pas l'équation précédente est par définition non linéaire.

Filtrage morphologique Ces méthodes constituent une atout dans un processus de segmentation. Il s'agit d'outils capables de niveler les images tout en préservant les contours importants. Cela simplifie l'opération de segmentation proprement dite. Dans certains cas, si le filtrage est important, il peut produire une partition pertinente.

Filtrage adaptatif Le filtrage adaptatif a la particularité de tenir compte les données statistiques locales sur l'image à traiter.

1.3.2 Compression

Étant donné que les images numériques sont composées des milliers de pixels occupant chacun un certain espace mémoire, la taille totale pour le stockage d'une image peut être significative. Si de plus nous sommes dans un cas d'observation de zone à l'exemple d'application de télédétection radar, stocker ces volumes d'images devient vraiment contraignant. Et que dire des traitements sur ces gigantesques volumes de données? Non seulement l'espace de stockage est un paramètre critique, mais également le temps de traitement et de transmission des images via les réseaux de communications augmente, les rendant presque inaccessibles et inexploitables.

Dans le but de réduire de façon significative la taille de stockage des images, on utilise un procédé de compression, dans lequel certaines informations peuvent être irréversiblement perdues, mais où la qualité de l'image est conservée et permet son

utilisation efficace par l'application appelante. Le principal problème est donc la mise en œuvre de procédés permettant de déterminer le degré d'acceptation d'une image compressée. A cela s'ajoute les principes pour déterminer quelles informations sont plus signifiantes que d'autres au sein d'une image, pour un meilleur rendu. La compression d'image est une opération consistant à produire une ou des séries d'images représentatives de l'image de départ, en vue d'un gain soit en temps de transmission, soit en terme d'espace de stockage. On parle de débit binaire, pour désigner le nombre de bits en moyenne, utilisé dans la représentation de l'image en machine. Il existe pour cela des techniques de compression variées, regroupées en deux grandes classes que sont la compression avec perte et la compression sans perte.

1.3.3 Traitement des bords

Selon l'opération à effectuer, le traitement des bords doit être pris en considération de façon spéciale. Pour par exemple appliquer un filtre à une image quelconque, ce dernier en fonction de la taille du filtre, ne pourra s'appliquer aux pixels qui sont sur la bordure de l'image. Que faire? La façon la plus simple est d'appliquer la politique de l'autruche (tête dans le sable): on ne fait rien du tout. Cependant, ceci peut en fonction du type d'image, s'avérer très mauvais pour la qualité de l'image attendue en sortie, car les pixels sont perdus à chacune de ce genre de transformation. La technique la plus utilisée, consiste à dupliquer les éléments de bordure pour permettre le traitement escompté.Ici, deux faons de faire sont employées:

- la continuité: Ici, pour un pixel manquant du voisinage d'un pixel considéré, sa valeur est celle du pixel le plus proche se trouvant dans l'image d'origine.
- Le miroir: Particulièrement ici, pour ne pas avoir les valeurs de pixel manquant dans le voisinage d'un pixel donné, on considère que l'image est projetée sur une sphère. c'est-à dire par exemple f(-1,y)= f(1,y);

Une autre technique est la mise à zéro, qui met à zéro, les pixels manquant ou inexistant dans le voisinage de la zone considérée.

2 État de l'art

2.1 Rapide État de l'art sur les ondelettes

théorie des ondelettes
Les ondelettes prennent leur origine dans la théorie du traitement du signal, alors qu'il fallait trouver des moyens et outils pour approximer des fonctions. Leurs racines proviennent de l'analyse fonctionnelle, principalement des notions avancées sur les espaces vectoriels et leurs propriétés. Jean Morlet et Alex Grossman en sont les introducteurs au début des années 1980. *Gabor* a par la suite dans ses travaux, décomposé un signal en fréquences sur plusieurs intervalles. Il a ainsi

	transformée en ondelettes	**transformee de Fourrier**
Base	ondelettes	sinusoïde
analyse fréquentielle	fréquence variable	fréquence variable
analyse temporelle	Limitée dans le temps	limitée en temps

Table 1: TOD vs TFD: différences notoires entre les deux approches

ramené le problème de représentation du signal, en la comparaison de ces intervalles, à plusieurs morceaux de courbes oscillantes de fréquences différentes. Ces morceaux de courbes sont en effet de petites ondes, communément appelées *ondelettes*, dont la taille est variable. Les transformées en ondelette se retrouvent dans le domaine tant continu que discret. Elles oeuvrent sur l'ensemble des décalages ou compressions possibles du signal considéré.

Les notions en analyse multirésolution par les travaux de *Mallat* [11] ont introduis de nouveaux outils puissants, permettant de relier les ondelettes orthogonales à des filtres miroirs. Par ce procédé, il est désormais possible d'approximer un signal φ par 2 éléments principaux: Une *fonction d'échelle* ψ obtenue par un *filtre passe-bas*, et les*détails* δ par un filtre passe-haut. Le filtre passe bas nous donnera le résumé du signal de manière macroscopique, et les détails dudit signal sont obtenus en sortie par le filtre passe-haut. Il est à rappeler que les filtres sont des éléments mathématiquement bien définis, donc ayant des propriétés établies. L'approche banc de filtres a de ce fait beaucoup été utilisée dans la téléphonie. Cependant, la littérature fait état de plusieurs types d'ondelettes, caractérisées par leurs bases respectives.

Types d'ondelettes

Les ondelettes sont très proches et comparables à l'analyse de fourrier très connue dans le domaine du traitement du signal. Alors que la transformée en ondelettes fait usage des ondelettes comme base de fonctions, la transformée de fourrier se sert plutôt des sinusoïdes comme fonctions de bases. Le tableau ci-dessous regroupe quelques caractéristiques de différenciation (Table.1). On y voit que la TFD donne les informations sur la décomposition fréquentielle des signaux, alors que la TOD distingue une résolution aussi bien dans le domaine fréquentiel que spatial. L'avantage des ondelettes sur la transformée de Fourier est qu'elles permettent une meilleure analyse des fonctions présentant des discontinuités ou des pics.

ondelettes de Haar La base des fonctions de la transformée de Haar est la plus vieille et la plus simpliste qui soit, connue des ondelettes orthonormées. Les ondelettes de Haar sont *séparables*, *symétriques*, et définies comme suit:

$$\varphi(x) = \begin{cases} 1 & pour \quad 0 \leq x < 1 \\ 0 & ailleurs \end{cases}$$

et la fonction d'échelle $\psi = \begin{cases} 1 & pour \quad 0 < x < 0.5 \\ -1 & pour \quad 0.5 < x < 1 \\ 0 & ailleurs \end{cases}$

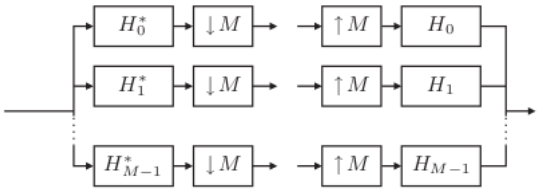

Figure 4: Un banc de filtres M-bandes

D'où on tire la relation entre la fonction d'ondelette et la fonction d'échelle de Haar: $\psi(x) = \varphi(2x) - \varphi(2x-1)$. Cette base est plus rapide en temps de calcul. Cependant, l'inconvénient est que l'ondelette associée n'est pas continue. La généralisation des fonctions de Haar produit les flatlets, qui sont des fonctions constantes dans l'intervalle $[0, 1]$

Ondelettes de Daubechies
la fonction d'ondelette se décompose ainsi, en fonction de la fonction d'échelle:
$\psi(x) = \frac{1-\sqrt{3}}{4}\varphi(2x+2) - \frac{3-\sqrt{3}}{4}\varphi(2x+1) + \frac{3+\sqrt{3}}{4}\varphi(2x) - \frac{1+\sqrt{3}}{4}\varphi(2x-1)$

2.1.1 Les Ondelettes par l'approche bancs de filtres

Les bancs de filtres sont une approche, voire même la plus connue pour le design des ondelettes. Un signal peut être réparti sur M bandes, on parlera dans ce cas de *transformée en ondelettes M-bandes* [6], (avec $M \geq 2$). La figure ci-dessous (fig. 2.1.1) illustre un banc de filtres M-bandes. Les bancs de filtres sont en général de 2 types: *Analyse* et *synthèse*.

- Les filtres d'analyse, décomposent un signal fournit en entrée, afin qu'il soit réparti sur M bandes.
- Les filtres de synthèse quant-à eux effectuent l'opération inverse des filtres d'analyse. Pour M bandes de signal fournis en entrée, ils sont recomposés afin de produire un unique signal en sortie.

Il est ensuite possible d'agencer en cascade plusieurs types de filtres M-bandes sur un certain nombre de niveaux k. On parle de l'analyse du signal en multirésolution sur k niveaux, lorsque les coefficients d'approximation du signal sont eux-même décomposés, k fois. La figure ci-dessous (fig. 2.1.1) illustre très bien ce procédé de décomposition en M bandes par banc de filtre sur 2 niveaux de résolution.

filtres orientables La littérature fait mention des filtres dits orientables. Ces types de filtres sont particulièrement très utiles en vision et traitement d'images, dans par exemple la détection de contours, et l'analyse de texture. Le procédé consiste à construire des filtres orientés suivant une orientation arbitraire; ces filtres orientés sont des combinaisons linéaires de filtres de base.

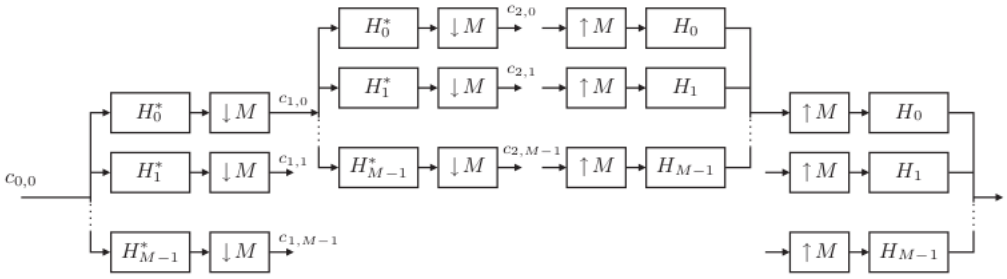

Figure 5: Multirésolution sur 2 niveaux avec banc de filtres M-bandes

dérivés des ondelettes Les ondelettes sont l'élément de base, qui ont permis des extension pour l'étude des structures géométriques. On parvient ainsi à mieux analyser la discontinuité dans le signal, surtout à partir de la dimension 2. C'est dans ce but que l'on note:

Les bandelettes : Elles permettent d'utiliser les contours d'une image lors de sa représentation. Elle se base sur une décomposition en ondelettes 2D et requiert une estimation locale explicite de la géométrie de l'image. La décomposition en bandelettes consiste en 3 étapes successives:

- Étape 1: il faut procéder à une segmentation de la région en une union de domaines plus restreints (carrés, rectangles, ...).
- Étape 2: On effectue une décomposition en ondelettes.
- Étape 3: il s'agit de la *bandelettisation*, qui permet de remplacer les fonctions d'échelles n'ayant pas de moment nul, par une base d'ondelettes orthonormale lorsque cela est nécessaire.

Les Ridgelets : Ils s'appuient sur une transformée de Radon et une transformée en ondelettes 1D.

Les curvelets : Les curvelets sont basées sur une transformée multi-échelles de type ridgelets combinée à un filtrage passe-bande. On procède en 4 principales phases:

- Une décomposition en sous-bandes de l'image
- Toutes les sous-bande sont individuellement scindés en partitions carrés dyadiques (on les multiplie par une fenêtre carrée).
- On normalise chaque partition.
- Les carrés sont analysés par une transformée en ridgelets.

Les contourlets : Elles se basent sur un banc de filtre directionnel pyramidal.

Ces dérivées des ondelettes sont particulièrement adaptées pour mieux prendre en compte les aspects géométriques présents dans une image, car elles s'adaptent mieux aux contours. Caroline Chaux [6] donne d'amples détails sur ce sujet.

2.1.2 Les ondelettes par l'approche Lifting-scheme

La notion d'ondelettes a été issue à la fois des communautés mathématiques de l'analyse, et de celles du traitement de signal. En analyse mathématique, elles sont définies comme les translation et dilatations d'une fonction fixe, et sont utilisées à la fois pour l'analyse et la représentation des fonctions. Plus tard, l'introduction de la théorie de l'analyse multirésolution et de la rapide transformée en ondelettes par Mallat et Meyer, fourniront une interconnexion entre les bancs de filtres et les ondelettes. Par la suite, les notions d'ondelettes semi-orthogonales ou bi-orthogonales(qui permettent la construction d'ondelettes symétrique) voient le jour et sont généralisées. C'est en 1994 que Wim Sweldens développe l'approche lifting scheme. L'idée de base derrière le lifting est qu'il fournit une simple relation entre toutes les analyses multirésolution qui partagent le même filtre passe haut ou passe bas. L'ondelette peut donc être vue comme une combinaison linéaire des fonctions d'échelle, où les coefficients sont donnés par le filtre passe haut. Le lifting possède plusieurs avantages, au nombre desquelles nous pouvons citer:

1. le lifting permet une transformation "in place", allégeant l'implémentation pour la transformée en ondelettes rapides. Ceci dit, pas besoin d'allouer un espace mémoire auxiliaire.

2. L'usage du lifting permet particulièrement de construire les transformée en ondelettes non linéaires: on peut par exemple faire les transformées entier-entiers. ceci est important pour les implémentations matérielles et pour le codage d'image.

3. Toute transformation faite avec le lifting est immédiatement inversible et la transformée inverse a exactement le même coût en complexité, que la transformée elle même.

4. Le lifting permet une adaptation de la transformée en ondelette.

5. Le lifting permet une construction des ondelettes sans faire usage de la transformée de Fourrier. Ceci signifie qu'il peut être utilisé pour construire les ondelettes qui ne sont pas nécessairement translatés ou dilatés d'une fonction(on parle d'ondelettes de seconde génération).

6. Le lifting est plus souple pour les non puristes mathématiciens, car ne fait pas appel aux notions de la transformée de Fourrier, et peut être de la sorte facilement introduite, uniquement par ses arguments dans le domaine spatial.

7. Enfin, le lifting expose le parallélisme inhérent de la transformée en ondelettes.Toutes les opérations d'un pas de lifting peuvent être faites totalement en parallèle.

structure de l'approche Lifting scheme
Le lifting scheme est un procédé de construction d'ondelettes, mieux efficace que l'approche par banc de filtres. Une de ses caractéristiques les plus importantes, est

qu'ils permettent pour tout banc de filtre basé sur le lifting, de satisfaire automatiquement la propriété de reconstruction parfaite [14]. Il se résume par la figure 2.1.2 ci-dessous. Il se décompose en 4 grandes étapes, telles qu'observable sur la figure ci-contre(2.1.2):

- Split: Cette étape scinde le signal supposé à l'initial de taille paire, en deux ensembles de coefficients, ceux aux index paires et ceux aux index impairs. On appelle cette phase la phase de transformée en *ondelette paresseuse*,connue sous l'appellation en terme anglais *lazy wavelet*.

- Predict: Étant donné une corrélation entre les coefficients obtenus par la phase ci-dessus, certains de ces coefficients peuvent être prédits en fonction des autres. l'opérateur de prédiction P est appliqué sur les coefficients pairs, et le résultat est soustrait des coefficients impairs, en vue d'obtenir la partie *détail* du signal.
$$d^{j+1} = odd^{j+1} - P(even^{j+1}) \tag{7}$$

- Update: l'opérateur de mise à jour U, est similaire au précédent, et, lorsqu'il est appliqué aux coefficients impairs et additionné aux coefficients pairs, il permet de mettre à jour les coefficientsc^{j+1}.
$$c^{j+1} = even^{j+1} + U(d^{j+1}) \tag{8}$$

- scale: Il s'agit d'un facteur k de mise à échelle, qui consiste à multiplier le signal obtenu en sortie, en vue de la normalisation. Les coefficients d'approximation c^{j+1} est multiplié par k et les détails d^{j+1} par $1/k$

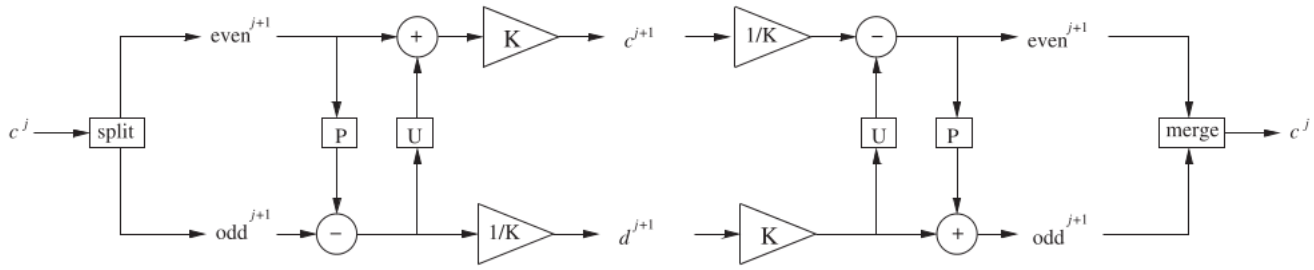

Figure 6: un niveau du lifting scheme classique. A gauche: phase forward du lifting. A droite: phase inverse du lifting. Ici, "split" est considéré comme une transformée en ondelette triviale (ou lazy wavelet)."merge" est l'opération inverse, P est l'étape de prediction, U l'étape de mise à jour, and K le facteur d'échelle (ou scaling factor).

Soit la figure 7 ci-dessous montrant la décomposition en ondelettes par bancs de filtres. La phase "forward" (transformée) fait usage de 2 filtres d'analyses $h^{\sim}$(passe bas) et $g^{\sim}$(passe haut), suivis d'un sous-échantillonnage de chaque sortie. Pour le processus de transformée inverse, il est d'abord question de faire un sur-échantillonnage

des signaux par les filtres de synthèses correspondants h(passe bas) et g(passe haut). Pour que la reconstruction soit parfaite, il est question que les transformées en z de chaque filtre vérifient le système des 2 équations suivantes.

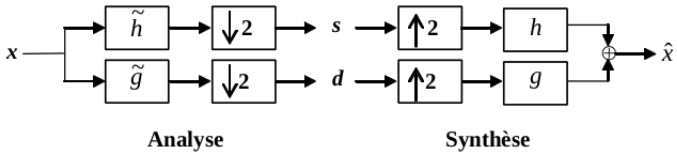

Figure 7: Transformée en ondelettes 1D sur un niveau.

$$h(z)h^{\sim}(z^{-1}) + g(z)g^{\sim}(z^{-1}) = 2 \tag{9}$$

$$h(z)h^{\sim}(-z^{-1}) + g(z)g^{\sim}(-z^{-1}) = 0 \tag{10}$$

On définit ainsi la matrice de modulation par $M(z) = \begin{bmatrix} h(z) & h(-z) \\ g(z) & g(-z) \end{bmatrix}$
Pour un filtre h, la représentation en polyphase est donnée par l'équation

$$h(z) = h_e(z^2) + z^{-1}h_o(z^2). \tag{11}$$

avec h_e le vecteur de coefficients paires, et h_o celui des coefficients impairs.

$$h_e(z) = \sum_k h_{2k}z^{-k} \; h_o(z) = \sum_k h_{2k+1}z^{-k} \tag{12}$$

Les puristes de la théorie font appel à la notion de matrice polyphase, pour désigner la matrice suivante: $P(z) = \begin{bmatrix} h_e(z) & g_e(z) \\ h_o(z) & h_o(z) \end{bmatrix}$

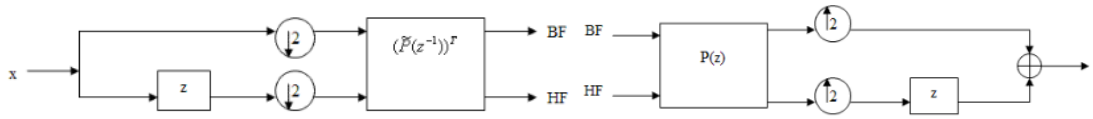

Figure 8: ondelettes avec la matrice polyphase

2.2 Un mot sur les paquets d'ondelettes

La transformée en paquets d'ondelettes mise au point par *Coifman* en 1992,[Coifman et al., 1992] est née de la volonté de s'adapter aux caractéristiques temps-fréquence des signaux. La méthode des paquets d'ondelettes est une généralisation de la décomposition en ondelettes, qui offre une analyse de signal plus riche. Un niveau de transformée est indexé par trois paramètres que sont la position, l'échelle, et la fréquence. Pour une fonction d'ondelette orthogonale, le procédé génère un ensemble de bases, appelées bases des paquets d'ondelettes. Chacune de ces bases offre une possibilité particulière d'encodage du signal, tout en préservant son énergie globale, et une reconstruction exacte.Il est contraire au principe général de la TOD classique qui ne permet les itérations que sur les sous-bandes passe-bas.

principe des paquets d'ondelettes La figure ci-dessous nous donne un aperçu de l'arbre généré par la décomposition en paquets d'ondelettes sur trois niveaux de décomposition. Plusieurs critères permettent de choisir la meilleure base, au nombre desquelles l'entropie d Shannon.

$$Entropie(S) = \sum_i {S_i}^2 \log {S_i}^2 \tag{13}$$

2.3 État de l'art sur les architectures many-coeurs

le Parallélisme fut depuis longtemps, réservé uniquement pour les spécialistes hautement qualifiés et très minutieux. En effet, la connaissance poussée pour mieux comprendre ce domaine ne s'acquérait que dans les hautes sphères telles que celles du calcul scientifique ou encore des télécommunications, avec cependant, seulement quelques experts à même de produire des programmes robustes, efficaces, et à grande échelle, tirant partie de ces architectures. Mais cela a subitement changé par l'apparition des processeurs multi-coeurs et many-coeurs. On possède désormais des plates-formes hétérogènes, composées à la fois d'unités de calcul classiques, et de processeurs graphiques dédiés. De plus, la génération actuelle de terminaux électroniques, tels les téléphones portables ou les lecteurs de musique propose désormais la possibilité de mise œuvre d'exploitation de leurs capacités en traitements parallèles, afin d'offrir des fonctionnalités au-delà de celles de leurs prédécesseurs.

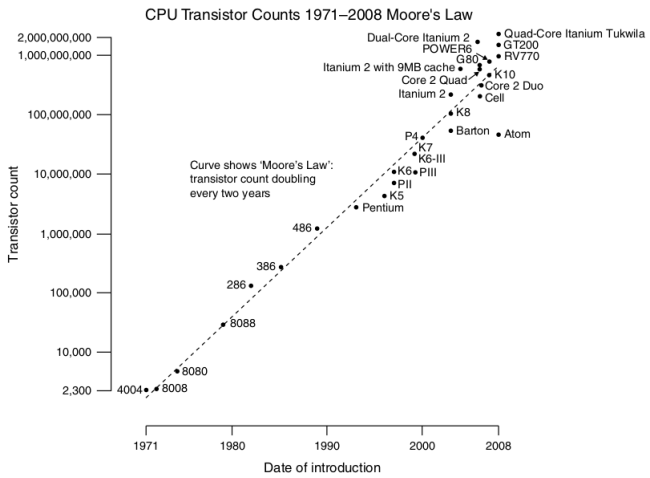

Figure 9: Loi de Moore. (Source: wikipedia)

La loi de Moore [7] prévoyait pour une évolution des performances des processeurs, que le nombre de transistors allait doubler tous les deux ans dans un circuit intégré(voir figure 2.3 ci-dessus). En conséquences, les appareils électroniques à processeurs sont devenus de plus en plus puissantes, à des coûts de plus en plus réduits.

7. Énoncée dans *Electronics Magazine* par Gordon Moore, en 1965 ingénieur de Fairchild Semiconductor, un des trois fondateurs d'Intel. Il la réajusta en 1975 en s'étendant au nombre de transistors des microprocesseurs et non plus de simples circuits intégrés.

Les processeurs ont gagné en puissance, en capacité, en vitesses, et en fréquence. Cependant, depuis 2005 la fréquence des processeurs semble stagner, du aux effets de dissipation thermique empêchant la croissance en fréquence des composants, malgré leur taille de plus en plus faible. Pour désormais gérer la puissance dissipée par la fréquence d'horloge des microprocesseurs, les constructeurs se retournent vers les architectures multi-coeurs. Le traitement parallèle devient donc désormais possible.

les GPU s'inscrivent dans le registre des architectures SIMD. En effet, la figure ci-dessous (fig xxx.) montre l'évolution de performance entre le CPU et le GPU, où on peut apercevoir le fait que le GPU est spécialisé dans le calcul haute performance, et équipé surtout pour les rendements 3D. Le GPU de Nvidia cuda vient avec un environnement logiciel permettant aux développeurs, d'utiliser aussi bien le langage C que d'autres langages connus tels Java, Fortran,C++, Python, OpenAcc. Il fournit des directives rendant possible un parallélisme assuré et simplifié, avec la manipulation des threads. Les processeurs graphiques sont donc équipés d'une technologie mettant en oeuvre un modèle facilitant l'exploitation de ces capacités qu'offrent les GPU.

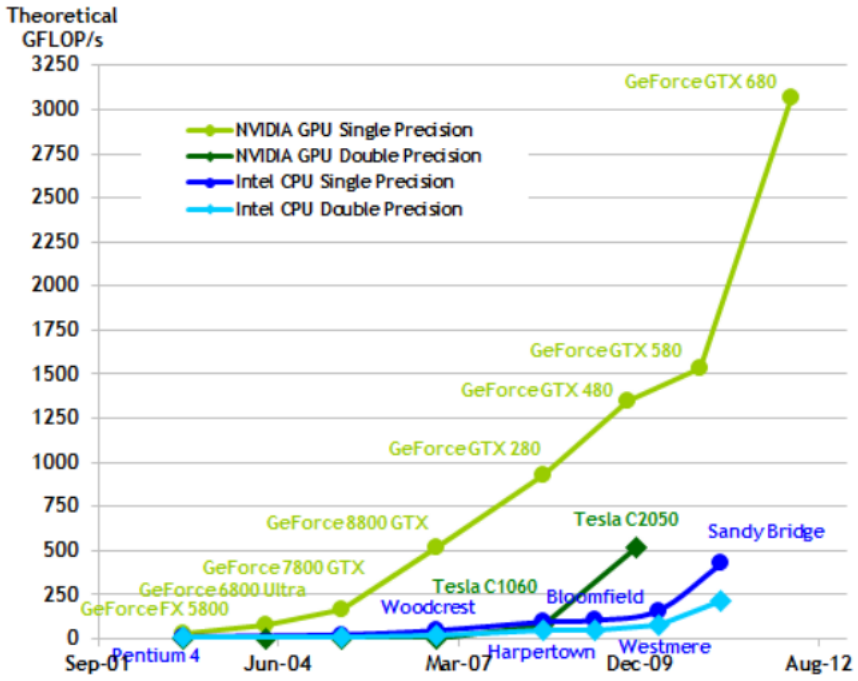

Figure 10: comparaison en terme de puissance, ente GPU et CPU. (Source: nvidia cuda c programming guide. octobre 2012)

Modèle de programmation

Dans le cadre de la programmation en C/C++, Cuda offre aux programmeurs

une interface pour la définition d'un type de fonctions particulières, nommées kernels. Les kernels permettent lors de leurs appels, d'effectuer un traitement qu'ils décrivent, en parallèle, par un nombre de threads. Les kernels sont définis par la signature **__global__ void** NomDuKernel $\lll \alpha, \beta, \gamma \ggg$(liste de paramètres) Avec:

α : représente la taille de la grille sur laquelle on lance le kernel. on spécifie ses paramètres en terme de dim3 ($dim_3\alpha = (x_1, y_1, z_1)$), afin de savoir si notre grille sera 1,2 ou 3D. Le produit $(x_1 * y_1 * z_1)$ détermine le nombre de blocs de threads qui seront créés au total.

β : représente la taille de chaque blocs, et par analogie au paramètre α, il définit également le nombre total de threads qui seront lancés dans un seul bloc.

γ : Ce paramètre est utilisé pour définir un espace de mémoire partagée, que pourront utiliser tous les threads d'un bloc, afin de se partager les données et éviter au maximum les accès en mémoire globale qui sont trop coûteux. Cet espace se retrouve aussi représenté dans le kernel par
[extern] [__shared__] type Nomvariable[taille].

Un kernel est organisé comme un ensemble de blocs de threads. Tous les threads d'un bloc peuvent coopérer entre eux, en se partageant des données, au travers de la *mémoire partagée*, et par là aussi, synchroniser leur exécution pour mieux gérer l'accès à la mémoire partagée. On utilise pour cela la primitive **__syncthreads()**. Chaque bloc s'exécute sur un multiprocesseur, et les multiprocesseurs ne communiquent pas par transfert de données. Donc la communication entre les threads est uniquement intra-multiprocesseur pour les threads lui appartenant. Chaque thread est identifié par un threadId, qui est local au bloc dans lequel il se trouve. Aussi, chaque bloc a son propre identifiant blockId en fonction de la grille sur laquelle il se trouve. étant donné que les dimensions peuvent varier de 1 à 3, ces identifiantrs sont donc d'un type de donné spécial appelé *dim3*.

La carte graphique GPU œuvre comme coprocesseur spécialisé pour les calculs, sur la machine hôte ou CPU. Il est bon de noter que les mémoires sont bien distinctes, l'une de l'autre, mais avec un mécanisme de copie dans les deux sens, ce qui toutefois est extrêmement coûteux, et doit être pris en compte lors de son utilisation. Le kernel est donc appelé depuis le CPU, et lancé comme une simple fonction multithreadée, dont le nombre de threads est spécifié, et le nombre d'unités de calcul de ces *threads* également. En effet, la figure xxx ci-dessous nous donne un schéma de l'organisation physique de la mémoire du GPU. La mémoire du GPU contient des grilles ou grid, qui sont de dimension 1,2,ou 3. Chaque cellule d'une grille définit ce que l'on appelle un bloc. un bloc lui aussi pouvant être de dimension 1 à 3. Chaque cellule d'un bloc définit une organisation de threads, qui peuvent être de dimension 1,2, ou 3. Cependant, lors du découpage, il est question de prendre en considération le fait selon lequel, les threads ne s'exécutent que par blocs de 32 threads appelés *warps*. Seuls les threads contenus dans un même bloc peuvent se synchroniser, par

les mécanismes de barrières [8] Ainsi, chaque thread ou chaque élément d'une grille, connaît sa position dans son enveloppe parente, car le thread est au bloc, ce que le bloc est à la grille.

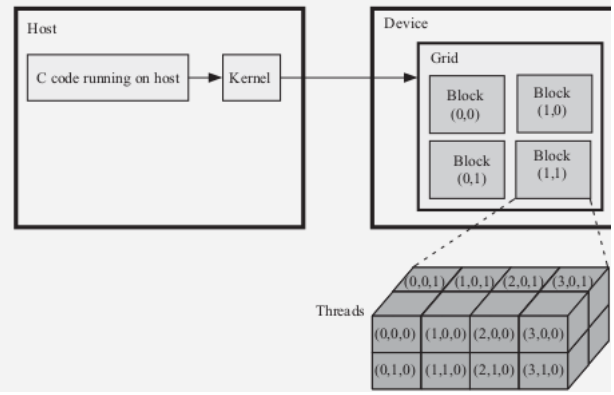

Figure 11: rapport entre grille, bloc et thread.

L'accès aux blocs des grilles (et par extensions aux threads des blocs selon le principe d'inclusion) se résume ci-dessous, selon la configuration choisie:

- Une grille 1D avec des blocs en 3D
  $UniqueBlockIndex = blockIdx.x; UniqueThreadIndex = blockIdx.x*blockDim.x* blockDim.y + threadIdx.y * blockDim.x + threadIdx.x;$

- Une grille 1D avec des blocs en 3D
  $UniqueBockIndex = blockIdx.x; UniqueThreadIndex = blockIdx.x*blockDim.x* blockDim.y*blockDim.z+threadIdx.z*blockDim.y*blockDim.x+threadIdx.y* blockDim.x + threadIdx.x;$

- Une grille 2D avec des blocs en 3D.
  $UniqueBockIndex = blockIdx.x; UniqueThreadIndex = blockIdx.x*blockDim.x* blockDim.y*blockDim.z+threadIdx.z*blockDim.y*blockDim.x+threadIdx.y* blockDim.x + threadIdx.x;$

- Une grille 2D avec des blocs 1D
  $UniqueBlockIndex = blockIdx.y*gridDim.x+blockIdx.x; UniqueThreadIndex = UniqueBlockIndex * blockDim.x + threadIdx.x;$

- Un grille 2D avec des blocs 2D
  $UniqueBlockIndex = blockIdx.y*gridDim.x+blockIdx.x; UniqueThreadIndex = UniqueBlockIndex * blockDim.y * blockDim.x + threadIdx.y * blockDim.x + threadIdx.x;$

8. Bien connu des programmeurs systèmes, dans la manipulation des mutex et l'appel implicite du "fork" pour la création de processus sous-jacents.

- Une grille 2D avec des blocs 3D
  $UniqueBlockIndex = blockIdx.y * gridDim.x + blockIdx.x; UniqueThreadIndex = UniqueBlockIndex * blockDim.z * blockDim.y * blockDim.x + threadIdx.z * blockDim.y * blockDim.x + threadIdx.y * blockDim.x + threadIdx.x;$

2.4 État de l'art sur la compression

Les applications cibles, à l'instar de l'imagerie médicale, la visiophonie, l'imagerie de télédétection et la télésurveillance, diligentent les méthodes CODEC employées, car exigent chacune des contraintes spécifiques parmi lesquelles:

La nature des images : Cette contrainte tient compte de ce que l'image peut être de nature fixe(une seule image), ou en séquences (vidéo) suivant une certaine cadence.

Utilisation des ressources : Elles mettent l'accent sur le débit de transmission de signal, et donc la capacité de stockage. On parle généralement de compression à contrainte de débit.

Fidélité des images reconstruites : Ici, on met l'accent sur les méthodes de reconstruction des images, préalablement compressées avec perte contrôlée ou sans perte. un critère de qualité pour jauger la fidélité est alors requis.

contraintes temps réel : Le temps d'exécution des processus de CODEC peuvent être asymétriques ou pas. Elles sont généralement symétriques dans le cas des applications en visiophonie et télésurveillance. Par contre elles ne le sont pas pour les applications nécessitant la consultation interactive des images (base de données biométriques).

Les normes les plus connues dans le domaine de la compression d'image, sont les normes JPEG et MPEG, qui utilisent les algorithmes adaptés aux scènes naturelles, et donc exigent une forte corrélation entre les pixels de l'image.

2.4.1 Formulation mathématique du problème de compression

Nguyen se place dans le cas d'un problème de compression à débit constant, de débit d'allocation global R_g bpp[9]. il énonce ainsi le problème comme suit:

Soit I un signal image. Il est question de coder I sous la contrainte R_g, pour obtenir $I^{'}$, de telle façon que la distorsion $D_{I,I'}$ soit minimale, et la qualité de l'image quant-à elle soit maximale.

9. qui représente ici la longueur binaire du signal

La Quantification la quantification est le procédé qui permet d'approximer un signal continu (ou à valeurs dans un ensemble discret de grande taille) par des valeurs d'un ensemble discret d'assez petite taille. Dans la chaîne de compression, c'est l'étape qui dégrade de façon irréversible le signal. Son importance se trouve dans la réduction du débit binaire. La quantification d'un signal peut se faire échantillon par échantillon, on parle de quantification scalaire (QS ou SQ). Elle peut aussi se faire par blocs d'échantillons, on parle de quantification vectorielle(QV ou VQ). Il est important de donner une définition au sens mathématique du terme, afin d'en avoir une compréhension au sens général.

Soit un signal S échantillonné $(X_1, ..., X_N) avec (X_i \in \mathbb{R}_n)$, c'est-à-dire $X_i = (X_i^0, ..., X_i^n)$. La quantification Q du signal S (S est représenté ici par le vecteur X), constitue le passage du continu au discret, car revient à projeter X dans un sous-ensemble de $\mathbb{R}^n$ dénombrable et borné.

$$\begin{array}{c} Q : \mathbb{R}^n \mapsto C \\ X \to Y \end{array} \tag{14}$$

où $n \in N$ et $C \subset R^n$. C est appelé *dictionnaire*, et est l'ensemble des valeurs possibles d'un échantillon du signal quantifié. Lorsque $n = 1$, on parle de *quantification scalaire*. Cependant, lorsque $n > 1$, on parle de quantification vectorielle. Dans la quantification scalaire, prends les éléments du signal, échantillon par échantillon, et on leur applique la fonction Q qui doit être définit. Rappelons toutefois qu'ici, l'image est au préalable partitionné [10] pour avoir des régions indépendantes, vérifiant:

- $\bigcup X_i = S$
- $\forall i, j, (i \neq j), X_i \cap X_j = \emptyset$

Nous notons aussi que, une représentation du signal initial par le signal S, n'est pas forcément exacte. il existe des représentation qui se trouvent êtres exactes, comme par exemple les représentations espace-fréquence des ondelettes et par paquet d'ondelettes. Cependant, on peut dans certains cas seuiller un certain nombre de coefficients issus de ces transformations de représentation, afin de filtrer le speckle [11].

La quantification vectorielle a prouvé ses meilleures performances vis à vis de la quantification scalaire, en terme de complexité, et de ce fait plusieurs méthodes de quantification permettant un codage avec coûts calculatoires moindres, dérivés de la QV généralisée, entre autres:

- QV multi-étages: on effectue successivement une QV grossière, suivie d'une deuxième portant sur l'erreur de quantification, et ainsi de suite.
- QV arborescente: on utilise ici une collection de dictionnaires composés chacun de deux vecteurs comme un arbre binaire. Sur un nœud donné il y a

10. plusieurs méthodes et critères de partitions telles les quadtree existent pour ce faire.
11. bruit observé sur les images. Mais le fait de les filtrer bien que consistant une opération irréversible, diffère de la quantification!

2 étapes: A la deuxième étape, on sélectionne le plus proche voisin dans le premier dictionnaire, et à la seconde étape on sélectionne le plus proche voisin dans l'un ou l'autre des deux dictionnaires en fonction du premier choix. La construction de ce dictionnaire demande un algorithme LBG réadapté, car il doit fournir tous les dictionnaires intermédiaires.Cependant, au décodage, un seul dictionnaire est utilisé.

- QV avec un automate à états finis.
- QV par produit cartésien: Ici, on ne peut exploiter la dépendance statistique pouvant exister entre toutes les composantes du vecteur. En effet, on décompose un vecteur en sous-vecteurs de dimensions éventuellement différentes, et on applique une quantification vectorielle pour chaque sous-vecteur. La complexité est proportionnelle à la somme des vecteurs constituant le dictionnaire.
- QV algébrique: Ici, le dictionnaire ne s'obtient plus par l'algorithme LBG. il est fait de telle façon à être indépendant des propriétés statistiques de la source. On répartit les vecteurs de reproduction de façon régulière dans l'espace. les dictionnaires n'ont pas besoin d'être mémorisés, car ils sont largement utilisés et permettent de réduire très fortement la charge de calcul. On parle encore de quantification vectorielle sur réseaux.

2.4.2 Taxinomie des critères de qualité

Le CNES [12] [4] définit la qualité de l'image de manière générale, comme relative aux besoins des applications en aval. Les critères de qualités qualifient toute la chaîne depuis la scène observée, la prise en compte de l'instrumentation (optique, chaîne électronique), les traitements de bords(quantification, égalisation, compression), mais aussi les traitements sols effectués avant la distribution de l'image. C'est une notion assez complexe à cerner. Pour la compression d'une image, surtout dans le cas de compression avec perte, l'utilisation d'une mesure de qualité est indispensable pour l'évaluation de performances. Deux grands types de méthodes sont donc souvent employées, à savoir les méthodes subjectives(basées sur l'évaluation de qualité par les observateurs *photo-interprètes* humains) et les méthodes objectives(basées sur les critères mathématiques). Les premières se prêtent plus aux images satellites, et les secondes aux autres types d'image. Dans le cas des méthodes objectives, on retrouve trois sous-catégories, selon que l'on a accès ou pas à l'image originale: les méthodes *bivariantes* (plus efficaces, et exigent l'accès à l'image originale), les méthodes *uni-variantes* (se basent sur une estimations des critères de qualités lorsque l'on ne possède que l'image finale), et les méthodes *intermédiaires* (Ici en plus de transmettre l'image codée, on transmet aussi certains paramètres de l'image originale, tels la moyenne, la variance, ...).On a plusieurs critères, à l'exemple du rapport signal/bruit (PSNR ou RSB), l'erreur quadratique moyenne(EQM ou MSE), et bien d'autres décris plus en détails dans [7].

12. Centre National des Études Spatiales

Les ondelettes sont des outils mathématiques pour la décomposition hiérarchique de fonctions. Il est question ici de nous intéresser à la représentation d'un signal mono-dimensionnel ou bidimensionnel. L'objectif visé est de le décrire à l'aide des ondelettes, c'est-à-dire de telle façon qu'il soit décomposé en deux grands blocs d'information: un bloc qui résume les informations sommaire sur le signal, et des blocs donnant les détails sur le signal. Une telle décomposition est faite dans le but d'avoir des éléments moins complexes à traiter et faciliter de ce pas l'usage et l'interprétation du signal. Il existe cependant plusieurs moyens d'y parvenir, au nombre desquels nous retenons pour notre étude l'approche lifting scheme, les raisons nous poussant à ce choix étant énumérées plus haut dans l'état de l'art qui a été faite des ondelettes. Avec les exigences du HPC [13] exigeant une puissance de calcul sans cesse évolutive, plusieurs paradigmes et modèles de parallélisme ont vu le jour, sous la concurrence des différents constructeurs. Les contextes applicatifs du signal et en particulier de l'image, prennent donc une nouvelle tournure afin de tirer au mieux partie de ces nouvelles capacités technologiques. On arrive alors à un stade où la plupart des simples algorithmes séquentiels existants pour toute catégorie de problèmes, tendent à être parallélisés. Notre cadre d'étude concernera le parallélisme des images fixes, utilisées généralement dans le bases de données de systèmes d'information géographiques distribués. Ce type de base de données pose des contraintes majeurs en terme de capacités de stockage et de débits de transmission qui doivent être très élevés. De ce fait, les images radar RSO sont celles qui conviennent le mieux, mais nous nous limiterons dans le cadre de notre étude à un cas particulier, sans nuire à la généralité.

Notre objectif principal étant donc de concilier les expériences dans ces trois domaines que sont l'imagerie, le parallélisme, et la compression, nous avons procédé à des séances d'expérimentations dont nous ferons une succincte description des procédés, et une présentation des résultats, dans le chapitre suivant.

3 Expérimentations

Le procédé expérimental que nous avons suivi consiste en deux grandes phases. La première consiste en la mise en place d'un environnement permettant l'extraction d'images depuis un fichier sous un format en niveau de gris pgm, afin d'en effectuer la transformée en ondelettes 2D par lifting scheme, sur un nombre de niveau spécifié. Pour cette cause, notre image est considérée selon le formalisme décrit dans la section 1. Pour des raisons d'efficacité dans la transformée en ondelettes, notre image sera lue et stockée dans un vecteur, qui contiendra les lignes de notre image, les unes suivies des autres, comme si l'on parcourait notre image selon que le décrit la figure ci-dessous.

Afin de permettre l'exécution de la 2D DWT, nous avons procédé comme suit. Dans un premier temps, chaque ondelette pour s'exécuter requiert le passage par

13. calcul haute performance

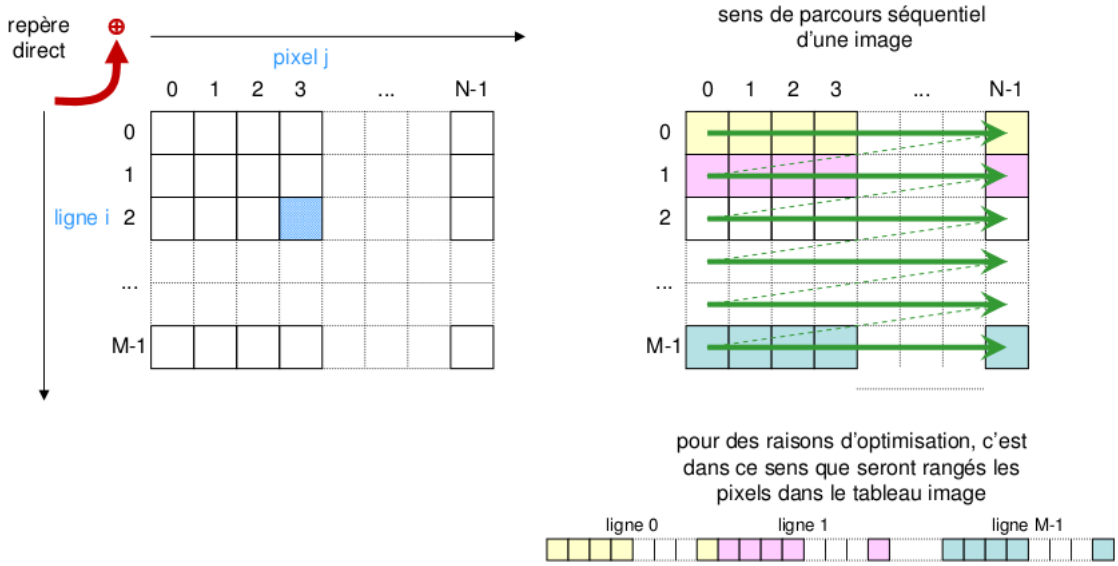

Figure 12: convention sur l'image

Standard	Non standard

procedure Décomposition1(C:Vecteur[1..M, 1..N]réels)
-normaliser les coefficients de C
Pour chaque niveau de décomposition h faire
pour chaque ligne *row*, faire
DWT(row)//vecteur de N éléments
fpour
fpour
Pour chaque niveau de décomposition h faire
pour chaque colonne *col* faire
DWT(col)//vecteur de M éléments
fpour
fpour
fin
	procedure Décomposition2(C:Vecteur[1..M, 1..N]réels)
	-normaliser les coefficients de C
	Pour chaque niveau de décomposition h faire
	pour chaque ligne *row*, faire
	DWT(row)//vecteur de N éléments
	fpour
	pour chaque colonne *col* faire
	DWT(col)//vecteur de M éléments
	fpour
	fpour
	fin

Table 2: Les deux procédés possibles pour la DWT en 2D.

une transformée en ondelette 1D pour chacune des lignes, puis pour chacune de ses colonnes. Deux approches dont les algorithmes de ces deux approches sont consignées dans le tableau ci-dessous s'offrent alors à nous:

1. Pour chaque niveau de la transformée, effectuer successivement une transformée sur les lignes et une transformée 1D DWT sur les colonnes, avant de passer au niveau suivant. C'est la décomposition non standard. Concernant les ondelettes dites *séparables*[14], une observation est visible sur la figure 14 de la section 3.2.

2. La décomposition standard consiste à effectuer pour tous les niveaux, une transformée 1D DWT sur toutes les lignes. Ensuite faire de même sur toutes les colonnes. On obtient alors pour une image quelconque, ce qui suit.

14. ce sont les ondelettes pour lesquelles il est possible de séparer les opération en terme de passage horizontal et vertical

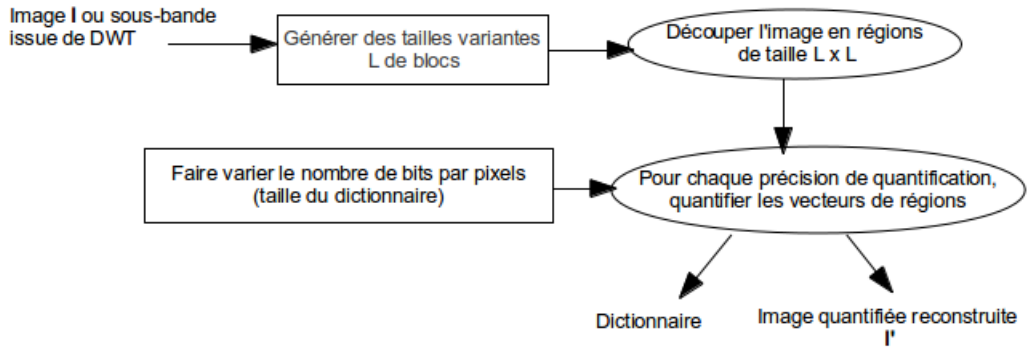

Figure 13: schéma des expérimentations que nous avons faites.

3.1 Architecture d'expérimentation

- Nos tests ont été effectués en C/C++ sur une machine Asus équipée d'un processeur CPU Intel Cores .... et dotée d'une carte graphique Nvidia GT520M ayant 1Go de mémoire globale. La carte est constituée de 6 multiprocesseurs ayant chacun 8 cœurs, soit un total de 48 cœurs CUDA. La version Nvidia installée est CUDA 6.5, tournant sur une architecture Windows 7, version 64bits. Toutefois, la version séquentielles de nos algorithmes ont été implémentées sans tenir en compte le nombre de CPU, donc étaient purement séquentielles. Les versions parallèles proposées ont de ce fait été adaptées à cette configuration spécifique de la carte graphique. Nous rappelons toutefois que plus il y a de cœurs CUDA, plus les performances donnent satisfaction.

3.2 Description de notre approche parallèle de la DWT

Ondelettes implémentées en version lifting
Plusieurs types d'ondelettes ont leur version en lifting. Ce sont ces dernières que nous avons implémenté dans notre procédé d'expérimentation, et en parallèle. Nous nous sommes donc attardés dans le cadre de ce mémoire, sur quelques unes des versions lifting d'ondelettes, à savoir: *Haar*, *LeGall 5/3*, et *Daubechies9/7*, *Deslaurier Dubuch*. Leurs formes correspondant à une étape de lifting scheme pour un signal $x = x_k$ sont proposées ci-dessous, bien évidemment, en supposant tel que c'est bien le cas, la commune phase *split* déjà effectuée.partant de l'étape $j-1$ avec comme signal S_{j-1}, et comme détail D_{j-1}. on évolue vers l'étape j en effectuant les opérations suivantes, selon le type d'ondelette. Les étapes de Prédiction et de mise à jour (resp. *predict* et *update*) se font l'une après l'autre, dans l'ordre donné.

Les ondelettes de Daubechies9/7 en version lifting scheme, contiennent deux phases de lifting successives, alternant chacune prédiction et mise à jour. Ensuite vient une étape de mise à échelle (*scaling*) On note les valeurs des paramètres dans le tableau ci-dessous.

wavelet	**operations**
Haar	$Predict : d_j[n] = \frac{\sqrt{2}}{2}(d_j[n] - s_j[n])$ $Update : s_j[n] = \sqrt{2}(s_j[n] + d_j[n])$
LeGall5/3	$Predict : d_j[n] = d_j[n] - \frac{1}{2}(s_j[n] - s_j[n+1])$ $Update : s_j[n] = s_j[n] + \frac{1}{4}(d_j[n-1] + d_j[n])$
DeslaurierDubuc	$Predict : d_j[n] =$ $Update : s_j[n] =$
Daubechies	$Predict1 : d_j[n] = d_j[n] + \alpha(s_j[n] + s_j[n+1])$ $Update1 : s_j[n] = s_j[n] + \beta(d_j[n-1] + d_j[n])$ $Predict2 : d_j[n] = d_j[n] + \gamma(s_j[n] + s_j[n+1])$ $Update2 : s_j[n] = s_j[n] + \delta(d_j[n-1] + d_j[n])$ $Scaling : s_j[n] = \zeta s_j[n]$ $d_j[n] = \zeta d_j[n]$

Table 3: Versions lifting scheme des ondelettes implémentées.

parametre	*valeur*
α	-1.586134342
β	-0.05298011854
γ	0.8829110762
δ	0.4435068522
ζ	1.149604398

Table 4: paramètres pour lifting scheme ondelettes de Daubechies9/7

Méthode de parallélisation sur CUDA

Des approches pour l'implémentation parallèle des ondelettes version lifting scheme au moyen des architectures parallèles *MPI*[15] sur des many-cores ont été proposées [citer ICI]. Mais étant donné les différences avec l'architecture de CUDA, où chaque processeur a en plus sa mémoire cache, même si cela n'est que de quelques kilo octets. Or, naît alors le problème de synchronisation entre les processeurs, qui mérite d'être prise au sérieux. La solution consiste à scinder les données de telles façon à effectuer un parallélisme sur ces dernières, et agencer les opérations pour les répartir convenablement sur les différentes unités de calcul ou device[16]. Les données doivent pouvoir être traitées de manière totalement indépendante, pour prétendre à un parallélisme efficient. plus encore, sur Cuda, elles doivent tenir en compte le principe de haute priorité suivant: Ne jamais mettre sur des unités de calculs différentes, ou blocs, des opérations de synchronisation de leurs données, car il ne peut y avoir synchronisation que par blocs, c'est-à-dire par des threads d'un même blocs. Cette tache n'est pas triviale, pour l'implémentation des versions lifting des ondelettes.

15. Message passing interface

16. terminologie anglaise couramment utilisée dans le parallélisme et donc par les "maniacs" de Nvidia Cuda, pour désigner le processeur graphique, contrairement à *host* pour désigner le CPU ou allocateur.

Une approche pourrait être de dupliquer les données en partie, sur chaque processeur. Cette solution est efficace, mais vétuste dès lors que le jeu de données devient important. Encore plus, on se retrouve coincé si l'on compte tirer parti de la mémoire partagée. On en conclut donc que ce n'est pas la meilleure solution. En remarque, nous notons que, étant donné que l'invocation de kernels consomme en effet un temps considérable, effectuer le travail que possible dans un seul thread serai la meilleure chose à faire.Ainsi, nous minimiserions le temps d'occupation du GPU et par conséquence, gagnerons en temps d'exécution de ce qui pour nous est l'objectif à atteindre. Pour ce faire, nous nous sommes tournés pour adopter la démarche effectuée en 2011 par Wladimir J., membre senior de IEEE, dans son article *Accelerating wavelet lifting on graphics hardware using cuda.*

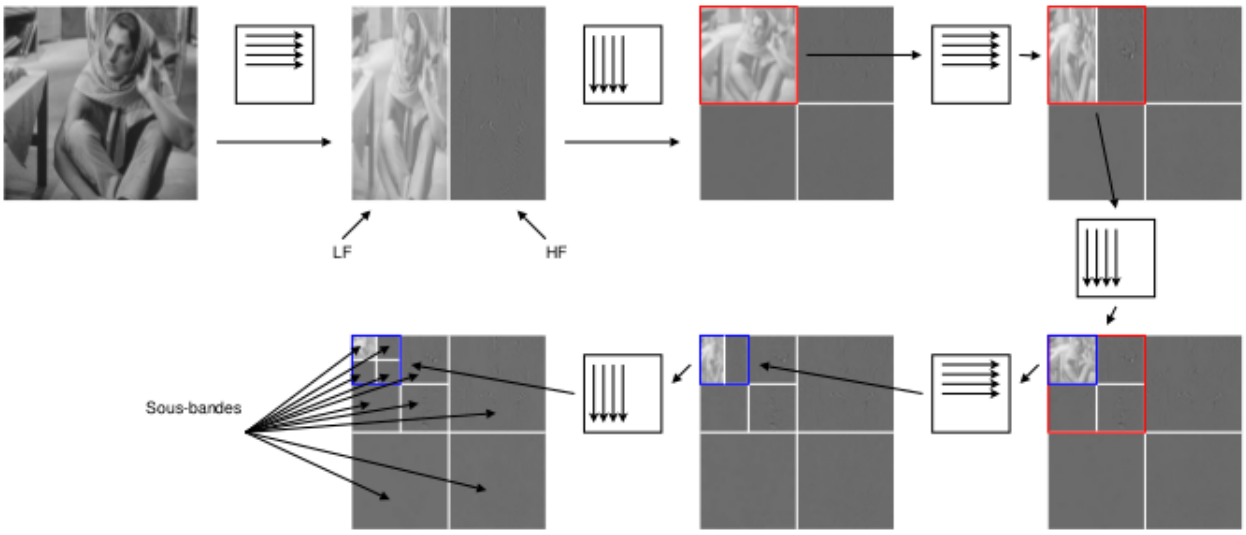

Figure 14: Décomposition multirésolution classique sur 3 niveaux: on décompose successivement suivant les lignes et les colonnes. LF est la partie basse fréquence et HF la partie haute fréquence.

Les coefficients de la zone grise représentent les valeurs proches de zéro, tandis que les noirs et blancs correspondent respectivement aux valeurs fortement négatives ou positives.

Algorithme simplifié

- `Faire la lecture du fichier image pour le mettre dans un buffer en mémoire. Nous avons opté pour le format .pgm en raison de sa portabilité et de sa simplicité de codage.`
- `L'image mise en buffer, sera ensuite copiée dans la mémoire globale.`

- `Appliquer à chaque ligne de l'image, la 1D-DWT`
- `au résultat de la phase précédente, appliquer la transposition.`
- `Répéter les 2 opérations précédentes.`
- `Recopier le résultat obtenu, de la mémoire globale pour son stockage dans le buffer principal, pour sauvegarde de résultat.`

3.2.1 Passage horizontal

Nous le résumons en ces 5 points ci-dessous:

- Lecture d'une ligne de la mémoire globale du GPU, pour la placer en zone de mémoire partagée.
- Traiter les cas de bordures. Selon le filtre appliqué, nous dupliquons les éléments de bordures.
- Application d'une étape du lifting scheme 1D sur les éléments se trouvant en mémoire partagée.
- Répéter les étapes 2 et 3 pour chaque pas de lifting à effectuer selon la transformée.
- Réécrire les résultats de la ligne d'image concernée, de la mémoire partagée pour la mémoire globale.

3.2.2 Passage vertical

Le passage en lifting vertical est trivial, une fois que le passage horizontal a été faite. Seulement, vu le procédé de stockage de notre image en un seul vecteur lignes après lignes, il est impératif d'employer un algorithme de transposition.Dans ce cas, l'image a besoin d'être transposé deux fois. En effet, après le passage horizontal, il faudrait dans un premier temps s'assurer d'un regroupement des coefficients de manière désentrelacée pour chaque ligne de l'image. Une fois ceci fait, procéder à une première transposition, avant d'appliquer le même kernel employé pour le lifting horizontal ci-dessus. Par la suite, se rassurer que les coefficients de la transformée après le lifting vertical sont rangés de manière désentrelacée, puis appliquer de nouveau la transposition pour avoir l'image dans son sens initial. Cette façon de procéder exige donc de manière résumée:

- Un kernel pour le lifting scheme sur un vecteur 1D, dans lequel on se rassure que les coefficients sont rangés en désentrelacés. On y fera appel à deux reprises.
- Un kernel efficace pour la transposée de la matrice, qui sera appelé immédiatement après l'appel au kernel.

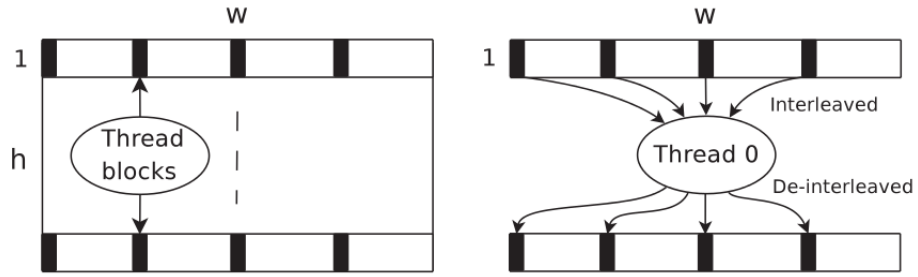

Figure 15: une étape parallèle du lifting horizontal. Je découpe l'image en h thread blocs, chaque bloc contenant T threads. Ainsi, chaque thread traitera $N/(hT) = w/T$ données. Les cases noires illustrent les coefficients d'entrée pour le thread0. w et h sont les dimensions de l'image ayant $Ncoefficients$ entrée. $N = w.h$

Cependant Wladimir J. décrit une version meilleure [10] que celle que nous venons de décrire dans le paragraphe précédent pour remplacer cette version du lifting vertical. Il désigne cette technique par le mot anglais "slidding windows".

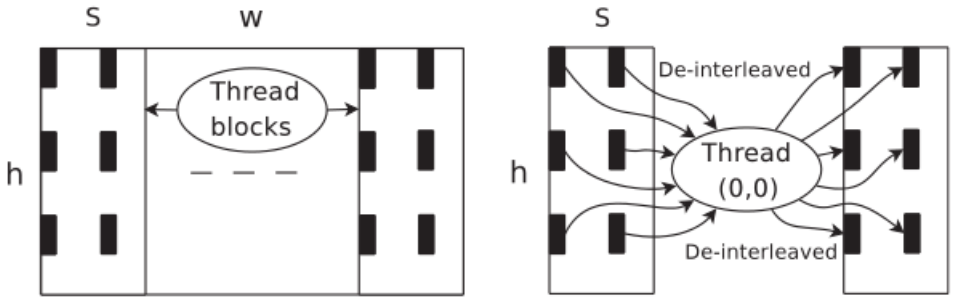

Figure 16: Passage du lifting vertical en slidding windows. Chaque thread d'un bloc traite une petite fenêtre Vs*Vy.

slidding windows

La méthode de la fenêtre coulissante ou "slidding windows" est la suivante. Afin de gagner en temps après un ...

3.2.3 Traitement des bords

Pour une efficacité dans la prise en compte des bords de l'image, nous avons opté pour la duplication en miroir des bordures de l'image, pour chaque ligne. Ceci garanti une meilleure prise en compte de la cohérence entre pixels voisins, pour le traitement des bordures.

3.2.4 Quantification vectorielle

La quantification vectorielle consiste a représenter tout vecteur x de dimension k, par un vecteur y de dimension analogue appartenant à un ensemble fini appelé dictio-

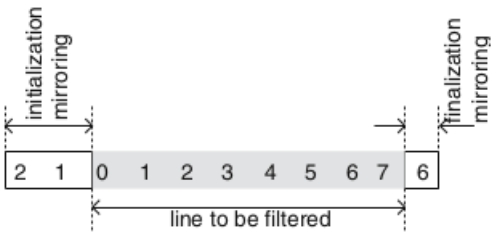

Figure 17: exemple de traitement des bords sur une ligne d'image, pour l'ondelette de LeGall5/3.

nnaire. Lors du codage de la quantification, l'étape primordiale est dans l'élaboration du dictionnaire, dont la création se fait au moyen d'une séquence d'apprentissage. L'une des techniques les plus connues et employées pour parvenir à la construction de ce dictionnaire, est l'utilisation de l'algorithme LBG [2]. Un problème toutefois avec cet algorithme, surgit si on veut faire usage d'images couleurs. En effet ces images sont généralement en trois composantes RGB [17]. On pourrait envisager de traiter alors ces composantes de l'image séparément. Si on procède de la sorte, on ne pourrait tenir compte des corrélations existantes entre les composantes, ni tenir compte aussi des facteurs liés à la perception de ces couleurs. Dans cette optique, une technique permet d'adapter l'algorithme LBG en vue d'exploiter les dissymétries couleurs [8] pour mieux prendre en compte la répartition spatiale des couleurs, et avoir une meilleure image couleur à la reconstruction.

La résolution du problème de couleur étant proposée par plusieurs sortes de techniques, nous nous sommes attardés sur les images .pgm en niveau de gris, avec pour objectif de mettre sur pied une adaptation du même algorithme itératif LBG, mais qui tire sa puissance du parallélisme rendu possible désormais dans des architectures à processeurs graphiques spécialisés. Ainsi, nous nous sommes proposés dans cette phase d'effectuer une quantification vectorielle sur une image en niveau de gris, elle même issue d'une transformée en ondelettes discrète 2D par lifting.

Notre codeur utilise:

- la *règle du plus proche voisin*:

$$\forall(i,j),\ si\ d(x,y_i) \leq d(x,y_j)\ alors\ C(x) = y_i \tag{15}$$

  Lors de la sélection d'un vecteur y dans le dictionnaire pour encoder le vecteur x, nous utilisons la mesure de distance séparant les vecteurs. Pour cela, nous avons le choix entre: la distance euclidienne (formule 16), la distance de corrélation (formule 17), la notion de maximum (formule 18).

  –

$$d_1(x,y) = \sqrt{\sum_{i=0}^{N}(X_i - Y_i)^2} \tag{16}$$

17. red-blue-grey: décrivent les intensité de ces trois couleurs de base, à partir desquelles nous retrouvons toutes les gammes de couleurs connues.

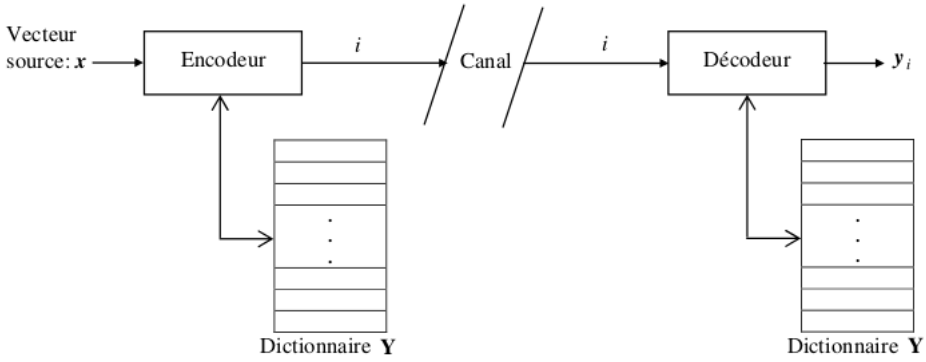

Figure 18: schema général d'un quantificateur vectoriel

–

$$d_2(x,y) = \sum_{i=0}^{N}(X_i * Y_i)/\sqrt{\sum X_i^2 * \sum Y_i^2} \tag{17}$$

–

$$d_3 = \tag{18}$$

- la règle du centrosome: Pour un ensemble de m points x_i d'une classe $C_k$donnée, le représentant de la classe est donné par $C_k = c|c =$ qui n'est rien d'autre que le centrosome de la classe.

Afin d'évaluer les critères de performance dans le cas d'une image $M * N$, nous faisons usage des paramètres suivants:

- MSE[18] $\frac{1}{M*N}\sum_{i=0}^{M}\sum_{j=0}^{N}|I_{i,j} - I'_{i,j}|$ qui permet de mesurer la distorsion entre l'image de départ I, et l'image reconstruite I' après quantification.
- PNSR[19] $PNSR = 20 * \log\left(\frac{255}{\sqrt{MSE}}\right)$

Dans sa version initiale, l'algorithme LBG pour produire le dictionnaire, repose sur l'initialisation de ce dernier par la méthode de dichotomie vectorielle par "splitting", consistant à diviser chaque vecteur représentant y du dictionnaire, en deux nouveaux vecteurs $y + \epsilon$ et $y - \epsilon$, où ϵ est un vecteur de perturbation aléatoire et de faible énergie. Par la suite on itère jusqu'à obtenir le nombre de vecteurs représentants désiré (il double à chaque itération), et jusqu'à la stabilité (convergence) de l'algorithme: c'est la phase d'apprentissage

Méthodes possibles d'obtention des régions avant quantification

Article mq.....

18. mean square error
19. mean square error

Algorithme LBG

Entrée: Données de la séquence d'apprentissage.

1. Initialisation du dictionnaire $\{y_1^0, y_2^0, ..., y_m^0\}$.
   Répéter 2,3,4 tant que la condition d'arrêt 5 n'est pas vérifiée.
2. classification: Pour chaque vecteur de la séquence d'apprentissage, affecter une étiquette en utilisant la condition du plus proche voisin.
3. optimisation: Pour la partition de l'espace des vecteurs d'apprentissage obtenue, calculer les nouveaux représentants par la condition du centrosome. $\{y_1^k, y_2^k, ..., y_m^k\}$
4. calcul de la distorsion:$D^k = \sum_{i=1}^{L} \sum_{x \in c} d(x, y_i^k)$
5. condition d'arrêt:$|\frac{D^k - D^{k+1}}{D^k}| \leq \varepsilon$

Sortie: Codebook $\{y_1^0, y_2^0, ..., y_m^0\}$

Pour notre besoin de le rendre efficace sur GPU, nous avons donc modifié cet algorithme pour l'adapter à notre environnement cuda. Pour notre image donc, nous appliquons pour

Algorithme LBG pseudo-parallèle Pour le mettre au point, nous procédons par les étapes décrites ci-dessous. ***Entrée***: Données de la séquence d'apprentissage.

1. Initialisation du dictionnaire $\{y_1^0, y_2^0, ..., y_m^0\}$.
   Répéter 2,3,4 tant que la condition d'arrêt 5 n'est pas vérifiée.
2. classification: Pour chaque vecteur de la séquence d'apprentissage, affecter une étiquette en utilisant la condition du plus proche voisin.
3. optimisation: Pour la partition de l'espace des vecteurs d'apprentissage obtenue, calculer les nouveaux représentants par la condition du centrosome. $\{y_1^k, y_2^k, ..., y_m^k\}$
4. calcul de la distorsion:$D^k = \sum_{i=1}^{L} \sum_{x \in c} d(x, y_i^k)$
5. condition d'arrêt:$|\frac{D^k - D^{k+1}}{D^k}| \leq \varepsilon$

Sortie: Codebook $\{y_1^0, y_2^0, ..., y_m^0\}$

3.2.5 Implémentation

CUDA simplifie d'une certaine manière le modèle de programmation parallèle, par le procédé des threads et la programmation hétérogène, pour les développeurs avisés. La gestion des threads est faite de manière plus souple contrairement aux primitives bas niveau du standard C ANSI (fork, mutex, etc...). Cependant, ces dernières instructions sont sous-jacente, et transparentes aux programmeurs, mais quelques fois ils faudrait encore s'y replier les manches pour attaquer à ce niveau les épines dures souvent rencontrées en programmation système. Cuda inclut la gestion des synchronisations dans les situations multiples, et avec la gestion du cache et de la mémoire partagée. Toute mauvaise manipulation à ce niveau conduit systématiquement à de mauvais résultats.

environnement de test

La plate-forme utilisée pour les test CPU est constituée d'un processeur Intel Core i3 avec ... coeurs, tournant chacun à ... Ghz avec ... Go de mémoire RAM. Cependant, notons que malgré la qualité de multi-cœurs, notre programme sur CPU a été purement séquentiel. Les résultats sur GPU ont été obtenus sur une carte NVIDIA GPU GT 520M avec 1Go de mémoire globale, 1 multiprocesseurs de 48 cœurs chacun, pour un total de 48 cœurs cuda, à 1.48Ghz. La version du driver utilisé fut le 5.0 avec un visual studio 2008.

Résultats

Nous résumons dans cette section, les résultats obtenus par nos procédés décrits plus haut.

wavelets	***Horizontal***	***transpose***	***vertical***	***sliding***
Haar	0.707	0.899	0.868	
Deslaurier	7.511	0.8994	6.768	
LeGall53	0.520	0.9	0.552	0.2437
Daubechies	0.552	0.9	0.552	0.240

Table 5: performances sur une image $512 * 512$ (temps en ms)

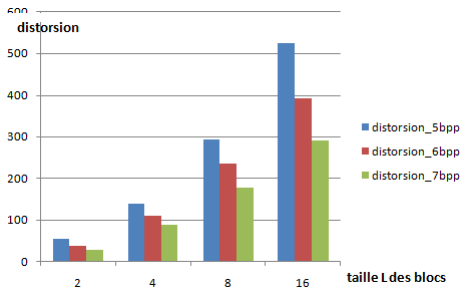

Figure 19: distorsion

blocks size	bit per pixel	iterations	distorsion	PNSR	cpu time	gpu time
2x2	5	236	56.14	70.55	21	4
	6	396	38.95	74.2	67	9
	7	233	28.32	77.39	77	15
4x4	5	166	141.04	61.34	12	3
	6	140	111.72	63.67	21	5
	7	251	88.54	65.96	75	9
8x8	5	37	294.86	53.96	2	0.8
	6	53	236.7	56.16	7	1.3
	7	28	179.32	58.93	8	2
16x16	5	28	525.26	48.19	2	1
	6	18	395.14	51.03	2	1
	7	12	292.75	54.03	3	1

Table 6: performances on an $512 * 512$ image quantization, cpu and gpu version

Conclusion et perspectives

Les récentes années ont vu un intérêt très signifiant dans la communauté scientifique, pour l'adoption des architectures GPU pour la résolution des problèmes scientifiques à large échelle, surtout ceux demandant une énorme puissance de calcul. Dans ce document, nous avons présenté la notion d'ondelettes, son apport dans le traitement d'images, et son application directe dans la chaîne de compression d'images. La compression d'image est un vaste domaine de recherche extrêmement actif en traitement du signal. Plusieurs approches de traitement du signal sont dérivés des ondelettes, mais nous nous sommes focalisés sur certaines ondelettes par la méthode lifting scheme, que nous avons implémentés. Étant dans un contexte de traitement de volumes importants d'images vouée à une utilisation à grande échelle, et voulant tirer parti de l'évolution des architectures de machine, nous avons associé notre étude à une implémentation sur processeurs graphiques dédiés, que sont les GPU. Par la suite, dans un souci de compression efficace avec optimalité débit/distorsion, nous avons également mis sur pied et implémenté une version sur GPU, de l'algorithme LBG pour la construction de dictionnaire dans le processus de quantification vectorielle, avant le codage entropique, pour boucler la chaîne de compression du signal, telle que connue. Nous notons un facteur d'accélération non négligeable, qui montre la puissance des GPU face aux version séquentielles connus des dits algorithmes. Une évolution future de nos travaux, nous donneraient de faire le même travail, mais avec d'autres types de quantificateurs tels les quantificateurs algébriques, ou bien d'autres, tout en tenant en compte le les représentation des régions d'intérêts dans notre processus de compression, donnant de compresser à débit variable, chaque région de l'image. Nous pourront également étendre notre étude non plus aux signaux 2D, mais également aux signaux 3D, comme mentionné sur la figure 3.2.5 ci-dessous.

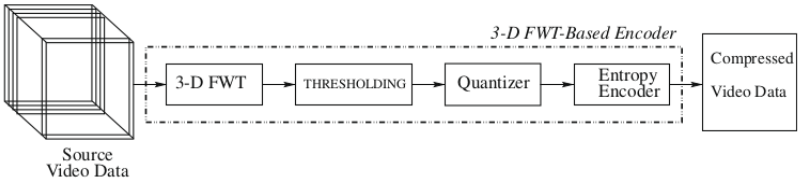

Figure 20: Étapes de réalisation d'un encodeur basé sur la 3D-FWT.

References

[1]

[2] *Vector Quantization and Signal Compression.* Khiwer Academic Publishers, 1991.

[3] *Digital image processing, second Edition.* Tom Robbins, 2002.

[4] V. Michau L. Mugnier M. Duplaa P. Henry et C. Valorge. (A. Kattnig, D. Leger. Pôle optique spatiale, groupe de proposition et d'exécution, sous-groupe : qualité et restauration d'image état des lieux. Technical report, Centre Nationale d'Etudes spatiales/Onera, 1999.

[5] Ouafi A. Baarir E. Etude de la transformée en ondelettes dans la compression d'images fixes. Technical report, Laboratoire de recherche LESIA, Département d'Electronique Université Mohamed Khider, 2004.

[6] Caroline Chaux. *Analyse en ondelettes M-bandes en arbre dual:application à la restauration d'images.* PhD thesis, Signal and Image processing, Université de Marne la Vallée, 2006.

[7] Emmanuel Christophe. *Compression des Images Hyperspectrales et son Impact sur la Qualité des Données.* PhD thesis, Ecole nationale supérieure de l'aéronautique et del'espace, 2006.

[8] Hocine C. Christophe C. Optimisation du dictionnaire pour la quantification vectorielle d'images couleur. In *Seizième colloque Gretsi.*

[9] González J. Gregorio B., Jose M. A lossy 3d wavelet transform for high-quality compression of medical video. *The Journal of Systems and Software*, 2009.

[10] Fernandez J. Acacio E. Joaquin F., Gregorio B. A parallel implementation of the 2d wavelet transform using cuda. 2011.

[11] Stephane G. Mallat. A theory for multiresolution signal decomposition:the wavelet representation. *IEEE Transactions on patttern analysis and machine intelligence*, 1989.

[12] Ole Moller Nielsen. Vector parallel fast wavelet transforms.

[13] Caitali C. Tinku A. A survey on lifting-based discrete wavelet transform architectures. *Journal of VLSI Signal Processing*, 2006.

[14] Maria Trocan. *Décompositions spatio-temporelles et allocation de débit utilisant les coupures des graphes pour le codage vidéo scalable*. PhD thesis, Ecole Nationale Supérieure des Télécommunications, Paris, 2007.

[15] Geoffrey M. Wim Sweldens, Roger L. Non linear wavelet transforms for image coding via lifting. *IEEE Transactions on Image Processing*, 1999.

[16] Andrei C.J. Jos B. Wladimir J., van D. Accelerating wavelet lifting on graphics hardware using cuda. *IEEE Transactions on parallel and distributed systems*, 2011.

[17] Yann-Gaudeau. *Contributions en compression d'images médicales 3D et d'images naturelles 2D*. PhD thesis, Université Henri Poincaré, Nancy 1, 2006.

Annexe