الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne Démocratique et Populaire
وزارة التعليم العالي والبحث العلمي
Ministère de l'enseignement supérieur et de la recherche scientifique

Université Mohamed Khider-Biskra
Faculté des sciences et de la technologie
Département: Génie électrique

جامعة محمد خيضر بسكرة
كلية العلوم و التكنولوجيا
قسم: الهندسة الكهربائية

Réf:.....20182018/.....:المرجع

Thèseprésentée en vuede l'obtention
du diplôme de
Doctorat en Sciences en :Génie électrique
Spécialité(option): Electronique

Compression orientée"Contenu" d'images

Présentée Par :
HETTIRI Messaoud

Soutenue publiquement le…….

Devant le jury composé de :

Dr. Salim Sbaa	Président	MCA	Université deBiskra
Dr. Zine-Eddine Baarir	Rapporteur	Prof	Université de Biskra
Dr. Noureddine Doghmane	Examinateur	Prof	Université d'Annaba
Dr. Abdelmalik Taleb-Ahmed	Examinateur	Prof	Université de Valenciennes

Dédicaces

- A l'âme de ma mère qu'elle repose en paix (00/00/1938-18/11/2005).
- A mon père pour son éducation et ses encouragements sans limites.
- A toute ma grande famille, sans lesquels, je ne serai pas ici aujourd'hui.

Remerciements

Je remercie tout d'abord Dieu le tout puissant de m’avoir donné le courage, la force et la patience d'achever ce travail.

Je tiens à remercier très chaleureusement, Dr. Baarir Zineddine, qui m’a permis de bénéficier de son encadrement.

Je remercie les membres de jury qui ont accepté de juger ce travail et d’y apporter leur caution :

Dr. Sbaa Salim, qui me fait le grand honneur de présider ce jury.

Dr. Noureddine Doghmane et Dr. Abdelmalik Taleb-Ahmed pour avoir accepté de faire partie de ce jury en tant qu’examinateurs.

J’adresse mes vifs remerciements également à tous ceux qui ont participé, de près ou de loin, au bon déroulement de cette recherche.

Résumé

Les représentations utilisées en compression d'images sont basées sur des transformées séparables (DCT, ondelettes..). Celles-ci ne capturent pas la régularité géométrique des images le long des contours. Donc, il est nécessaire de mettre au point des compresseurs qui s'intéressent à la représentation permettant de capturer à la fois cette régularité le long des contours et la régularité de zones. Nous nous intéressons dans ce travail, à l'élaboration d'un algorithme de compression d'images fixes basé sur une nouvelle famille de bases, les bandelettes, permettant de capturer les singularités le long de contours. Le codeur proposé utilise est le codage imbriqué qui offre la propriété de la transmission progressive de l'image codée.

Nous proposons dans ce travail une nouvelle approche basée sur une nouvelle version de l'algorithme SPECK "Set Partionning Embedded Block" appliqué à une image transformée par bandelettes. L'approche proposée est caractérisée par sa simplicité et donne de meilleurs résultats par rapport au SPECK original car elle exploite plus les corrélations entre les coefficients de bandelettes.

Mots-clés : Image fixe, Contours, Compression, Bandelettes, Codage imbriqué, SPECK, Régularité géométrique.

ملـــــخص

تستند التمثيلات المستخدمة حاليا في ضغط الصور على التحويلات المنفصلة (DCT، المويجات، الخ...)، هذه التمثيلات لا تأخذ بعين الاعتبار الانتظام الهندسي للصور على طول حدودها، على الرغم من أن هذا هو الجانب الأساسي لتحليل الصور. وبالتالي، فمن الضروري وضع طرق لضغط للصور تعتمد على تمثيل هذا الانتظام على حدود الصور وانتظام المناطق فيها. وما يهمنا في هذا العمل، هو إيجاد خوارزمية لضغط الصور على أساس التحويل بواسطة الشرائط الموجية، هذه الأخيرة تكشف لنا الانفراديات الموجودة على طول حدود الصورة. يستخدم أيضا التشفير المقترح في هذا العمل ترميزا مضمنا الذي يوفر خاصية النقل التدريجي للصورة المشفرة.

ونقترح في هذا العمل طريقة جديدة تقوم على تعديل خوارزمية SPECK "Set Partionning Embedded bloCK" تطبق على صورة محولة بواسطة الشرائط الموجية. تتميز هذه الطريقة المقترحة ببساطتها وتعطي نتائج أفضل مقارنة بالخوارزمية الأصلية، لأنها تستغل أكثر العلاقة بين معاملات الشرائط الموجية.

الكلمات المفتاحية: صورة ثابتة، حدود الصورة، ضغط الصور، الشرائط الموجية، ترميز مضمن، خوارزمية SPECK ، انتظام هندسي.

Abstract

The representations currently used in image compression are based on separable transforms (DCT, wavelets, etc.). These do not capture the geometric regularity of the images along the contours. Thus, it is necessary to develop compressors that are able for the representation in capturing both this regularity along the contours and the regularity of zones. We are interested in this work, by an algorithm of compression of fixed images based on a new family of bases, the bandelets, for capturing singularities along contours. The proposed encoder uses also the embedded coding that provides the property of progressive transmission of the coded image.

We propose in this work a new approach based on a new version of the SPECK algorithm "Set Partionning Embedded bloCK" applied to a fixed image transformed by bandelets . The proposed approach is characterized by its simplicity and gives better results compared to original SPECK because it exploits more the correlations between the bandelets coefficients .

Keywords: Fixed image, Contours, Compression, Bandelets, Embedded coding, SPECK, Geometric regularity.

Table des matières

Chapitre 2 : Transformée en bandelettes

Chapitre 3 : ***Codeur SPECK, Proposition d'une Optimisation SPECK modifié***

Chapitre 3 : Résultats et Discussions

Liste des abréviations

notation	**signification**
B	Bloc
bpp	Bit par pixel
C_i	Coefficient de bandelettes d'ordre i
DCT	Transformée de cosinus discrète
EZW	Arbre de zéros imbriqué des coefficients d'ondelettes.
EQM	Erreur Quadratique Moyenne
i	indice
Inf	Valeur du pointeur d'un coefficient significatif
L	Lagrange
LIS	Liste de bits insignifiants.
LSP	Liste de bits signifiants.
r	rang
s	super résolution pour la géométrie
SPECK	Partition imbriqué d'un groupe de blocs
SPIHT	Partition d'un groupe dans un arbre Hiérarchique
PSNR	rapport du signal sur bruit crête
TC	Taux de Compression
T_g	Seuil de la géométrie
T_n	Seuil de la signifiance d'ordre n

Liste des figures

Liste des tableaux

Introduction générale

La compression d'images est devenue aux cours de ces dernières années un sujet de très grand intérêt étant donné les sévères limitations qui contribuent de s'appliquer sur les capacités de stockage et de transmission alors que la résolution des instruments ne cesse de s'accroitre.

La recherche d'une bonne représentation est un problème central de la compression d'image. On cherche ici des bases permettant une représentation creuse des images, c'est-à-dire permettant de bien les approcher avec peu de coefficients. Cette propriété est en effet à la base de nombreux algorithmes de compression.

Les méthodes de compression d'images les plus performantes à l'heure actuelle, sont basées sur la transformée en ondelettes qui permet de décorréler spatialement les images. Les méthodes utilisées jusqu'à présent sont en effet des méthodes génériques, c'est-à-dire qu'elles ne tiennent pas compte du contenu des images étudiées. Or, pour pouvoir être plus performant et dépasser les taux de compression limites atteints avec les transformations (décorrélations spatiales) actuelles (DCT, Ondelettes,…), il est nécessaire de mettre au point des compresseurs (intelligents) qui s'intéressent au contenu de l'image avant la compression. L'objectif de cette thèse est de mettre en place une nouvelle méthode de compression applicable aux images 2D appelée transformée en Bandelettes de façon à exploiter la géométrie de l'image (le contenu de l'image) afin d'améliorer le rapport qualité/ compression.

Les bases d'ondelettes permettent d'obtenir une telle représentation. Elles ne sont cependant pas optimales pour les images. Bien que très efficaces pour les zones régulières, les textures homogènes et les singularités ponctuelles, elles ne peuvent exploiter la régularité de nature géométrique des contours [8]. Les bases de bandelettes présentées ici permettent une exploitation efficace de cette régularité géométrique. Les bases de bandelettes sont construites à partir d'ondelettes bidimensionnelles déformées le long du flot géométrique dans chaque région de l'image [3].

Dans ce travail, nous avons effectué sur l'image la transformée en bandelettes associé à un codage imbriqué. Les algorithmes de codage imbriqué EZW[3], SPIHT[5] et SPECK[1] exploitent les corrélations entre les coefficients

d'une image transformée en ondelettes. Les deux premiers algorithmes sont basés sur les corrélations d'un groupe de coefficients coordonnés dans les différentes sous-bandes comme un ensemble nommé arbre. Par contre, le dernier est basé sur la partition successive de l'image transformée en ondelettes en blocs de tailles différentes selon les signifiances de coefficients d'ondelettes.

L'efficacité du SPECK est claire par rapport aux autres codeurs car le test de signifiance débute du plus grand bloc jusqu'au plus petit bloc et le bloc rassemble plus de coefficients que l'arbre [9].

Pour accomplir ce travail, cette thèse est organisée selon les chapitres suivants :

Dans le premier chapitre, nous décrivons tout d'abord les principes généraux de la compression, (transformation, quantification et codage) ; ensuite, nous décrivons la classification des méthodes de codage qui sont divisées en deux classes :

- Les méthodes sans pertes d'informations (codage de shannon-fano, codage de Hufmann et codage arithmétique,…)

- Les méthodes avec pertes d'informations (quantification vectorielle, codage par transformées et codage imbriqué).

Dans le deuxième chapitre, nous détaillons la transformation en ondelettes et la transformation en bandelettes en particulier. L'idée étant de montrer son intérêt lorsqu'elle est utilisée pour le codage imbriqué.

Le troisième chapitre décrit l'algorithme SPECK avec un certain nombre d'exemples d'utilisation et détaille notre contribution appelée SPECK modifié, qui consiste en l'optimisation de l'algorithme SPECK.

Enfin, dans le dernier chapitre, nous présentons les résultats de simulation obtenus avec différentes images de test, nous effectuons aussi une analyse des résultats obtenus ainsi qu'une étude comparative en nous basons sur la métrique taux de compression et PSNR.

Nous terminons cette thèse par une conclusion générale et des perspectives envisagées.

1-1-Introduction

La compression est la réduction du nombre de bits par pixel à stocker ou à transmettre, en exploitant la redondance informationnelle dans l'image. Les techniques de compression se différencient par le fait qu'elles permettent de compresser sans pertes d'informations (méthodes réversibles) ou avec perte d'informations (méthodes irréversibles).

1.2 Image numérique

L'image numérique est définie comme une étendue plane entièrement remplie par des éléments graphiques atomiques appelés "pixels". Chaque pixel correspond à un nombre qui renvoie soit à une couleur, soit à un niveau de gris.

1.2.1 Pixel

Le pixel est la contraction de l'expression anglaise "Picture element"; le pixel est le plus petit point de l'image, c'est une entité calculable qui peut recevoir une structure et une quantification. La qualité d'information véhiculée dans chaque pixel donne des nuances entre une image à niveaux de gris et une image en couleurs [27].

Dans le cas d'une image à niveaux de gris, chaque pixel est codé sur huit bits, et la taille de mémoire nécessaire pour afficher une telle image est directement liée à la taille de l'image.

Dans une image en couleurs RVB, un pixel peut être représenté sur trois octets, un octet pour chacune des couleurs : Rouge (R), Vert (V) et Bleu (B).

1.2.2 Image à niveaux de gris

Le niveau de gris est la valeur de l'intensité lumineuse en un point, la couleur du pixel peut prendre des valeurs allant du noir ou blanc passant par un nombre fini de niveaux intermédiaires.

Donc, pour représenter les images à niveaux de gris, on peut attribuer à chaque pixel de l'image, une valeur correspondant à la quantité de la lumière renvoyée. Cette valeur peut être comprise entre 0 et 255. Chaque pixel n'est donc plus représenté par un bit, mais par un octet. Pour cela, il faut que le matériel utilisé pour afficher l'image soit capable de produire les différents niveaux de gris correspondants.

1.2.3 Image en couleur

La représentation des couleurs s'effectue de la même manière que les images à niveaux de gris avec quelques particularités. En effet, pour choisir un modèle de représentation, on peut représenter les couleurs à l'aide de leurs composantes primaires. Les systèmes émettant de la lumière (écrans d'ordinateur,…) sont basés sur le principe de la synthèse additive ; les couleurs sont composées d'un mélange de rouge, vert et bleu [26].

1.3 Compression d'images

1-3-1 Définition de la compression d'image

Les méthodes de compression et de codage réduisent le nombre de bits par pixel à stocker ou à transmettre, en exploitant la redondance informationnelle dans l'image [21]. Les principaux critères d'évaluation de toute méthode de compression sont :

- La qualité de la reconstitution de l'image.
- Le taux de compression.
- La rapidité du codeur et du décodeur (codec).

1.3.2 Schéma fonctionnel de la compression

Le schéma fonctionnel de la compression d'un tel codeur d'images représenté ci-dessous [1].

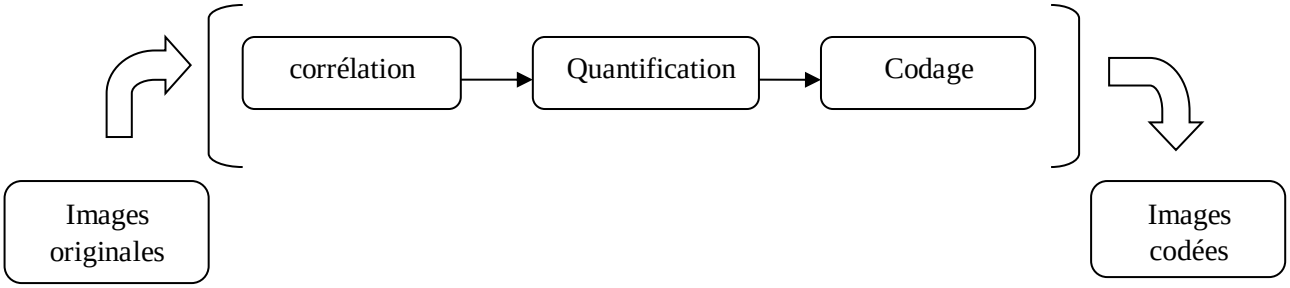


Figure 1.1 : Schéma synoptique d'un codeur source

1.3.2.1 Décorrélation

La dépendance existante entre chaque pixel et ses voisins (la luminosité varie très peu d'un pixel à un pixel voisin) traduit une corrélation très forte sur l'image. On essaie donc de tirer profit de cette corrélation, pour réduire le volume d'information en effectuant une opération de décorrélation des pixels.

La décorrélation consiste à transformer les pixels initiaux en un ensemble de coefficients plus corrélés, c'est une opération réversible.

1.3.2.2 Quantification

La quantification des coefficients a pour but de réduire le nombre de bits nécessaires pour leurs représentations. Elle représente une étape clé de la compression. Elle approxime chaque valeur d'un signal par un multiple entier d'une quantité "q", appelée quantum élémentaire ou pas de quantification, elle peut être scalaire ou vectorielle. C'est à ce niveau qu'il y a perte de l'information.

1.3.2.3 Codage

Les coefficients quantifiés forment un train binaire le plus réduit possible. Cette opération ne provoque aucune distorsion ou perte de l'information contenue dans l'image.

1-4-Méthodes de compression :

La compression peut être définie comme étant un système dont l'entrée est une image originale et la sortie est un flux de données numériques relativement court représentant l'image compressée. Le processus inverse est appelé décompression permettant la reconstruction de l'image à partir du flux de données numériques. Parfois, les systèmes de compression et de décompression ensembles sont appelés "*Codec*" (codage pour compression et décodage pour décompression). Suivant la qualité de la sortie de ces systèmes, on distingue deux types de méthodes de compression.

1-4-1- Compression conservatrice :

La compression conservatrice est une méthode de compression sans pertes d'informations (sans distorsion ou réversible). Dans ce cas, les données compressées et décompressées sont tout à fait équivalentes. Elle est plus particulièrement utilisée pour les données informatiques dont l'exactitude est décisive (Exemple : données textuelles, programmes, images médicales…) [20].

· Il existe diverses méthodes sans pertes d'informations telles que le codage de Shannon-Fano, le codage de Huffman, le codage arithmétique, les méthodes statistiques, les méthodes de plage.

1-4-1-1 Codage de Shannon-Fano

Shannon du laboratoire Bells et R. M. Fano du MIT ont développé presque en même temps une méthode de codage basée sur de simples connaissances de la probabilité d'occurrence de chaque symbole dans le message [13].

Le procédé de Shannon-Fano construit un arbre descendant à partir de la racine, par divisions successives. Le classement des fréquences se fait par ordre décroissant, ce qui suppose une première lecture du fichier et la sauvegarde de l'en-tête.

Le principe est le suivant :

- ✓ Classer les **"n"** fréquences non nulles $\{f_i\}$ par ordre décroissant.
- ✓ Répartir la table des fréquences en deux sous tables de fréquences proches.
- ✓ Poursuivre l'arborescence jusqu'à ce que toutes les fréquences soient isolées.
- ✓ Attribuer dans l'arborescence le bit 0 à chaque première sous-table.
- ✓ Attribuer aux symboles les codes binaires correspondants aux bits de description de l'arborescence.

1-**4-1-2 Codage de Huffman**

Cette méthode repose sur la construction d'un arbre basée sur les probabilités d'apparition des éléments ; on représente chaque élément "**i**" par une séquence de bits de longueur inversement proportionnelle à la probabilité d'apparition **p(i)**. Ce codage est optimal pour les codes à nombre de bits entiers [20].

Exemple : Soit à coder les éléments suivants, avec les probabilités suivantes :

Symbole	a	b	c	d	e	f
Probabilité	0.5	0.2	0.1	0.05	0.05	0.1
Code	0	10	1100	11010	11011	111

Tableau 1.1 : Symboles avec leurs probabilités d'apparition et le code de Huffman correspondant.

Le principe de la construction de l'arbre est comme suit:

1. Les feuilles sont des nœuds libres.

2. Choisir les deux nœuds libres de probabilités les plus faibles.

3. Créer un nœud père de poids somme des poids des nœuds fils.

4. Associer '0' à la branche gauche et '1' à la branche droite.

5. Le père est un nœud libre et les fils ne le sont plus.

6. Tant qu'il y a plus d'un nœud libre, répéter les étapes 2,3,4 et 5.

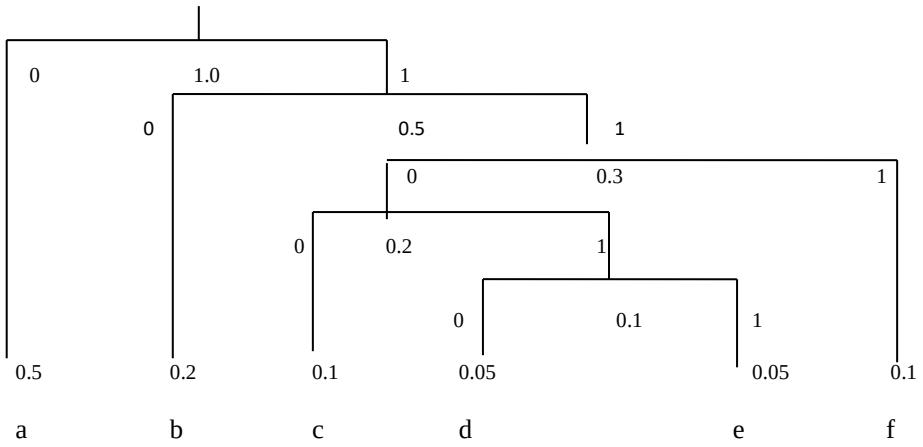


Figure 1.2 : Construction de l'arbre de Huffman

Si par exemple, on veut coder le mot « café », on obtient le code 1100011111011.

1-4-1-3 Codage arithmétique

Le codage arithmétique évite la contrainte du codage par un nombre de bits entiers ; il consiste à coder une chaîne de symboles par un nombre appartenant à l'intervalle [0,1[. On décrit brièvement la procédure de ce codage ; le décodage opère de manière inverse [5].

1. Calculer la probabilité associée à chaque symbole dans la chaîne à coder.

2. Associer à chaque symbole un sous-intervalle de [0,1[proportionnel à sa probabilité (l'ordre de rangement des intervalles sera mémorisé car il est nécessaire au décodeur).

3. Tant qu'il reste un symbole dans la chaîne à coder :

(a) largeur = limite supérieure – limite inférieure

(b) limite inférieure = limite inférieure + largeur x (limite basse du sous intervalle du symbole)

(c) limite supérieure = limite inférieure + largeur x (limite haute du sous intervalle du symbole)

4. La limite inférieure finale code la chaîne de manière unique.

Pour coder un message contenant **n** éléments, il faut β bits avec :

$$\beta = \sum_{i=1}^{n} \log\ p(i) \qquad (1.4)$$

Exemple [20]**:**

Considérons l'exemple précédent:

Symbole	a	b	c	D	e	f
Probabilité	0.5	0.2	0.1	0.05	0.05	0.1
Sous- intervalle	$[0,0.5[$	$[0.5,0.7[$	$[0.7,0.8[$	$[0.8,0.85[$	$[0.85,0.9[$	$[0.9,1[$

Tableau 1.2 : Symboles avec leurs probabilités d'apparition et sous-intervalles.

Soit le codage du message café :

1. $c \in [0.7,0.8[$ et le symbole suivant $a \in [0,0.5[$, ce qui réduit l'intervalle courant à $[0.7,0.7+(0.8-0.7)\times 0.5[= [0.7,0.75.[$.

2. Le symbole suivant $f \in [0.9,1[$, qui à l'échelle de $[0.7,0.75[$ devient:

$$[(0.7+0.05\times 0.9),(0.7+0.05\times 1)[\ \text{Soit}\ [0.745,0.75.[\ .$$

3. Le dernier symbole e, initialement dans $[0.85,0.9[$ donne l'intervalle final $[(0.745+0.005\times 0.85),(0.745+0.005\times 0.9)[= [0.74925,0.7495[$.

Le décodage procède de la manière suivante:

1. 0.74925 est dans l'intervalle $[0.7,0.8[$, ce qui correspond au symbole c.

2. l'intervalle $[0.7,0.8[$ est divisé selon les probabilités ordonnées en $[0.7,.0.75,0.77,0.78,0.785,0.79,0.8[$, 0.74925 est dans le premier sous- intervalle, et correspond donc au symbole a.
3. Etc.…

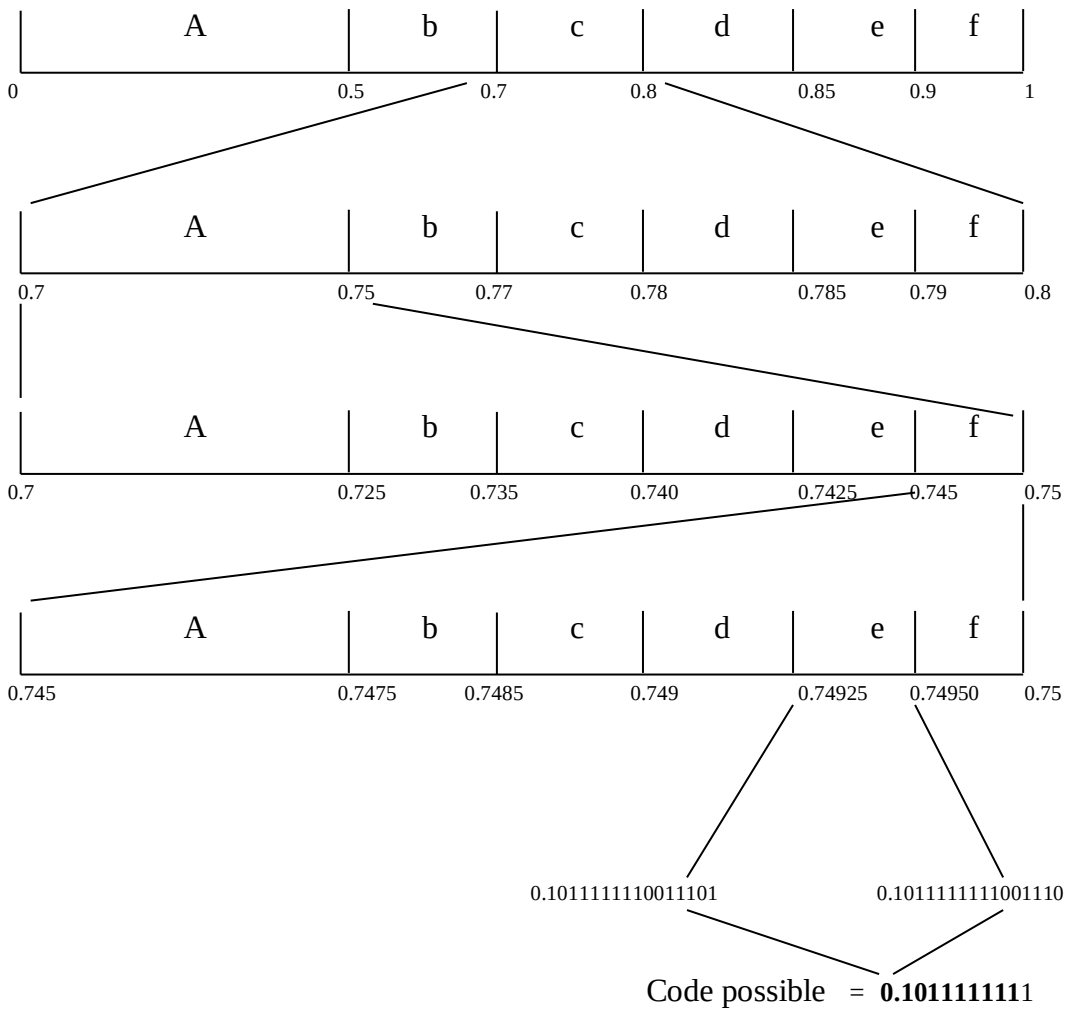


Figure 1.3 : Processus du codage arithmétique

La séquence « café » est codée avec le codage de Huffman en utilisant 14 bits, mais seulement en utilisant 12 bits avec le codage arithmétique; donc cette dernière technique est plus lente que le codage de Huffman mais elle présente des taux de compression supérieurs.

1-4-1-4 Méthode des plages

Lorsqu'on considère une ligne de la matrice représentant une image numérique, plusieurs échantillons successifs sur cette ligne peuvent posséder la même valeur. L'ensemble de ces échantillons est appelé " plage ". Cette méthode consiste donc à décrire les suites des pixels identiques par leurs longueurs et leurs valeurs. Par exemple, une plage de vingt pixels noirs équivaut à la donnée de 2 nombres : 20 et 0 [9].

1-4-2- Compression non conservatrice :

La compression non conservatrice est basée essentiellement sur l'élimination d'une partie de l'information jugée redondante. Ceci a pour effet l'obtention de meilleurs taux de compression, de plus en plus élevés, au détriment d'une certaine dégradation. Pour cette raison ces techniques de compression sont dites irréversibles. En effet, l'image originale et l'image reconstruite après compression ne sont pas les mêmes [8].

Il existe diverses méthodes avec pertes d'informations telles la quantification vectorielle, les méthodes de transformations, les méthodes prédictives, méthodes hybrides et les codeurs JPEG, JIBG.

1-5 Compression d'images fixes par ondelettes

1-5-1 Compression basée sur les approches classiques

Dans la compression de l'image avec ou sans pertes, les coefficients de la transformée en ondelettes sont habituellement quantifiés pour réduire le plus possible le nombre de valeurs destinés au codage entropique (figure 1.4). Certaines méthodes de compression appliquent la quantification scalaire sur les coefficients de la transformée en ondelettes ; d'autres méthodes avec la quantification vectorielle sont aussi proposées [26].

1-5-1-1 Quantification scalaire

La quantification scalaire est sous-optimale par rapport à la quantification vectorielle. Néanmoins, elle a été souvent utilisée pour des codeurs « universels » rapides. L'algorithme EPWIC (Embedded Predictive Wavelet Image Coder) est typique **.** Il est basé sur une quantification scalaire uniforme. Certains algorithmes ont, bien sûr, utilisé les caractéristiques statistiques connues à priori de la distribution en forme de gaussienne généralisée des

coefficients d'ondelettes, ou en faisant référence au système visuel humain pour limiter les artefacts causés par la quantification [13].

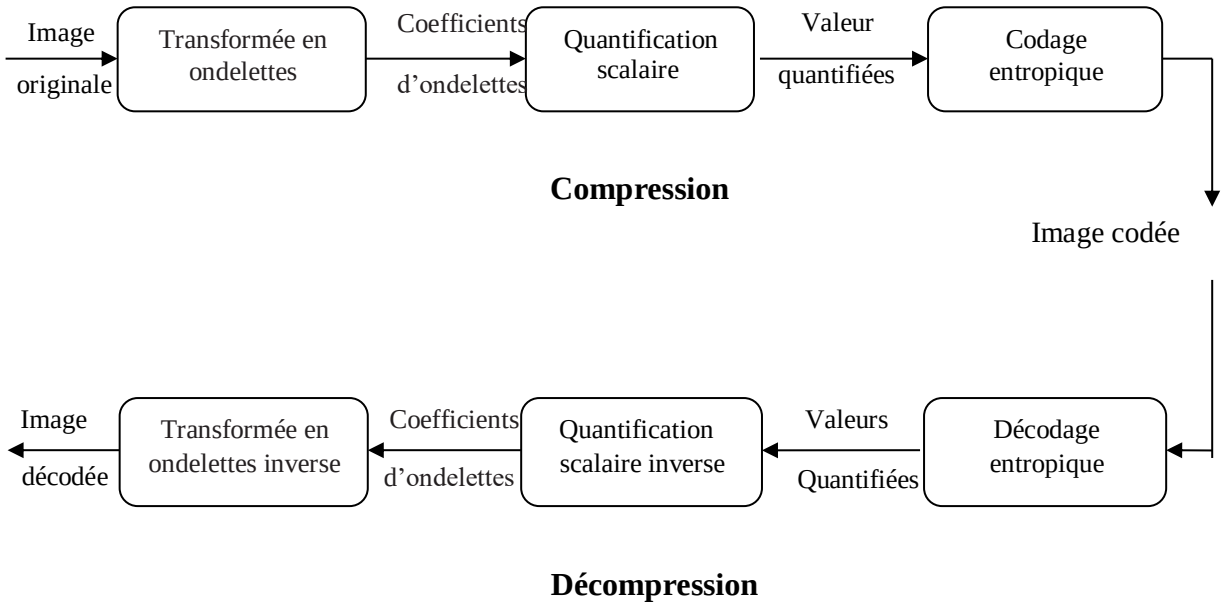


Figure 1.4 : Schéma synoptique de la compression /décompression basées sur les approches classiques

1-5-1-2 Quantification vectorielle

La quantification scalaire associe à une variable continue, une variable discrète pouvant prendre un nombre plus faible et fini de valeurs. Ces valeurs ne sont jamais totalement décorrélées, ou indépendantes. Shannon a montré qu'il est toujours possible d'améliorer la compression de données en codant des vecteurs plutôt que des scalaires [23].

La compression des images fixes par ondelettes utilisant la quantification scalaire est rapide par rapport à la quantification vectorielle, mais cette dernière donne un meilleur taux de compression vu sa grande capacité de mieux décorréler les coefficients d'ondelettes.

1-5-2 Compression basée sur la similarité des coefficients d'ondelettes

La méthode de compression basée sur la similarité des coefficients d'ondelettes utilise le codage imbriqué (figure 1.5), le rôle de ce dernier est de sélectionner les coefficients d'ondelettes significatifs pour un niveau de qualité souhaité et de considérer les autres comme non significatifs. Le codage s'avère particulièrement efficace lorsque l'on exploite les liens de

structure entre les différents coefficients non significatifs au sens d'une précision donnée. Donc, le but de ce codage est d'exploiter les corrélations entre les zones de coefficients non significatifs [26].

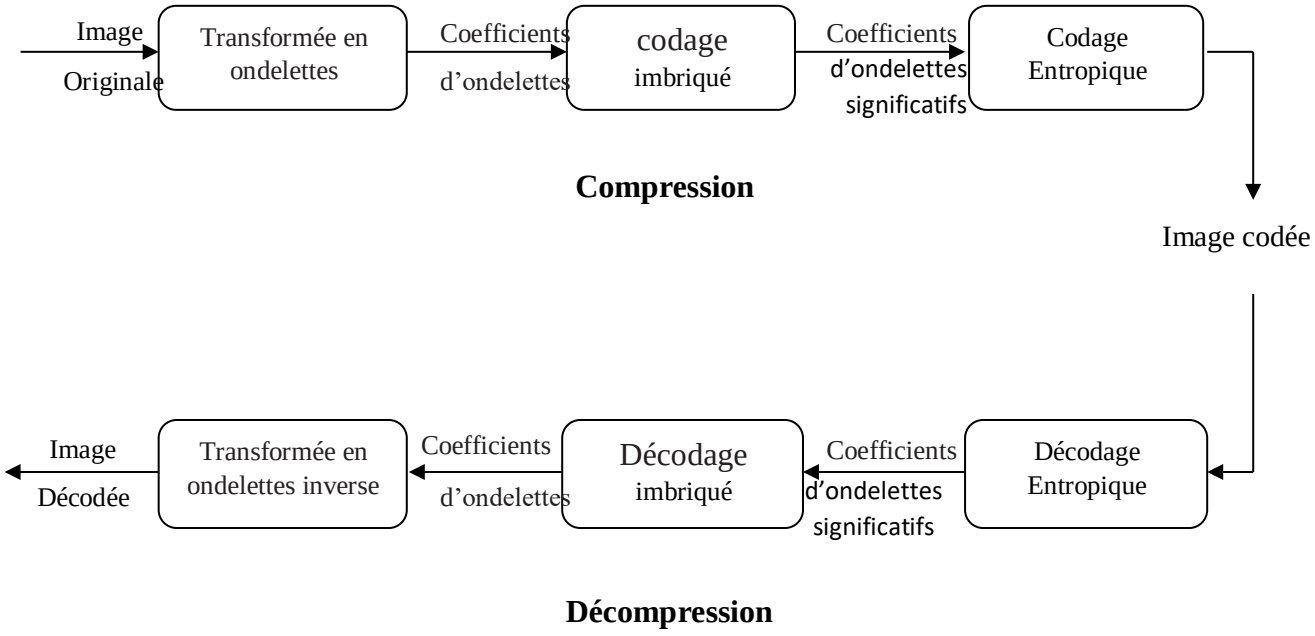


Figure 1.5 : Schéma synoptique de la compression/décompression basées sur la similarité des coefficients d'ondelettes

En effet, le codage imbriqué possède une propriété extrêmement intéressante : il code l'information par ordre exact d'importance dans l'image, c'est à dire, les coefficients sont codés du plus significatif au moins significatif. L'intérêt est évident : pour obtenir une compression sans pertes, on code tous les coefficients ; et pour obtenir une compression avec pertes, il suffit d'interrompre le décodage à l'instant souhaité, en fonction de l'image requise [20].

Pendant la décennie passée, beaucoup de travaux ont été proposés concernant un grand nombre de méthodes de codage imbriqué. Parmi ces méthodes, les plus connues sont l'EZW de Shapiro (embedded zerotree of wavelet coefficients), SPIHT de Said et Pearlman (Set Partitioning In Hierarchical tree), EBCOT de Taubman (Embedded Block Coding with Optimized Truncation) et SPECK de Islam et Pearlman (Set Partitioning Embedded Block).

1-5-2-1 EZW de Shapiro

On définit un niveau de seuil. Un coefficient d'ondelettes est dit significatif si son amplitude est plus grande que ce seuil. La valeur initiale du seuil est déterminée par :

$$T_0 = 2^{[\log_2(C_{max})]} \qquad (1.11)$$

Où $c_{\max}$ est l'amplitude maximale des coefficients d'ondelettes. On construit alors une liste de coefficients significatifs et une liste de coefficients non significatifs. A chaque passe, on code les coefficients significatifs [15]. Les bits non significatifs deviennent les bits de départ de la seconde passe sur lesquels un nouveau calcul de seuil est appliqué par division par deux du seuil précédent, et ainsi de suite par récurrence jusqu'au codage global déterminé par un nombre maximal de passes**.**

1-5-2-2 SPIHT de Said et Pearlman

Le SPIHT est une amélioration de l'EZW où on rassemble les coefficients nécessaires au codage par ondelettes dans un arbre. Dans le cas du SPIHT, cet arbre est divisé en sous-arbres. Ces sous-arbres sont organisés de manière à garder les coefficients importants ensembles (au sens de ceux qui sont à la base de la reconstruction du signal).

Le SPIHT ordonne les arborescences des coefficients significatifs et non significatifs avec plus de liberté que la simple hiérarchie de Shapiro [7-11].

1-5-2-3 Algorithme SPECK d' Islam et Pearlman

Offrant des performances comparables à l'algorithme SPIHT, l'algorithme SPECK [2-17] exploite des structures d'ensembles de coefficients non significatifs en blocs plutôt qu'en arbres. Ces structures de blocs permettent de s'affranchir efficacement de la non- stationnarité (d'ordre 1) des coefficients en adaptant localement la statistique utilisée pour le codage.

1-5-2-4 Algorithme EZBC

Le principe du codage par l'algorithme EZBC (Embedded Zero Blocks coding based on Context modeling) est similaire à l'algorithme SPECK. L'innovation de ce codeur provient principalement de l'exploitation de la dépendance entre les nœuds du quad-tree de signifiance. Le partitionnement en quad-tree est de plus réalisé indépendamment dans chaque sous-bande

permettant une meilleure séparation des statistiques de signifiance et un apprentissage plus efficace à l'aide de contextes plus étendus [4].

Ce codeur offre des performances comparables à EBCOT, et a également été adapté au codage vidéo sous le nom de MC-EZBC .

1.6 Conclusion

Dans ce chapitre, nous avons expliqué que les méthodes de compression sans pertes donnent un taux de compression inférieur à celui donné par les méthodes de compression avec pertes. Par contre, elles sont caractérisées par une rapidité de codage meilleure. Pour améliorer le taux de compression, la qualité de l'image et le temps de calcul, on peut combiner ces deux méthodes.

Pour la compression d'images, les ondelettes apparaissent donc comme un outil extrêmement efficace. Elles permettent en effet d'obtenir simultanément une qualité de compression supérieure aux autres méthodes et un taux de compression progressif.

2.1. Introduction

La transformée en ondelettes est l'une des premières représentations temps-fréquence, multi-résolutions à support compact qui devient une alternative à la transformée de Fourier. Cependant, l'inconvénient majeur de cette transformée est qu'elle ne parvient pas à saisir la régularité géométrique le long des contours dans les images.

De nouvelles transformées ont été proposées par de nombreux chercheurs ; il s'agit des bandelettes de seconde génération qui utilisent les structures géométriques contenues dans les images. Le but de ces transformées est d'améliorer la concentration de l'énergie dans un minimum de coefficients de la transformée pour aller dans le sens d'une représentation creuse.

2.2 Transformée en ondelettes

Les ondelettes sont des familles de fonctions obtenues par dilatations et translations successives d'une unique fonction appelée ondelette mère Ψ possédant certaines propriétés analytiques [15]. Sous certaines conditions, on peut obtenir une famille d'ondelettes formant une base orthogonale de $£^2(R)$:

$$\Psi_{m,n}(x) = \frac{1}{\sqrt{2^m}} \Psi(\frac{1}{2^m} x - n) \tag{2.1}$$

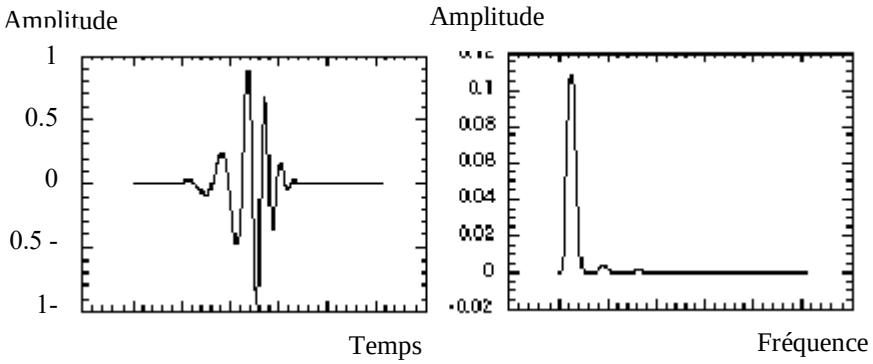
Par conséquent, tout signal $s(x)$ pourra être décomposé sur cette base :

$$s(x) = \sum_m \sum_n d_{m,n} \Psi_{m,n}(x) \tag{2.2}$$

Avec : $d_{m,n}$ sont les coefficients d'ondelettes obtenus par la discrétisation de la fonction suivante :

$$d_{m,n} \overset{\Delta}{=} \int s(x) \Psi_{m,n}(x) dt \tag{2.3}$$

L'intérêt d'une telle décomposition est que les fonctions $\psi_{m,n}$ possèdent la propriété remarquable d'être à la fois bien localisées en temps et en fréquence (filtre passe bande à support temporel étroit). Ainsi, le coefficient **"m"** correspond à la résolution d'analyse, tandis que **"n"** représente la localisation temporelle (figure 2.1).



(a) **(b)**

Figure 2.1 : Exemple d'ondelette mère, domaine temporel (a) et fréquentiel (b)

2.3 Analyse multirésolutions

2.3.1 Principe

Pour une ondelette mère $\Psi(x)$ donnée, on peut trouver une fonction particulière $\varphi(x)$ dite fonction d'échelle, ayant les propriétés d'un filtre passe-bas [20] (figure 2.2), et telle que l'approximation $s_m(x)$ du signal $s(x)$ à la résolution **"m"** s'écrive :

$$s_m(x) = \sum_m a_{m,n}\varphi_{m,n}(x) \qquad (2.4)$$

Et $a_{m,n}$ sont les coefficients d'approximation obtenus par la discrétisation de la fonction suivante :

$$a_{m,n} \overset{\Delta}{=} \int s(x)\varphi_{m,n}(x)dx \qquad (2.5)$$

Où : $$\varphi_{m,n}(x) = \frac{1}{\sqrt{2^m}}\varphi(\frac{1}{2^m}x - n) \qquad (2.6)$$

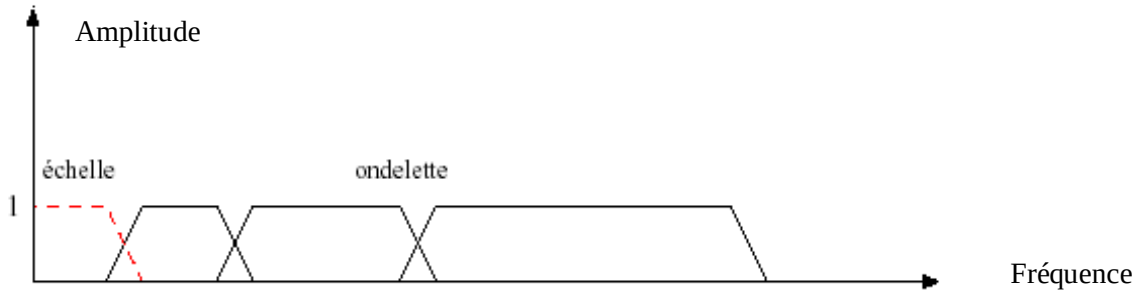


Figure 2.2 : Ondelettes formant une série de filtres passe-bande et la fonction d'échelle permettant d'assurer la couverture du spectre

Les coefficients de détails d'ondelettes $d_{m,n}$ représentent donc les détails du signal invisibles à la résolution "m" mais visibles à la résolution" m-1".

Etant donné un niveau de résolution maximal "M" fixé, il suffit alors de disposer des coefficients $a_{m,n}$ et les coefficients de détails $\{d_{m,n}\}_{m\leq M}$ pour reconstruire le signal original sans pertes.

2.3.2 Banc de filtres et analyse multirésolutions

L'intérêt de la représentation multirésolution est qu'elle peut être calculée à partir d'une transformée pyramidale mise en œuvre à l'aide de filtres numériques [20-23]. Le principe de la transformée pyramidale consiste en la décomposition du signal à analyser à l'aide d'une paire de filtres (h et g), l'un de ces filtres (g) fournira les coefficients d'ondelettes (ou détails), le second (h) les coefficients d'approximation.

L'approximation est elle-même à son tour décomposée par une seconde paire de filtres (figure 2.3), l'ensemble constituant une pyramide de filtres.

Les fonctions de transfert des filtres "h" et "g" étant calculées à partir des transformées de Fourier des fonctions $\Psi(x)$ et $\varphi(x)$:

$$\varphi(2w) = H(w)\varphi(w) \quad \text{Et} \quad \Psi(w) = G(w)\varphi(\frac{w}{2}) \tag{2.7}$$

Avec : $H(w) = TF\{h(x)\},\ G(w) = TF\{g(x)\}$

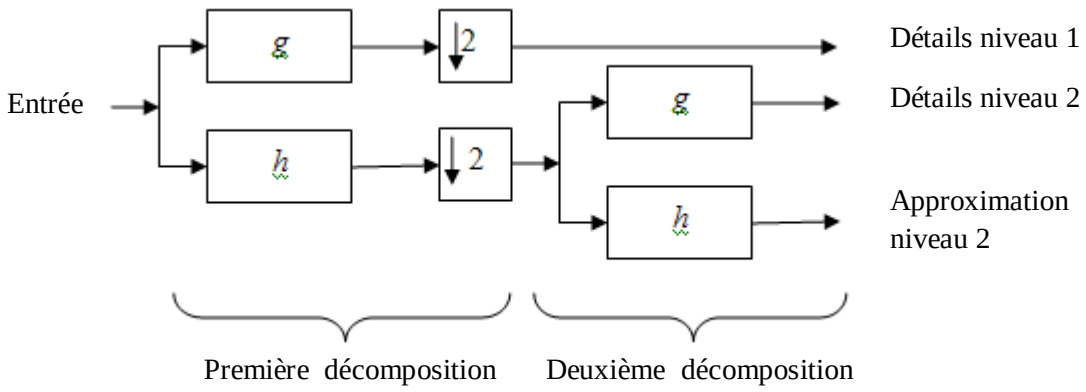


Figure 2.3 : Décomposition 1D à l'aide de bancs de filtres

2.3.2.1 Cas d'ondelettes orthogonales

Soit $s_0(n)$ un signal échantillonné correspondant au signal d'origine à la résolution "0". Ce signal est décomposé sur plusieurs niveaux de résolutions en deux bandes de fréquences (passe-haut et passe-bas) de la manière suivante, où s représentent l'approximation et d les détails (figure 2.4).

$$s_m(n) = \sum_k h(2n-k)s_{m-1}(k) \qquad (2.8)$$

Et

$$d_m(n) = \sum_k g(2n-k)s_{m-1}(k) \qquad (2.9)$$

Les filtres passe-bas (h) et passe-haut (g) définis par les équations (2.8) et (2.9) sont liés par la relation : $g(n) = (-1)^n h(1-n)$

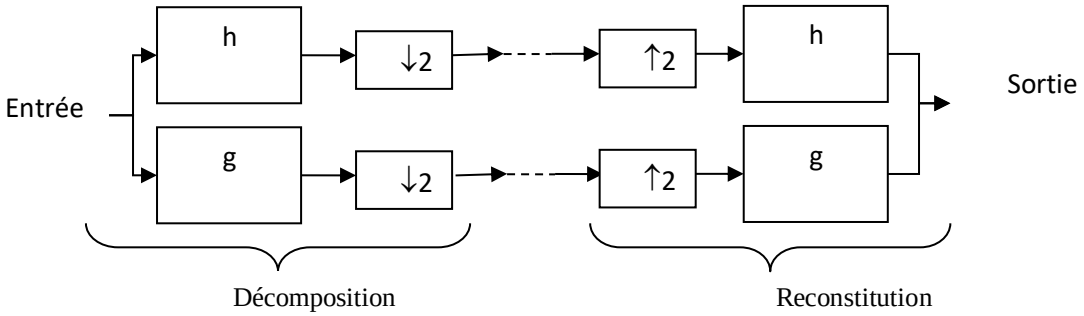


Figure 2.4 : Analyse et synthèse par filtres orthogonaux

La reconstruction est obtenue de la manière suivante :

$$s_{m-1}(n) = \sum_k h(2n-k)s_m(k) + \sum_k g(2n-k)d_m(k) \qquad (2.10)$$

2.3.2.2 Cas d'ondelettes bi-orthogonales

La construction des FCQ (filtres conjugués en quadrature) doit donc répondre à un certain nombre de critères, et leur mise en œuvre est donc restreinte pour deux raisons principales [9] : D'une part, leurs coefficients ne sont pas simples numériquement, et d'autre part ces filtres ne peuvent avoir de propriétés de symétrie, et donc ne peuvent être de phase linéaire, propriété utile en compression d'images notamment pour éviter certains artefacts. Nous préférons donc l'utilisation de filtres bi-orthogonaux dans notre travail.

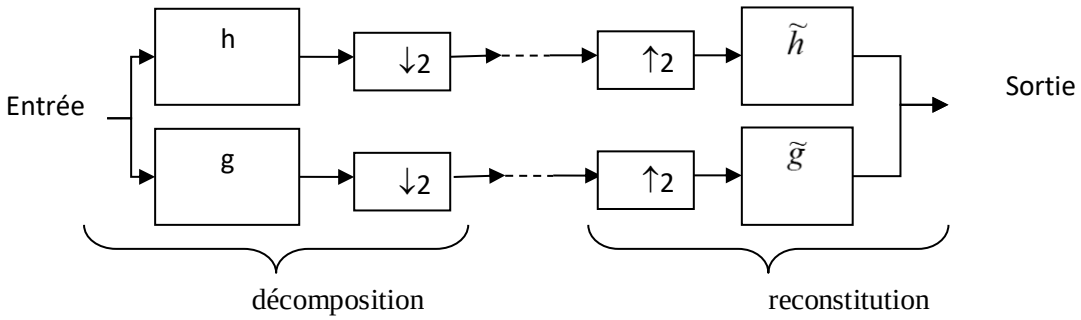


Figure 2.5 : Banc de filtres pour la TOD bi-orthogonale 1-D

Dans le cas bi-orthogonal (figure 2.5), l'analyse est effectuée de la même manière que dans le cas orthogonal [20] (équations 2.8 et 2.9), mais par contre la reconstruction est assurée par les filtres conjugués $\tilde{h}$ et $\tilde{g}$ (équation 2.11).

$$s_{m-1}(n) = \sum_k \tilde{h}(2n-k)s_m(k) + \sum_k \tilde{g}(2n-k)d_m(k) \qquad (2.11)$$

La relation imposée entre les filtres étant :

$$\tilde{g}(n) = (-1)^n h(1-n) \qquad \text{et} \qquad g(n) = (-1)^n \tilde{h}(1-n) \qquad (2.12)$$

2.3.3 Extension à deux dimensions de l'analyse multirésolutions

En deux dimensions, on dispose de trois ondelettes bidimensionnelles respectivement horizontale, verticale et diagonale, permettant le même type de décomposition [8]. On obtient alors, pour une résolution donnée "m", quatre images distinctes (Figure 2.6) : l'approximation de l'image originale à cette résolution (A_m ou LL_m), et les images de détails dans les trois directions horizontale (DA_m ou HL_m), diagonale (DD_m ou HH_m), et verticale (AD_m ou LH_m). En réitérant cette opération pour des résolutions successives sur l'image de l'approximation, on obtient une structure pyramidale en sous bandes. Un exemple de la décomposition de l'image « Lena » est présenté en figure (2.7)

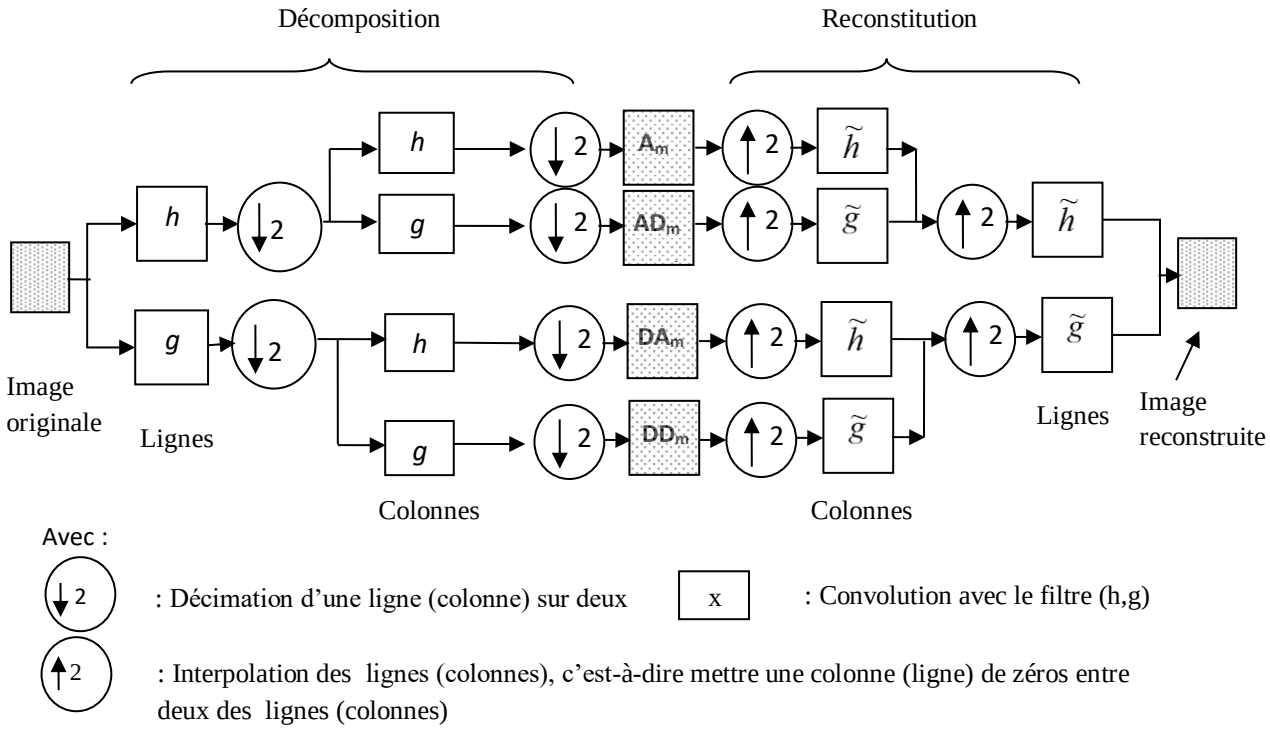


Figure.2.6 : Analyse multi-résolutions en deux dimensions (cas d'ondelettes bi-orthogonales)

Dans le cas où toutes les images (approximation et détails) sont encore décomposées, on obtient un arbre d'images. On parle alors de la décomposition en paquet d'ondelettes 2D [20].

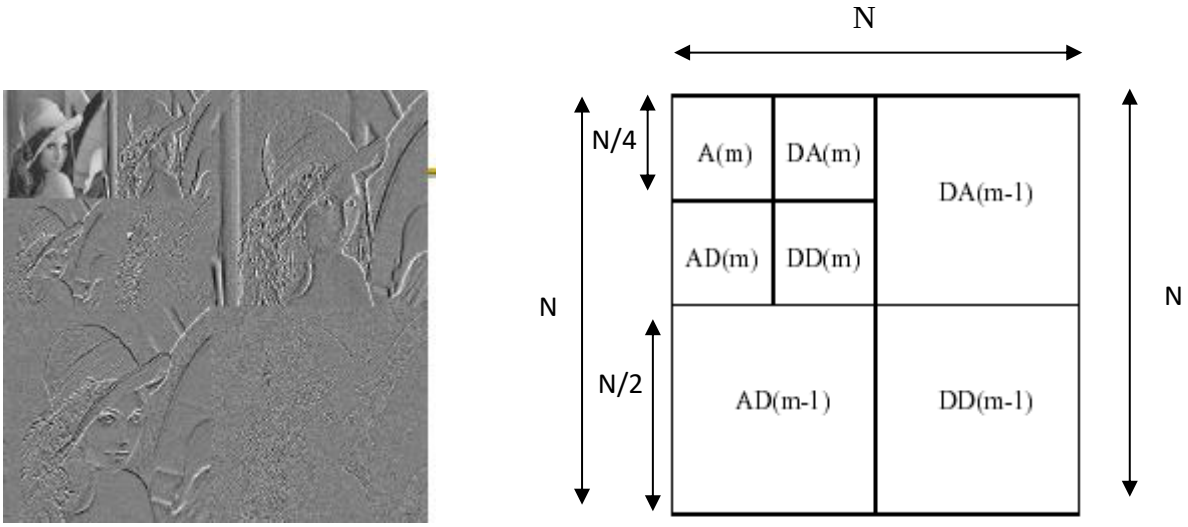


Figure 2.7 : Transformée en ondelettes 2-D de l'image Lena (N=512 pixels) avec la résolution m=2, 7 sous- bandes résultantes.

2-4 Transformée en Bandelettes

La transformée en bandelettes exploite avantageusement les structures géométriques où la fonction a une régularité maximale. Ces bandelettes sont obtenues à partir de la déformation d'ondelettes anisotropes. La base bandelettes décompose l'image le long de vecteurs multi-échelles allongés dans la direction d'un flux géométrique. Ce flux géométrique indique les directions où les niveaux de gris des images ont des variations régulières.

2-4-1 Définition de la géométrie

La géométrie de l'image est représentée par ses contours et ces courbures. Sachant que la tangente à un contour est une direction de la régularité maximale, il en résulte que la régularité peut être mesurée tandis que la position ne peut être déterminée que dans le cas de contours nets [12].

Chaque courbe (contour) "C" est paramétrée par sa abscisse curviligne "s" [28] :

$$C = \{c(s) = (c_1(s), c_2(s)), s\epsilon\mathbb{R}, c_1: \mathbb{R} \longmapsto [0,1] et\ c_2: \mathbb{R} \longmapsto [0,1]\} \quad (2.13)$$

Le flot τ le long du contour (figure 2.8) désigne le vecteur unitaire de cette direction de régularité maximale le long du contour. Il est défini par :

$$\tau(c(s)) = c'(s) = \begin{pmatrix} C'_1 \\ C'_2 \end{pmatrix} \quad (2.14)$$

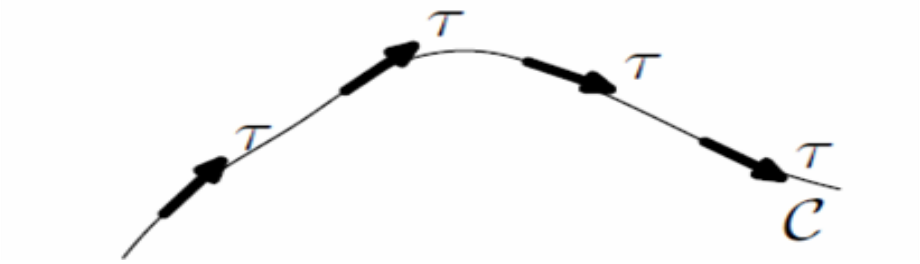


Figure (2.8) : Exemple d'un contour *C* et son flot τ défini par sa tangente [28].

La direction de régularité maximale est simple à déterminer lorsque la fonction f est régulière, il s'agit de la direction orthogonale au gradient [25] :

$$\tau(x) = \frac{\nabla f(x)^{\perp}}{|\nabla f(x)|} \quad (2.15)$$

L'idée centrale dans la construction des bandelettes est de définir la géométrie comme un champ vectoriel ou un flux géométrique pour effectuer une analyse orientée le long des contours. Ce flux indique la direction du déplacement des valeurs de niveau de gris, non pas dans le temps, mais dans l'espace.

Il existe deux versions de construction de la transformée en bandelettes à savoir : les bandelettes de la première et la deuxième génération.

2.4.2. Bandelettes de la première génération

La première génération de la transformée en bandelettes a été proposée par Le Pennec [18]. L'image est d'abord divisée en régions contenant chacune un flot géométrique parallèle. Ces régions sont régulières le long des lignes de flot. Autour d'un contour, le flot est parallèle aux tangentes de la courbe du contour.

La construction d'une base de bandelettes s'effectue sur un domaine "B" où la fonction f considérée est un modèle d'horizon, comme montré à la figure (2.9). La bande "B" contient une unique courbe de singularité [29], que l'on peut approcher par une courbe paramétrée (par exemple horizontalement (figure 2.8)), l'opérateur de déformation est définie comme suit :

$$\forall\, x \in B,\ \ w(x) = (x_1, x_2 - c(x_1)) \tag{2.16}$$

Cet opérateur permet de définir le domaine déformé $WB = w(B)$ (suppose carré), ainsi que la fonction warpée $Wf(x) = f(w^{-1}(x))$

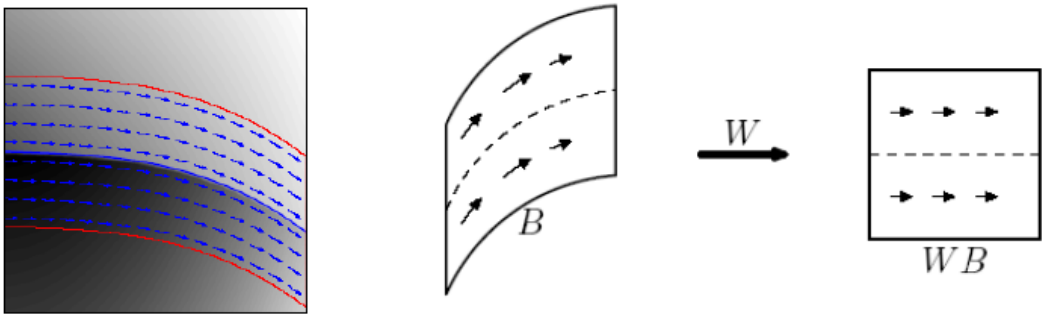


Figure 2.9 : Exemple de modèle d'horizon et déformation du domaine selon un flot géométrique [29].

Par conséquent, pour construire la première génération de bandelettes, il faut tout d'abord diviser l'image *e*n régions *où* le flot géométrique donne le sens de la variation régulière de l'image, ensuite effectuer les tâches énumérées ci-dessous [12-19] :

a- Un ré-échantillonnage qui calcule les valeurs de l'image le long du flot géométrique.

b- Une transformation en ondelettes effectuée par un filtrage en sous-bandes le long du flot ; si la région est régulière (sans flot géométrique), alors la décomposition est effectuée en utilisant une base d'ondelettes bidimensionnelles classique à l'aide d'un banc de filtres de Daubechies bi-orthogonales 9/7. Une transformation en ondelettes discrètes séparables est définie par son ondelette mère ψ et sa fonction d'échelle associée φ, exprimée par [21]:

$$\begin{cases} \varphi_{j,m_1}(x_1)\psi_{j,m_2}(x_2) \\ \varphi_{j,m_1}(x_1)\psi_{j,m_2}(x_2) \\ \psi_{j,m_1}(x_1)\psi_{j,m_2}(x_2) \end{cases} ; j\epsilon\mathbb{Z}, (m_1, m_1)\epsilon\mathbb{Z}^2 \qquad (2.17)$$

Autrement, si la région a un flot géométrique, la décomposition est effectuée sur une $W(\overrightarrow{\tau})$. La base orthonormée déformée pour un flot horizontal parallèle est formellement donnée par :

$$\begin{cases} \psi_{j,m_1}(x_1 - c_i(x_2))\varphi_{j,m_2}(x_2) \\ \varphi_{j,m_1}(x_1 - c_i(x_2))\psi_{j,m_2}(x_2) \\ \psi_{j,m_1}(x_1 - c_i(x_2))\psi_{j,m_2}(x_2) \end{cases} ;(j, m_1, m_2)\epsilon I_{w\mathrm{B}} \qquad (2.18)$$

Par conséquent, pour un flux horizontal l'image déformée est alors :

$$Wf(x_1, x_2) = f(x_1, x_2 + c(x_1)) \qquad (2.19)$$

c- Une "bandeletisation" qui transforme les coefficients des ondelettes déformées en coefficients bandelettes en exploitant la régularité de la fonction le long du flot.

Les bandelettes de la première génération utilisent avantageusement les structures géométriques des images. Cependant, elles ne sont pas directement définies dans le cas discret et elles n'offrent pas une représentation multi résolution de la géométrie. A cet effet, Peyré et al [18] ont défini la seconde génération de bandelettes.

2.4.3. Bandelettes de la seconde génération

La construction des bandelettes a été raffinée par Mallat et Peyré [30] pour obtenir des bases orthonormées adaptées aux fonctions géométriquement régulières. Ces bases de bandelettes de seconde génération sont définies à partir d'une représentation en ondelettes en ajoutant une étape d'une transformation géométrique sur les coefficients en ondelettes-eux-mêmes.

La transformée en bandelettes [22-24] est une transformée adaptative de l'image, qui s'effectue en Cinq étapes principales :

a- Transformation en ondelettes TO_M de l'image M.

b- Segmentation de chaque sous-bande en carrés dyadiques.

c- Détermination la meilleure géométrie qui définit la direction de la régularité géométrique dans chaque carré dyadique. La meilleure géométrie est celle qui minimise le lagrangien L comme suit :

$$L=|| TO_M - A_M||^2+mT^2 \qquad (2.20)$$

Tel que:

T : le seuil.

A_M: l'approximation de TO_M avec une base orthogonale.

m : le nombre des coefficients de l'ondelette supérieurs au seuil T.

d- Une projection orthogonale 1D est effectuée à la géométrie spécifiée pour définir un signal discret 1D S_d.

e- Une transformation en ondelettes discrète de Haar 1D est appliquée au signal $1DS_d$ donnant les coefficients de bandelettes b_k.

L'algorithme de la représentation en bandelettes de seconde génération d'une image est illustré en figure (2.12).

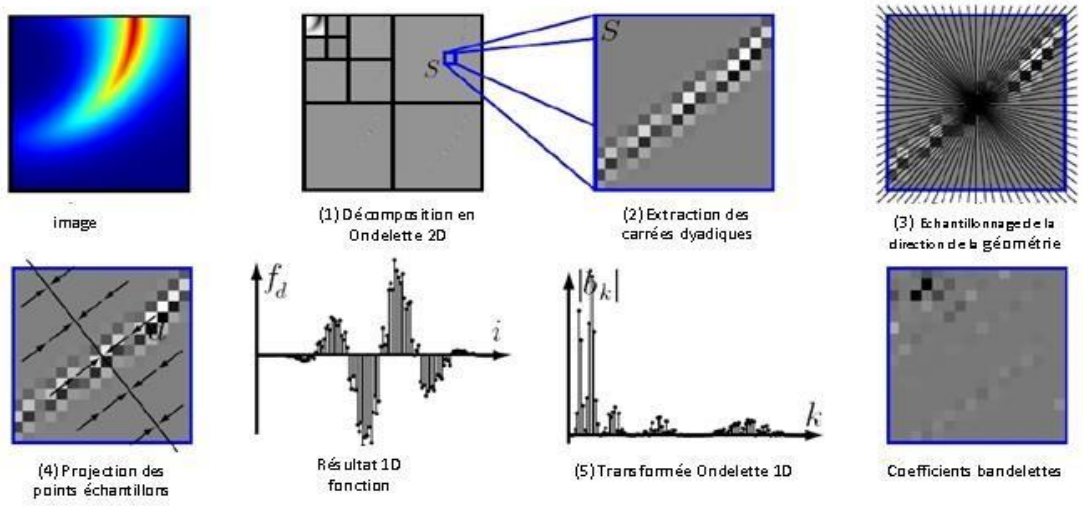


Figure 2.10: Algorithme de la transformée en bandelettes de seconde génération.

2.5. Conclusion

De nouvelles représentations d'images ont été développées de façon à pallier ces défauts de la transformée en ondelettes tout en conservant ses avantages, les bases d'ondelettes orthogonales très efficaces pour les zones régulières, les textures homogènes et les singularités ponctuelles. Cependant, elles ne sont pas optimales pour les images car elles ne peuvent exploiter la régularité de nature géométrique des contours.

La représentation en bandelettes donne efficacement les contours des images grâce aux bases orthogonales de bandelettes ainsi que les zones de régularité homogène sous la forme d'un résidu décomposé en ondelettes. Le cadre orthonormé permet d'obtenir des résultats théoriques sur les capacités d'approximation de cette représentation montrant leur intérêt par rapport aux bases d'ondelettes bidimensionnelles séparables. Cependant, il reste encore à confirmer ces éléments par des résultats numériques de compression.

3-1 Introduction *:*

Le but de ce chapitre est de présenter l'algorithme SPECK et ensuite notre algorithme SPECK modifié afin de montrer l'amélioration apportée.

L'utilisation de la transformée en ondelettes dans l'étape de transformation offre une analyse multirésolutions de l'image. Cette propriété est très intéressante pour la compression progressive des images numériques.

Après un bref rappel sur la méthode originale de quantification développée par Islam et Pearlman (SPECK), nous proposons une optimisation de l'algorithme SPECK nommée SPECK modifié (Modified- SPECK).

3-2 Principe de l'algorithme SPECK :

L'algorithme SPECK consiste à effectuer des répartitions successives de l'image transformée en blocs. Lorsque tous les coefficients d'ondelettes ont des valeurs inférieures à un seuil donné, on obtient un bloc de zéros (bloc insignifiant).

On calcule le seuil T_n comme suit : $T_n=2^n$, "n" étant la partie entière de logarithme népérien du maximum de toutes les valeurs des coefficients d'ondelettes "$C_{i,j}$" de l'image de taille "NxM".

$$n=\log_2 (|\max (C_{i,j})|) \text{ avec } i \in [1:N] \ et \ j \in [1,M] \qquad (3.1)$$

On dit qu'un coefficient de bandelette $C_{i,j}$ est signifiant par rapport à un seuil T_n s'il vérifie la relation suivante :

$$\mathcal{S}_{C_{i,j}} = 1 \Longrightarrow T_n \leq |C_{i,j}| < T_{n+1} \quad \text{avec } T_n=2^n \qquad (3.2)$$

On définit la signifiance d'un bloc "B" de coefficients d'ondelettes $C_{i,j}$ par rapport à un seuil T_n :

$$\mathcal{S}_n(B) = \begin{cases} 1 & si\ 2^n \leq (max_{(i,j)\epsilon B} \left|C_{i,j}\right| < 2^{n+1} \\ 0 & ailleurs \end{cases} \qquad (3.3)$$

Quand le résultat de test est "1", on dit que ce bloc est signifiant par rapport au seuil T_n ; autrement dit, un bloc est insignifiant si tous ses coefficients sont insignifiants. Par contre, le bloc est signifiant si on trouve au moins un coefficient signifiant dans le bloc.

3-3- Etapes de l'algorithme SPECK :

Pour l'algorithme SPECK, on détermine deux listes [3-10-16] :

- La liste des ensembles insignifiants (LIS) qui contient des blocs de type S de tailles variables qui n'ont pas encore été trouvés signifiants par rapport au seuil courant T_n .
- La liste des pixels signifiants (LSP) qui contient les pixels ayant été examinés signifiants par rapport au même seuil.

3-3-1- Initialisation :

Dans l'étape de l'initialisation, l'image "X" est partitionnée en deux ensembles, un ensemble "S" qui prend les coefficients de la sous-bande la plus basse et un ensemble "I" qui prend le reste de l'image tel que : I=X-S. La liste des pixels signifiants est initialisée vide $(LSP) = \emptyset$.

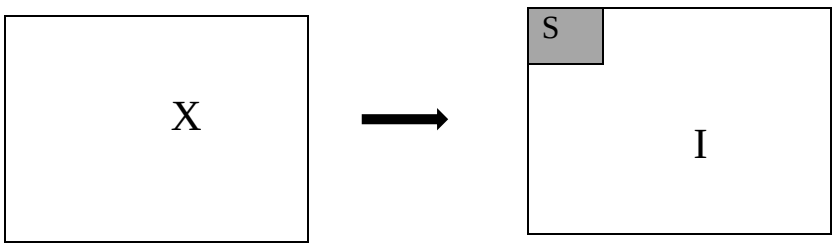


Figure 3.1. : Partition de l'image X en deux ensembles S et I

3-3-2 Etape de sortie de bits :

La sortie de bits dans l'algorithme SPECK est effectuée comme suit :

- ✓ Pour chaque ensemble $S \in LIS$:
  - Process $\mathcal{S}(S)$
  - Process I ()

Process $\mathcal{S}(S)$

{

- Sortie de $\mathcal{S}_n(S)$
- if $\mathcal{S}_n(S) = 1$

- if S est un coefficient, on code son signe et on le met à la sortie , puis, on ajoute S à LSP.
- else Code S(S).
- if $\in LIS$, on enlève S de LIS.

- else

- if $S \notin LIS$, on ajoute S à LIS.

}

Code $\mathcal{S}(S)$

{

- Partition S en quatre sous-ensembles égaux O(S) (voir figure 3.2)
- Pour chaque O(S)

- Sortie de $\mathcal{S}_n(O(S))$
- if $\mathcal{S}_n(O(S)) = 1$
  - if S est un coefficient, on code son signe et on le met à la sortie , puis, on ajoute O(S) à LSP.
  - else Code S(O(S))
- else

  on ajoute O(S) à LIS.

}

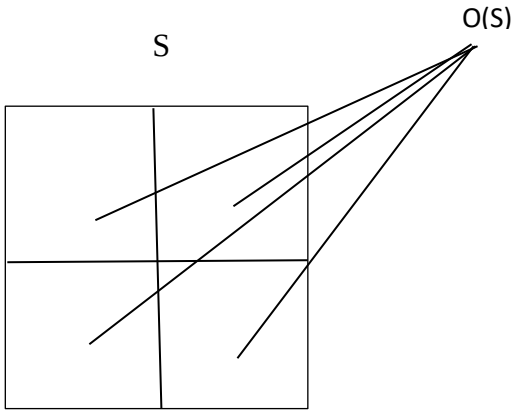


Figure 3.2 : Partition de l'ensemble S

Process I( )

{

- Sortie de $\mathcal{S}_n(I)$
- if $\mathcal{S}_n(I) = 1$

- Code I( ).

}

Code **I**()

{

- Partition I en quatre sous-ensembles trois S et un seul I (figure 3.3)
- Pour chacun des trois ensembles S

- Process S(S)

- Process I( )

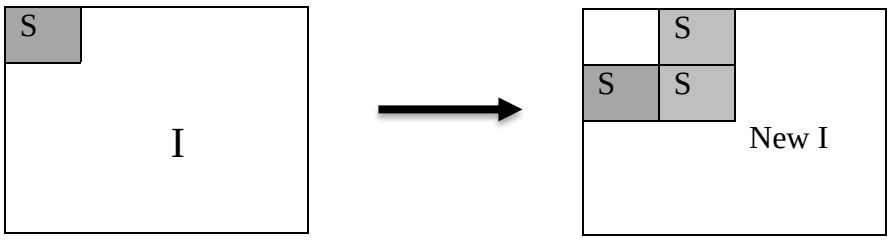


Figure 3.3 : Partition de l'ensemble I

3-3-3- Phase de raffinement :

Pour tous les coefficients signifiants déterminés dans l'étape précédente, on émet le bit d'ordre (n) pour augmenter la précision des valeurs significatives transmises, l'algorithme correspondant est le suivant [14]:

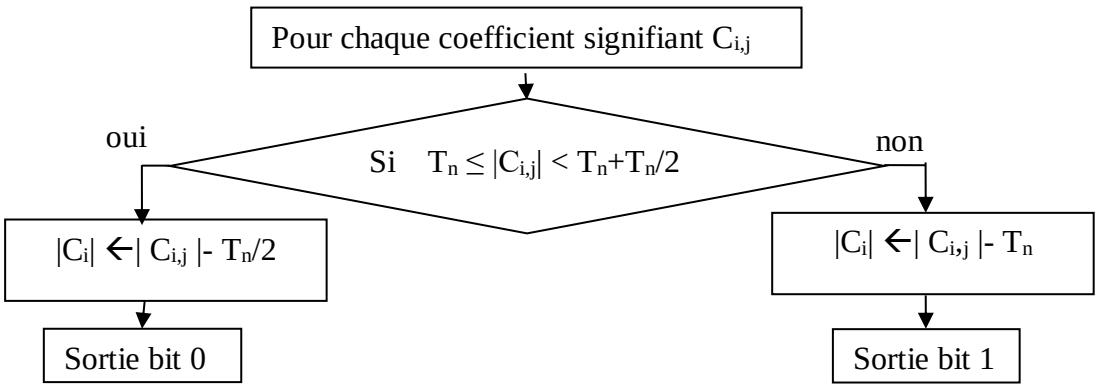


Figure 3.4 : Algorithme de l'étape de raffinement des coefficients signifiants

Le principe de l'algorithme de raffinement est l'utilisation d'un quantificateur scalaire uniforme défini par rapport au seuil T_n . Globalement les coefficients significatifs sont compris dans l'intervalle $[T_n, 2T_n[$. Le bit d'ordre **(n)** du coefficient prend "0" si le coefficient se trouve dans l'intervalle $[T_n, 3T_n/2[$ tandis que ce bit prend "1" si le coefficient se trouve dans l'intervalle $[3T_n/2, 2T_n[$.

A chaque nouveau seuil, nous réalisons un raffinement des intervalles d'incertitude en les divisant par deux. Ainsi, l'indice de quantification est mis à "0" pour les coefficients appartenant à la première moitié de l'intervalle alors que l'indice de quantification "1" est utilisé pour la seconde moitié (figure 3.5).

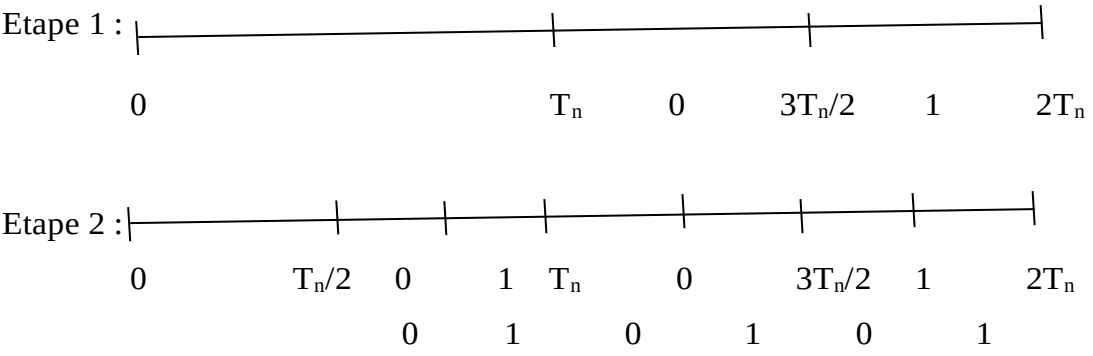


Figure 3.5 : Principe du raffinement successif

L'étape de raffinement sert à rendre les coefficients signifiants à l'étape courante négligeables à la prochaine étape pour augmenter les chances d'avoir des " zerotree" .

3-3-4- Étape de la quantification :

L'étape de sortie de bits sur résidu de l'image est obtenue en décrémentant **n** par 1. Nous étirons le processus jusqu'à ce que le critère de qualité de l'image soit atteint ou que le nombre de bits transférables soit dépassé.

3-4 Exemple d'application du SPECK:

La figure (3.6) montre une petite matrice, à titre d'exemple, de taille 8 x 8 coefficients. Le codage des blocs et des coefficients se fait avec une seule itération.

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

Figure 3.6 : Exemple de matrice de 8 x 8 coefficients d'ondelettes

Le seuil T_n est calculé comme suit :

- ✓ N (partie entière de $\log_2(63)$)=5
- ✓ $T_5=2^5=32$

Etape	Coefficient ou ensemble testés	Bit de sortie	action	Listes de controle
N=5 S=$S^1(0,0)$ I=reste				LIS={(0,0)1} $LSP = \emptyset$
	$S^1(0,0)$	1	Partitionnement en quatre Ajout à LIS	LIS={(0,0)0,(0,1)0,(1,0)0,(1,1)0} $LSP = \emptyset$
	(0,0)	1+	(0,0) à LSP	LIS={(0,1)0,(1,0)0,(1,1)0} $LSP = \{(0,0)\}$
	(0,1)	1-	(0,1) à LSP	LIS={(0,0)0,(1,1)0} $LSP = \{(0,0),(0,1)\}$
	(1,0)	0	Non	
	(1,1)	0	Non	

Test I	S(I)	1	Partitionnement en 3S, new I	
	S^1(0,2)	1	Partitionnement en quatre. Ajout à LIS (0)	LIS={(1,0)0,(1,1)0,(0,2)0,(0,3)0,(1,2)0,(1,3)0}
	(0,2)	1+	(0,2) à LSP	LIS={(1,0)0,(1,1)0,(0,3)0,(1,2)0,(1,3)0} $LSP = \{(0,0),(0,1),(0,2)\}$
	(0,3)	0	Non	
	(1,2)	0	Non	
	(1,3)	0	Non	
	S^1(2,0)	0	Ajout à LIS(1)	LIS={(1,0)0,(1,1)0,(0,3)0,(1,2)0,(1,3)0,(2,0)1}
	S^1(2,2)	0	Ajout à LIS (1)	LIS={(1,0)0,(1,1)0,(0,3)0,(1,2)0,(1,3)0, (2,0)1,(2,2)1}
Test I	S(I)	1	Partitionnement à 3S	
	S^2(0,4)	0	Ajout à LIS (1)	LIS={(1,0)0,(1,1)0,(0,3)0,(1,2)0,(1,3)0, (2,0)1,(2,2)1,(0,4)2}
	S^2(4,0)	1	Partitionnement en quatre. Ajout à LIS (1)	LIS={(1,0)0,(1,1)0,(0,3)0,(1,2)0,(1,3)0, (2,0)1,(2,2)1,(4,0)1,(4,2)1,(6,0)1,(6,2)1,(0,4)2}
	S^1(4,0)	0	Non	
	S^2(4,2)	1	Partitionnement en quatre. Ajout à LIS (0)	LIS={(1,0)0,(1,1)0,(0,3)0,(1,2)0,(1,3)0, (4,2)0,(4,3)0,(5,2)0,(5,3)0,(2,0)1,(2,2)1, (4,0)1,(4,2)1, (6,0)1,(6,2)1,(0,4)2}
	(4,2)	0	Non	
	(4,3)	1+	Déplacer (4,3)à LSP	LIS={(1,0)0,(1,1)0,(0,3)0,(1,2)0,(1,3)0, (4,2)0,(4,3)0,(5,2)0,(5,3)0,(2,0)1,(2,2)1, (4,0)1,(4,2)1, (6,0)1,(6,2)1,(0,4)2} $LSP = \{(0,0),(0,1),(0,2),(4,3)\}$
	(5,2)	0	Non	
	(5,3)	0	Non	
	S^1(6,0)	0	Non	
	S^1(6,2)	0	Non	
Fin sortie n=5	S^2(4,4)	0	Ajout à LIS (2)	LIS={(1,0)0,(1,1)0,(0,3)0,(1,2)0,(1,3)0, (4,2)0,(4,3)0,(5,2)0,(5,3)0,(2,0)1,(2,2)1, (4,0)1,(4,2)1, (6,0)1,(6,2)1,(0,4)2,(4,4)2} $LSP = \{(0,0),(0,1),(0,2),(4,3)\}$

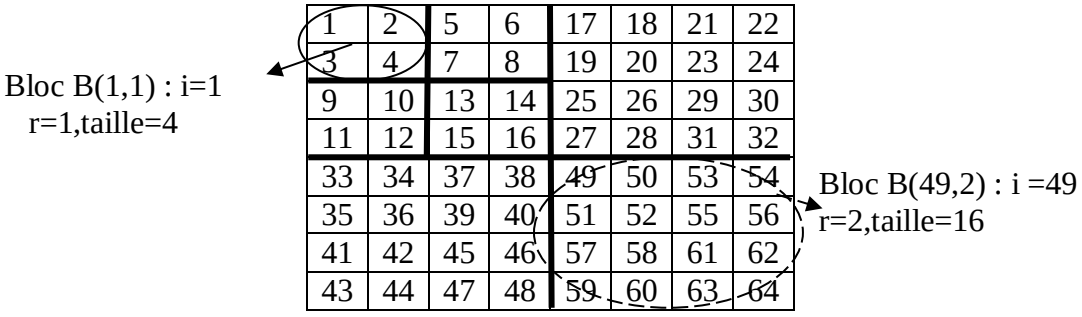
Tableau 3.1: Exemple de codage d'image par l'algorithme SPECK

3-5 Algorithme proposé SPECK modifié

3-5-1-Scan des coefficients d'ondelettes :

Les coefficients de l'image transformée en ondelettes sont d'abord indexés par un scan zigzag selon Morton Scan [6]. C'est-à-dire, nous associons à chaque coefficient "C", un indice 'i' (noté C_i) et à chaque bloc "B", deux indices 'i' et 'r' (noté B(i, r) voir figure 3.7), où 'i' représente l'indice du premier coefficient du bloc et l'indice 'r' représente le rang du bloc (par exemple : B(1,1) est le bloc de rang '1' avec l'indice du premier coefficient '1').

Le tableau 3.2 représente les tailles de blocs selon l'indice 'r'. Dans l'exemple précédant, le bloc B(1,1) est de taille (2^r)*(2^r)=4, et le bloc B(49,2) est de taille (2^r)*(2^r)=16.



1	2	5	6	17	18	21	22
3	4	7	8	19	20	23	24
9	10	13	14	25	26	29	30
11	12	15	16	27	28	31	32
33	34	37	38	49	50	53	54
35	36	39	40	51	52	55	56
41	42	45	46	57	58	61	62
43	44	47	48	59	60	63	64

Figure 3.7 : Indexation des coefficients d'ondelettes en Morton scan et exemples de tailles de blocs.

$r=\log_2$(taille de bloc)	0	1	2	3	4	5	6	7	8	9
Taille de bloc (2^r)*(2^r)	1*1	2*2	4*4	8*8	16*16	32*32	64*64	128*128	256*256	512*512

Tableau 3.2 : Différents indices de blocs et leurs tailles correspondantes

On peut remarquer qu'un bloc B(i,r) contient les coefficients C_k: $k \epsilon [i \;\; i+(2^{(2*r)})-1]$.

3-5-2 Initialisation :

Dans une image transformée en ondelettes, nous aurons un nombre de coefficients égal à (taille_image x taille_image) coefficients. Nous associons à chaque coefficient d'ondelettes C_i, un pointeur noté Pointeur(i).

Au début de l'algorithme :

- ✓ La liste des coefficients signifiants est vide ($LSP = \emptyset$).
- ✓ La valeur initiale de i est 1.
- ✓ La valeur initiale de Pointeur(1) est log2(taille_image).

3-5-3 Codage des blocs et des coefficients d'ondelettes :

Le test de signifiance d'un bloc varie en fonction de sa taille (figure 3.8). Il y a trois types de test de signifiance :

- Test de signifiance de blocs contenant plus de quatre coefficients.
- Test de blocs de quatre coefficients
- Test de blocs d'un seul coefficient.

Le test de signifiance se déroule du plus grand bloc jusqu'au plus petit bloc (figure 2):

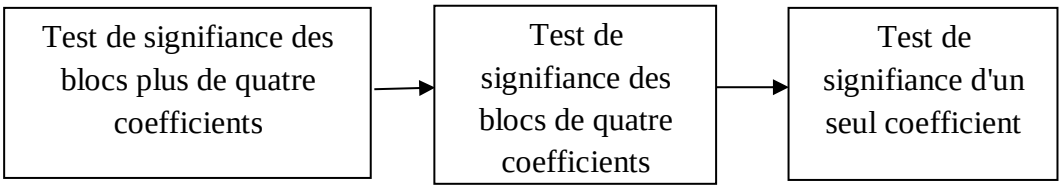


Figure 3.8 : Test de signifiance selon la taille de blocs.

Le bloc signifiant B(i,r) supérieur à quatre coefficients est partitionné en quatre blocs égaux B(i,r-1), B(i+2^(2*r),r-1) , B(i+2*(2^(2*r)),r-1) , B(i+3*(2^(2*r)),r-1) ; par contre le bloc insignifiant ne sera pas divisé .

Dans le cas des blocs signifiants de quatre coefficients dont trois coefficients insignifiants, nous les codons par deux bits représentant la position du coefficient signifiant dans le bloc.

Le coefficient d'ondelettes C_i est mis à LSP s'il est signifiant. Le signe de C_i est codé par le bit '1' s'il est négatif et par le bit '0' s'il est positif.

Le test de signifiance et le codage des blocs et des coefficients se font selon les valeurs de pointeurs (tableau 3.3).

Valeur du Pointeur(i)	Interprétation
de 2 jusqu'à log2(taille_image)	Le test de signifiance du bloc qui contient plus de quatre coefficients commence par le coefficient d'ondelettes C_i et la taille de bloc est (Pointeur(i) x Pointeur(i))
1	Le test de signifiance du bloc de quatre coefficients commence par le coefficient d'ondelettes C_i
3/4	C_i est l'un de trois coefficients insignifiants qui se trouve dans un bloc signifiant de quatre coefficients (le bloc contient un seul coefficient signifiant).
1/4	C_i est le seul coefficient signifiant dans un bloc signifiant de quatre coefficients.
0	Test de signifiance d'un seul coefficient C_i.
Inf	C_i est signifiant, on le met à LSP.

Tableau 3.3 : Interprétation des valeurs de pointeurs

L'algorithme suivant résume l'opération de codage des blocs et des coefficients d'ondelettes pour un seuil $T_n=2^n$:

La valeur initiale de n est déterminée par l'équation (3.1).
while n>0
i=1
Pointeur(1)= log2(taille_ image)
while i<= (taille_image x taille_image)

/*Codage des blocs plus de quatre coefficients B(i, r) avec r=Pointeur(i)*/
if Pointeur(i)>1 & Pointeur(i) <= log2(taille_ image)

```
                    if  le bloc est insignifiant
                            sortie  du bit '0'
                            i=i+2^(2* Pointeur (i))
                    else
                            sortie  du bit '1'
                            Pointeur(i)= Pointeur(i)-1
                                    for j=1:3
                                        Pointeur(i+j*2^(2* Pointeur(i)))= Pointeur(i)
                                    end
                     end

 /*Codage des blocs de quatre coefficient B(i, r)  avec r=Pointeur(i)=1*/
elseif  Pointeur (i)=1
                    if le bloc est insignifiant
                            sortie  du bit '0'
                            i=i+4
                    else
                            sortie  du bit '1'
                        if    un seul coefficient signifiant parmi  les quatre
                                    sortie  du bit '1'
                                if  le premier est signifiant
                                        sortie  du bit '00'
                                        Pointeur (i)=1/4
                                        Pointeur (i+1)=Pointeur (i+2)=Pointeur
    (i+3)=3/4
                                elseif  le deuxième est signifiant
                                        sortie  du bit '01'
                                        Pointeur (i+1)= 1/4
                                        Pointeur (i)=Pointeur (i+2)=Pointeur (i+3)= 3/4
                                 elseif  le troisième est signifiant
                                        sortie  du bit '10'
                                        Pointeur (i+2)=1/4
                                        Pointeur (i)= Pointeur (i+1)=Pointeur (i+3)=
    3/4
                                else
                                        sortie  du bit '11'
                                        Pointeur (i)=Pointeur (i+1)=Pointeur (i+2)= 3/4
                                        Pointeur (i+3)= 1/4
                                end
                        else
                            sortie  du bit '0'
                            Pointeur (i)=Pointeur (i+1)=Pointeur (i+2)=Pointeur (i+3)=0
                        end
                end

                            /*codage des coefficients*/
elseif   Pointeur (i)=0
                        if  Ci est signifiant
                            sortie  du bit '1'
                            Pointeur (i)=Inf
                            On met Ci  à LSP
                                    if  Ci  <0
                                       sortie  du bit '1'
                                    elesif
                                       sortie  du bit '0'
                                    end
                        elseif
                           sortie  du bit '0'
                        end
                        i=i+1
```

```
            elseif  Pointeur (i)=1/4
                        Pointeur (i)=Inf
                         On met Ci  à LSP
                                        if  Ci  <0
                                             sortie  du bit '1'
                                        elseif
                                             sortie  du bit '0'
                                        end
                         i=i+1

            elseif  Pointeur (i)=3/4
                         Pointeur (i)=0       /*pour tester Ci par rapport aux seuils inférieurs*/
                          i=i+1
            else
                 i=i+1
             end
end
n=n-1
end
```

3-6 Exemple d'application du SPECK modifié :

Le codage des blocs et des coefficients par l'algorithme de SPECK modifié de la matrice de la figure 3.6 se fait avec une seule itération.

La valeur initiale de i est 1 et la valeur initiale du pointeur(1) est Pointeur(1)=$\log_2(8)$=3, le tableau 3.4 résume l'opération de sortie des bits.

I	Pointeur (i)	Sortie de bits	Action	Modification de i et Pointeur(i)
1	3	1	Bloc B (1,3) est signifiant	Pointeur(i)= Pointeur(i)-1
	2		Partition du B(1,3) en 4 blocs: B(1,2), B(17,2), B(33,2), B(49,2)	Pointeur(i+j*2^(2* Pointeur(i)))=Pointeur(i) j=1:3 Pointeur(17)=Pointeur(33)=Pointeur(49)=2
1	2	1	B (1,2) est signifiant	Pointeur(i)= Pointeur(i)-1
	1		Partition du bloc B(1,2) en 4 blocs :B(1,1), B(5,1), B(9,1),bloc(13,1)	Pointeur(i+j*2^(2* Pointeur(i)))=Pointeur(i) j=1:3 Pointeur(5)= Pointeur(9)= Pointeur(13)=1
1	1	1	B (1,1) est signifiant	
1	1	0	Il y a plus d'un coefficient signifiant	Pointeur(1)=Pointeur(2)=Pointeur(3)=Pointeur(4)=0
1	0	1	Coefficient C_1 est signifiant $C_1 \epsilon$ LSP	
1	inf	0	C_1 est positif	i=i+1
2	0	1	Coefficient C_2 est signifiant $C_2 \epsilon$ LSP	
2	inf	1	C_2 est négatif	i=i+1
3	0	0	Coefficient C_3 est insignifiant	i=i+1

4	0	0	Coefficient C_4 est insignifiant	i=i+1
5	1	1	Bloc B (5,1) est signifiant	
5	1	1	un seul coefficient signifiant parmi les quatre dans le B (5,1)	
5	1	00	le premier coefficient est signifiant et les autres insignifiants	Pointeur (5)=1/4 Pointeur (6)=Pointeur (7)=Pointeur (8)=3/4
5	1/4		Coefficient C_5 est signifiant $C_5 \epsilon$ LSP	
5	inf	0	C_5 est positif	i=i+1
6	0			i=i+1
7	0			i=i+1
8	0			i=i+1
9	1	0	B(9,1) est insignifiant	i=i+2^(2* Pointeur (i)))=13
13	1	0	B(13,1) est insignifiant	i= i+2^(2* Pointeur (i)))=17
17	2	0	B(17,2) est insignifiant	i=33
33	2	1	B(33,2) est signifiant	Pointeur(i)= Pointeur(i)-1
33	1		Partition du bloc B(33,2) en 4 blocs: B(33,1), B(37,1), B(41,1),B(45,1)	Pointeur(i+j*2^(2* Pointeur(i)))=Pointeur(i) j=1:3 Pointeur(37)= Pointeur(41)= Pointeur(45)=1
33	1	0	B(33,1) est insignifiant	i=37
37	1	1	B(37,1) est signifiant	
37	1	1	un seul coefficient signifiant parmi les quatre dans le B(37,1)	
37	1	01	le deuxième coefficient est signifiant et les autres insignifiants	Pointeur (37)=Pointeur (39)=Pointeur (40)=3/4 Pointeur (38)=1/4
37	0			i=i+1
38	1/4		Coefficient C_{38} est signifiant $C_{38} \epsilon$ LSP	
38	inf	0	C_{38} est positif	i=i+1
39	0			i=i+1
40	0			i=i+1
41	1	0	Bloc(41,1) est insignifiant	i=45
45	1	0	Bloc(45,1) est insignifiant	i=49
49	2	0	Bloc(49,2) est insignifiant	

Tableau 3.4 : sortie de bits de codage de la matrice de la figure 3.6 par SPECK modifié

3.7 Décodeur SPECK modifié

Dans l'opération de décodage, nous différencions entre un bit et deux bits de la manière suivante : Dans le cas général, nous prenons bit par bit et nous faisons le décodage. Dans le cas où le Pointeur(i)=1 et le premier et le deuxième bit sont "1", nous prenons le troisième et le quatrième bit ensemble. Si par

exemple la séquence (1 0 1 1 1 0 1 . . .) représente les bits de sortie de compression d'une image de taille 8x8, l'opération de décodage est détaillée dans le tableau 3.5.

Pointeur(1)= $\log_2(8)=3$				
code	i	Pointeur(i)	Action	Modification de i et Pointeur(i)
1	1	3	Partition du bloc B(1,3) en 4 blocs: B(1,2), B(17,2), B(33,2),B(49,2)	Pointeur(i)= Pointeur(i)-1=2 Pointeur(i+j*2^(2* Pointeur(i)))=Pointeur(i) j=1:3 Pointeur(17)= Pointeur(33)= Pointeur(49)=2
0	1	2	i=i+2^(2*Pointeur(i))=17	
1	17	2	Partition du bloc B(17,2) en 4 blocs: B(17,1), B(21,1), B(25,1),B(29,1)	Pointeur(i)= Pointeur(i)-1=1 Pointeur(i+j*2^(2* Pointeur(i)))=Pointeur(i) j=1:3 Pointeur(21)= Pointeur(25)= Pointeur(29)=1
1	17	1		
1	17	1		
0	17	1		Pointeur (17)=Pointeur (19)=Pointeur (20)=3/4 Pointeur (18)=1/4
1				

Tableau 3.5 : Exemple d'opération de décodage

3.8 Comparaison entre les deux codeurs SPECK et SPECK modifié :

En comparant les sorties de bits de codage de la matrice des coefficients d'ondelettes de la figure 3.6 par les deux codeurs SPECK et SPECK modifié, on obtient les valeurs illustrées dans le tableau 3.6.

	Seuil	SPECK	SPECK modifié
Nombre de bits de sortie	32	29	28
	16	53	52
	8	118	114
	4	210	208
	2	292	290

Tableau 3.6: Comparaison entre les deux codeurs SPECK et SPECK modifié

3-9 Conclusion

Dans ce chapitre, nous avons proposé un algorithme de compression d'images le SPECK modifié, basé sur le même principe que l'algorithme SPECK. Cet algorithme est basé sur l'utilisation de deux bits au lieu de quatre employés dans l'algorithme SPECK original pour coder les blocs signifiants de quatre coefficients dont trois insignifiants. A cet effet, nous avons obtenu une quantité d'informations inférieure par rapport à l'algorithme SPECK original.

4-1 Introduction

Après l'étude de l'algorithme SPECK et la proposition d'une optimisation de ce dernier, nommée SPECK modifié (cf. chapitre 3); Nous effectuons dans ce chapitre une étude détaillée des résultats obtenus par les deux algorithmes (SPECK et SPECK modifié) appliqués sur des images de test fréquemment utilisées dans la littérature.

4-2 Paramètres de validation

En fonction de l'application recherchée, l'algorithme de compression doit pouvoir vérifier un certain nombre de critères de qualité; entre autres, on peut citer : le taux de compression, le rapport de dégradation. D'autres paramètres sont très utilisés dans le domaine de la compression telles que la quantité d'information et l'entropie. Ces paramètres permettent de caractériser l'image originale et d'estimer l'efficacité du codage (d'une manière globale) avant le calcul du taux de compression.

Nous rappelons brièvement les différents critères utilisés.

4-2-1 Taux de compression (TC)

Le taux de compression est le rapport entre le nombre de bits nécessaire pour stocker l'image d'origine et celui nécessaire pour stocker l'image compressée (exprimé en pourcent). Ce taux est obtenu par la formule suivante :

$$TC(\%) = 100 - \left(\frac{Taille\_image\_compressé}{taille\_image\_originale} \right) * 100 \qquad (4.1)$$

4-2-2 Mesure de la qualité d'image compressée

Dans un système de compression d'images, l'image compressée est toujours comparée à l'originale pour déterminer son rapport de ressemblance. Le critère quantitatif de comparaison le plus utilisé est l'Erreur Quadratique Moyenne "EQM", ou son équivalent le Rapport Signal/Bruit (Crête Peak Signal to Noise Ratio "PSNR"). L'Erreur Quadratique Moyenne est définie par :

$$EQM = \frac{1}{M * N} \sum_{m=1}^{M} \sum_{n=1}^{N} \left[x(m,n) - \hat{x}(m,n) \right]^2 \quad (4.2)$$

Le PSNR est défini par**:**

$$\text{PSNR} = 10 \log_{10} \frac{255^2}{EQM} \quad (4.3)$$

Où **:** (M × N) est la taille de l'image à coder.

x(m,n) et $\hat{x}$(m,n) sont les intensités des pixels de l'image d'origine et l'image codée respectivement.

4-2-3 Temps de calcul

La contrainte du temps est un facteur essentiel dans l'évaluation des performances de toute méthode de compression, elle revient à calculer le temps pris par la compression et la décompression des images. Cette contrainte est plus au moins imposée selon l'application visée par la compression (transmission ou archivage). En effet, il serait dommage, dans une application de transmission, que le temps gagné par une réduction de la taille des données à transmettre soit inférieur au temps passé à la compression décompression. Cette qualité sera cependant moins cruciale dans des applications visant l'archivage de données [9].

4-3 Images de test

Pour notre application, nous avons utilisé les images suivantes:

- ✓ image Penny en niveau de gris codée sur 8 bits et de taille 128x128 pixels (figure 4-1).
- ✓ image Woman en niveau de gris codée sur 8 bits et de taille 256x256 pixels (figure 4-2).
- ✓ image Hoed en niveau de gris codée sur 8 bits et de taille 256x256 pixels (figure 4-3).
- ✓ image Cameraman en niveau de gris codée sur 8 bits et de taille 256x256 pixels (figure 4-4)
- ✓ image Goldhill en niveau de gris codée sur 8 bits et de taille 256x256 pixels (figure 4-5).

- ✓ image Barbara en niveau de gris codée sur 8 bits et de taille 512x512 pixels (figure 4-6).
- ✓ image Baboon en niveau de gris codée sur 8 bits et de taille 512x512 pixels (figure 4-7).
- ✓ image Lena en niveau de gris codée sur 8 bits et de taille 512x512 pixels (figure 4-8).
- ✓ image Peppers en niveau de gris codée sur 8 bits et de taille 512x512 pixels (figure 4-9)

Figure 4.1 : image originale Penny 128x128 pixels

Figure 4.2 : image originale Woman 256x256 pixels

Figure 4.3 : image originale Hoed 256x256 pixels

Figure 4.4 : image originale Cameraman 256x256 pixels

Figure 4.5 : image originale Goldhill 256x256 pixels

Figure 4.6 : image originale Barbara 512x512 pixels

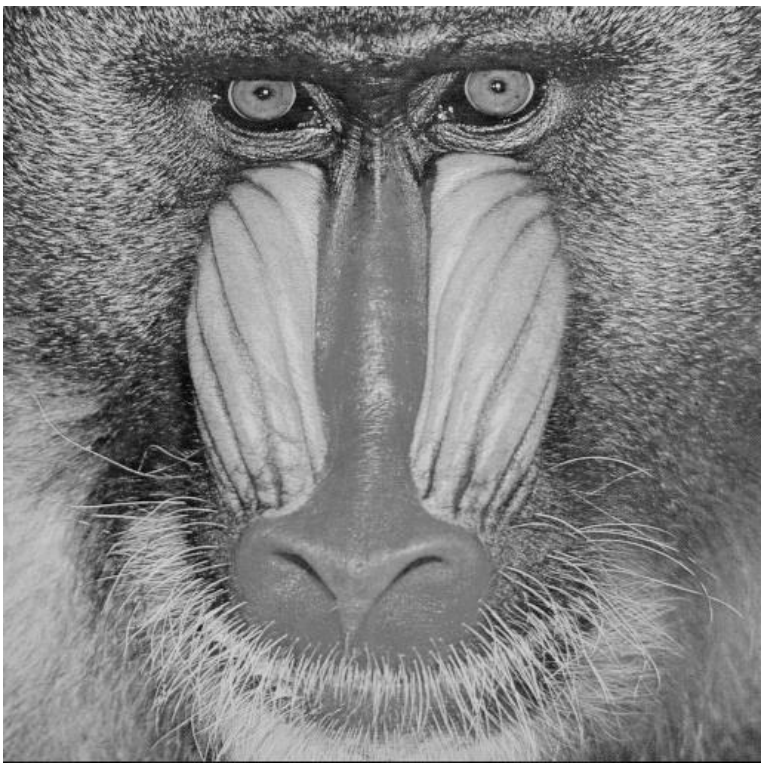


Figure 4.7 : image originale Baboon 512x512 pixels

Figure 4.8 : image originale Lena 512x512 pixels

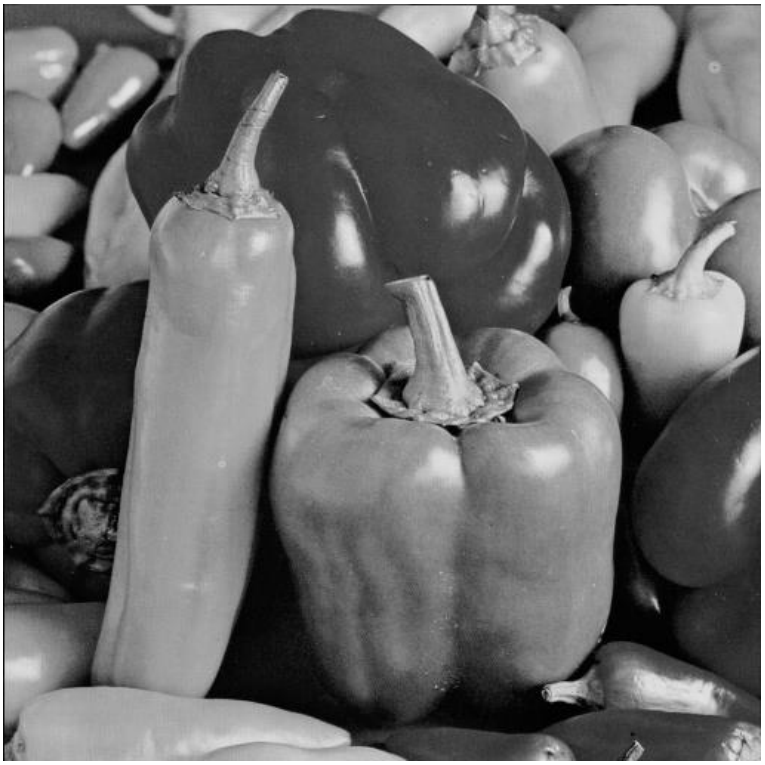


Figure 4.9 : image originale Peppers 512x512 pixels

4-4 Niveaux de décomposition

Un autre critère aussi important dans la compression est le niveau de décomposition de la transformée par ondelettes. Dans ce travail, nous avons choisi quatre résolutions pour pouvoir montrer l'efficacité de notre algorithme.

4-5 Approche proposée

Au début, nous appliquons la transformation en bandelettes sur l'image d'entrée. Les coefficients de bandelettes sont indexés par un scan en zigzag (Morton Scan).

Ensuite, nous calculons le seuil de compression "T_n" afin de tester la signifiance des blocs et des coefficients de bandelettes par rapport à ce seuil; le codage se fait par 'o' si les coefficients ou les blocs désignés sont insignifiants et par '1' s'ils sont signifiants par rapport au seuil calculé.

Enfin, nous terminons par une étape de raffinement des coefficients significatifs. Après la décrémentation du seuil "T_n", le programme se déroule jusqu'à l'obtention du taux de compression désiré (figure 4.10).

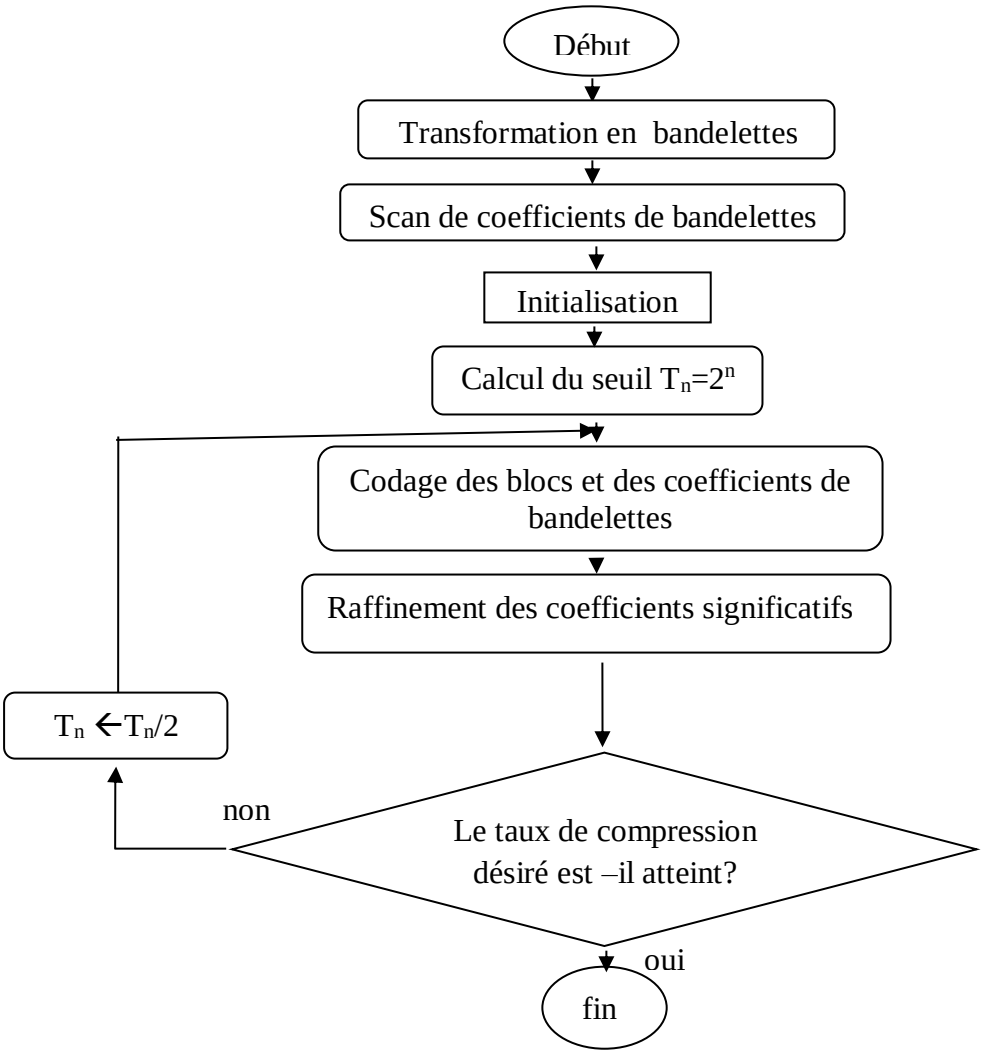


Figure 4.10: Organigramme de l'algorithme de compression considéré

4-6 Etapes détaillées de l'algorithme :

4-6-1-Transformation en bandelettes :

La transformée en bandelettes s'effectue en Cinq étapes principales (figure 4.11):

a- Transformation en ondelettes TO_M de l'image M.

b- Segmentation de chaque sous-bande en carrés dyadiques.

c- Détermination de la meilleure géométrie qui définit la direction de la régularité géométrique dans chaque carré dyadique. La meilleure géométrie est celle qui minimise le lagrangien L comme suit :

$$L=\| TO_M - A_M\|^2+m\, T_g{}^2 \qquad (4.4)$$

Tel que:

T_g : seuil de la géométrie.

A_M : Approximation de TO_M avec une base orthogonale (ondelette de Haar).

m : Nombre de coefficients de l'approximation de TO_M supérieurs au seuil T_g.

d- Une projection orthogonale 1D est effectuée à la géométrie spécifiée de chaque carré pour définir un signal discret 1D S_d.

e- Une transformation en ondelettes discrète de Haar 1D est appliquée au signal $1DS_d$ donnant les coefficients de bandelettes b_k.

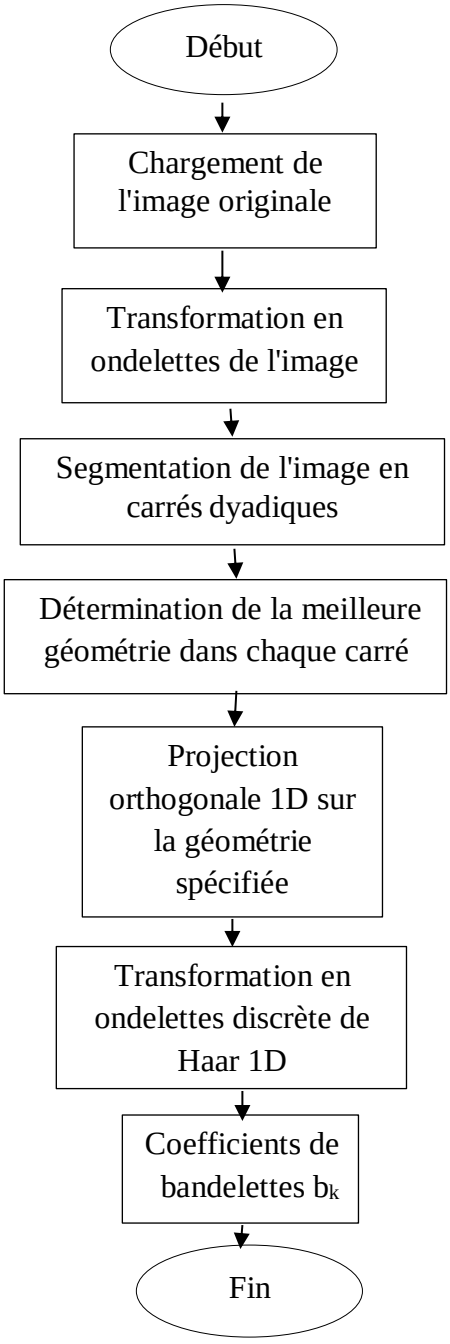


Figure 4.11: Etapes détaillées de la transformation en bandelettes

a- Transformation en ondelettes de l'image:

Nous avons chargé l'image originale parmi des différentes images de tailles :128x128, 256x256 et 512x512. Ensuite Nous avons appliqué sur cette image la transformée en ondelettes afin d'obtenir des sous-bandes corrélées entre elles.

b-Segmentation de chaque sous-bande en carrés dyadiques

Nous segmentons l'image transformée en ondelettes TO_M en carrés dyadiques selon les étapes suivantes:

- ✓ Calcul de la taille "N" de l'image originale transformée en ondelettes.
- ✓ Choix du rang "r" qui détermine la taille du carré (2^r).
- ✓ Segmentation de TO_M en (N/2^r) carrés.
- ✓ Recherche de la direction optimale pour chaque carré par rapport au seuil T_g.

c- Détermination de la direction optimale pour chaque carré

Nous cherchons la valeur de Theta qui représente l'angle optimal pour chaque carré (on met Inf pour aucune géométrie dans le carré). La direction optimale ($Theta_{(optimal)}$) est celle qui minimise le Lagrangien "L".

La détermination de la meilleure géométrie comprend les étapes suivantes:

1) Choix du facteur "s" de super résolution pour la géométrie par défaut (s=2).
2) Nombre de directions testés est : 2 * (2^r) * s
3) Le pas de test est:

$$t = \pi / (2 * (2^r) * s) \quad (4.5)$$

4) Calcul des angles de direction Theta qui prend les valeurs de l'intervalle [t/2 π-t/2] avec un pas de t et en ajoutant la valeur de Theta d'aucune géométrie (Inf).
5) Initialement "L" le vecteur pour stocker les valeurs de lagrangien est vide.
6) Calcul $W_{carré}$(Theta) la transformée warpée de Haar des coefficients de transformée d'ondelettes pour chaque Theta comme suit:

- ✓ Echantillonnage en deux dimensions (X,Y) d'ordre (2^r x 2^r).
- ✓ Projection sur la direction orthogonale sur le repère (-sin et cos) selon la relation suivante :

$$P_{Theta} = -\sin(Theta)* + \cos(Theta)*Y(:) \qquad (4\text{-}6)$$

- ✓ Mettre les valeurs de " P_{Theta} " selon l'ordre croissant.
- ✓ Application de la transformée de Haar sur les coefficients d'ondelettes selon l'ordre croissant des valeurs de " P_{Theta} ".

7) Calcul l'erreur entre la transformée warpée de Haar ($W_{carré}$(Theta)) et la transformée d'ondelettes ($TO_{Carré}$) pour chaque Theta.
8) Calcul de la valeur de "m" qui représente le nombre de coefficients de la transformée ($W_{carré}$(Theta)) supérieurs à T_g pour chaque Theta.
9) Calcul du nouveau lagrangien L pour chaque theta comme suit :

$$L = \sum_{i=1}^{2^r}\sum_{j=1}^{2^r}|W_{Carré}(\text{Theta}) - TO_{Carré}|^2 + m * T_g\text{^}2 \qquad (4\text{-}7)$$

10) Mettre la valeur du nouveau lagrangien pour chaque Theta au vecteur des lagrangiens.
11) Recherche de la valeur minimale du Lagrangien parmi le vecteur des lagrangiens.
12) L'angle optimal $Theta_{(optimal)}$ pour chaque carré est l'angle qui correspond à la valeur minimale du Lagrangien.

d- Projection orthogonale 1D sur la géométrie spécifiée pour chaque carré

La projection orthogonale sur la géométrie spécifiée du carré dyadique est obtenue comme suit:

- ✓ Aucune projection orthogonale sur la géométrie spécifiée si la valeur de Theta est Inf.
- ✓ La taille du carré est toujours (2^r).
- ✓ Echantillonnage en deux dimensions (X,Y) d'ordre (2^r x 2^r).

- ✓ Projection sur la direction orthogonale sur le repère (-sin et cos) selon la relation suivante :

$$P_{Theta(optimal)} = -\sin(Theta_{(optimal)})* + \cos(Theta_{(optimal)})*Y(:) \qquad (4\text{-}8)$$

- ✓ Mettre les valeurs de " $P_{Theta(optimal)}$ " selon l'ordre croissant.
- ✓ Obtenir les coefficients de bandelettes b_k par l'application de la transformée de Haar sur les coefficients d'ondelettes selon l'ordre croissant des valeurs de "$P_{Theta(optimal)}$ " .

2-6-2 Scan des coefficients de bandelettes

D'abord, les coefficients de bandelettes de l'image transformée sont indexés par un scan en zigzag selon Morton Scan (figure 3.7).

4-6-3 Initialisation

1) Nous calculons le seuil $T_n=2^n$ tel que (n) : représente la partie entière du logarithme base 2 de la valeur absolue du maximum des coefficients de l'image transformée en bandelettes.

$$n=fix(log2(max(abs(image\_ transformée\_bandelettes)))) \qquad (4.9)$$

2) La valeur initiale de "i" est 1.
3) La valeur initiale de l'indice "r" est log2(taille_image), si la taille de l'image est 256*256 donc r=8 et si la taille de l'image est 512*512 donc r=9.
4) Nous initialisons la liste des coefficients signifiants $LSP = \emptyset$

4-6-4 Sortie des bits :

Nous appliquons le codage suivant (figure 4.12) à partir de i=1 jusqu'à i=taille_image pour les coefficients de bandelettes C_i qui ne se trouvent pas dans la liste des coefficients signifiants (LSP).

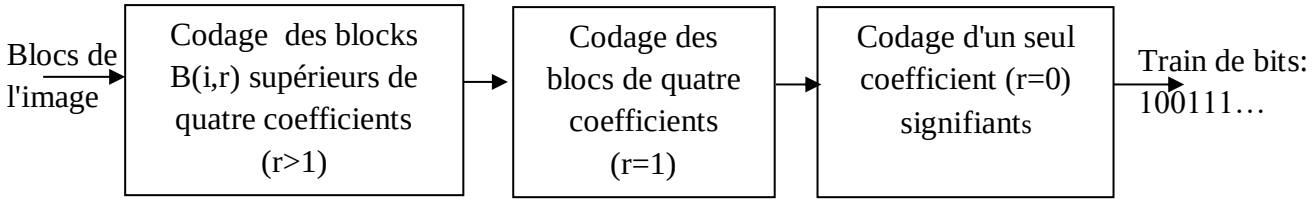


Figure 4.12 : Succession de partition des blocs du plus grand bloc jusqu'au plus petit bloc

a- Codage d'un bloc B(i,r): r>1

si B(i,r) est insignifiant **alors**
- sortie du bit '**0**'
- on passe au bloc suivant **B(i+(2^(2*r)),r)**

Sinon
- sortie du bit '**1**'
- partition **B(i,r)** en 4 blocs: r← r-1
  - **B(i,r)**
  - **B(i+(2^(2*r)), r)**
  - **B(i+2*(2^(2*r)), r)**
  - **B(i+3*(2^(2*r)),r)**

Fin

b- Codage d'un bloc B(i,r) de quatre coefficients (r=1):

Si B(i,1) est insignifiant **alors**
- sortie du bit '**0**'
- on passe au bloc suivant **B(i+4,1)**

Sinon
- sortie du bit '**1**'

Si B(i,1) contient un seul coefficient signifiant parmi le quatre **alors**
- sortie bit '1'

Si la signifiance des quatre coefficients successifs C_i, C_{i+1},C_{i+2},C_{i+3} selon le tableau suivant **alors**
(**S** le coefficient est signifiant et **I** le coefficient est insignifiant)

C_i	C_{i+1}	C_{i+2}	C_{i+3}	Bits de sortie	
S	I	I	I	00	$C_i \in LSP$
I	S	I	I	01	$C_{i+1} \in LSP$
I	I	S	I	10	$C_{i+2} \in LSP$
I	I	I	S	11	$C_{i+3} \in LSP$

Si S est négatif **alors** Sortie bit '1'
Sinon sortie bit '0'
Fin

- i→i+4

Sinon
- sortie bit '0'
- partition **B(i,1)** en 4 coefficient :
  - **C_i**
  - **C_{i+1}**
  - **C_{i+2}**
  - **C_{i+3}**

Fin

Fin

c- Codage d'un seul coefficient (r=0):

Si C_i est insignifiant **alors**
- sortie du bit '**0**'

Sinon
- sortie du bit '**1**'
- $C_i \in LSP$

si C_i est négatif **alors** Sortie bit '1'
Sinon sortie bit '0'
Fin

Fin

- i→i+1

4-7 Résultats :

Dans ces tableaux, nous commençons par comparer notre méthode SPECK modifié avec le codeur SPECK pour différentes valeurs du seuil T_g puis pour différentes valeurs du rang (la taille des carrés). Ensuite, nous faisons varier le PSNR en fonction du taux de compression (TC).

Les tableaux suivants illustrent les résultats obtenus avec différentes images de différentes tailles.

a- Image Penny (taille 128x128)

	PSNR (dB)		
Valeur du seuil T_g	SPECK	SPECK modifié avec ondelettes	SPECK modifié avec bandelettes
1	39.63	39.88	39.*88*
2			39.97
3			40.07
4			40.12
5			40.14
6			40.15
7			40.17
8			40.10
9			40.11
10			40.09
15			39.80
20			39.75

Tableau 4-1: Variation du PSNR en fonction de T_g pour SPECK, SPECK modifié avec ondelettes et SPECK modifié avec bandelettes appliqués sur l'image Penny de taille 128x128 (TC=90%, nombre de bits est 13107 et rang=2)

	PSNR (dB)		
rang	SPECK	SPECK modifié avec ondelettes	SPECK modifié avec bandelettes
1	39.63	39.88	40.08
2			40.17
3			40.01
4			39.92
5			39.88
6			39.88
7			39.88

Tableau 4-2: Variation du PSNR en fonction du rang pour SPECK, SPECK modifié avec ondelettes et SPECK modifié avec bandelettes appliqués sur l'image Penny de taille 128x128 (TC=90%, nombre de bits 13107 et T=7)

			PSNR (dB)		
TC (%)	Débit (bpp)	Nombre de bits	SPECK	SPECK modifié avec ondelettes	SPECK modifié avec bandelettes
90	0.8	13107	39.63	39.88	40.17
91	0.72	11796	38.79	38.97	39.25
92	0.64	10485	37.98	38.12	38.44
93	0.56	9175	37.08	37.25	37.56
94	0.48	7864	35.73	36.03	36.14
95	0.40	6553	34.77	34.86	35.04

Tableau 4-3: Variation du PSNR en fonction du TC pour SPECK, SPECK modifié avec ondelettes et SPECK modifié avec bandelettes appliqués sur l'image Penny de taille 128x128 (T_g=7 et rang=2)

Figure 4.13: Image Penny reconstruite 128×128 par SPECK modifié avec ondelettes (TC=95% et PSNR= 34.86 dB)

b- Image Woman (taille 256 x 256)

	PSNR (dB)		
Valeur du seuil T_g	SPECK	SPECK modifié	SPECK modifié avec bandelettes
1	34.97	35.12	35.06
2			35.15
3			35.23
4			35.29
5			35.34
6			35.39
7			35.41
8			35.43
9			35.45
10			35.45
11			35.46
12			35.45
13			35.43
14			35.42
15			35.38
20			35.21

Tableau 4-4: Variation du PSNR en fonction de T_g pour SPECK, SPECK modifié avec ondelettes et SPECK modifié avec bandelettes appliqués sur l'image Woman de taille 256x256 (TC==90%, nombre de bits est 52428 et rang=2)

	PSNR (dB)		
rang (r)	SPECK	SPECK modifié	SPECK modifié avec bandelettes
1	34.97	35.12	35.42
2			35.46
3			35.30
4			35.18
5			35.15
6			35.13
7			35.13

Tableau 4-5: Variation du PSNR en fonction du rang pour SPECK, SPECK modifié avec ondelettes et SPECK modifié avec bandelettes appliqués sur l'image Woman de taille 256x256 (TC=90%, nombre de bits 52428 et T_g=11)

			PSNR (dB)		
TC (%)	Débit (bpp)	Nombre de bits	SPECK	SPECK modifié avec ondelettes	SPECK modifié avec bandelettes
90	0.8	52428	34.97	35.12	35.46
91	0.72	47185	34.60	34.72	35.09
92	0.64	41943	34.20	34.31	34.66
93	0.56	36700	33.52	33.73	33.81
94	0.48	31457	32.81	32.93	33.15
95	0.40	26214	32.00	32.12	32.33

Tableau 4-6: Variation du PSNR en fonction du TC pour SPECK, SPECK modifié avec ondelettes et SPECK modifié avec bandelettes appliqués sur l'image Woman de taille 256x256 (T_g=11 et rang=2)

Figure 4.14: Image Woman reconstruite 256×256 par SPECK modifié avec bandelettes (TC=91% ,T_g=11 , rang=2 et PSNR=35.09 dB)

c- Image Hoed (taille 256 x 256)

	PSNR (dB)		
Valeur du seuil T_g	SPECK	SPECK modifié	SPECK modifié avec bandelettes
1	32.61	32.88	32.81
2			32.93
3			32.98
4			33.08
5			33.15
6			33.23
7			33.27
8			33.33
9			33.38
10			33.40
11			33.44
12			33.48
13			33.51
14			33.51
16			33.48
20			33.48
25			33.31

Tableau 4-7: Variation du PSNR en fonction de T_g pour SPECK, SPECK modifié avec ondelettes et SPECK modifié avec bandelettes appliqués sur l'image Hoed de taille 256x256 (TC==90%, nombre de bits est 52428 et rang=2)

	PSNR (dB)		
rang	SPECK	SPECK modifié	SPECK modifié avec bandelettes
1	28.78	28.91	29.25
2			29.27
3			29.00
4			28.91

Tableau 4-8: Variation du PSNR en fonction du rang pour SPECK, SPECK modifié avec ondelettes et SPECK modifié avec bandelettes appliqués sur l'image Hoed de taille 256x256 (TC=95%, nombre de bits 26214 et T_g=14)

			PSNR (dB)		
TC (%)	Débit (bpp)	Nombre de bits	SPECK	SPECK modifié avec ondelettes	SPECK modifié avec bandelettes
90	0.8	52428	32.61	32.88	33.51
91	0.72	47185	31.87	32.00	32.60
92	0.64	41943	30.99	31.19	31.74
93	0.56	36700	30.28	30.40	31.01
94	0.48	31457	29.66	29.80	30.36
95	0.40	26214	28.78	28.91	29.27

Tableau 4-9: Variation du PSNR en fonction du TC pour SPECK, SPECK modifié avec ondelettes et SPECK modifié avec bandelettes appliqués sur l'image Hoed de taille 256x256 (T_g=14 et rang=2)

Figure 4.15: Image Hoed reconstruite 256×256 par SPECK modifié avec bandelettes (TC=91% ,T_g=14 , rang=2 et PSNR=32.60 dB)

d- Image Cameraman (taille 256 x 256)

	PSNR (dB)		
Valeur du seuil T_g	SPECK	SPECK modifié	SPECK modifié avec bandelettes
1	30.33	30.45	30.35
2			30.49
3			30.61
4			30.70
5			30.73
6			30.76
7			30.80
8			30.82
9			30.87
10			30.90
11			30.93
12			30.97
13			30.99
14			31.01
15			31.02
16			31.04
17			31.05
18			31.08
19			31.11
20			31.13
21			31.13
22			31.14
23			31.13
25			31.13
30			31.06

Tableau 4-10: Variation du PSNR en fonction de T_g pour SPECK, SPECK modifié avec ondelettes et SPECK modifié avec bandelettes appliqués sur l'image Cameraman de taille 256x256 (TC==94%, nombre de bits est 31457 et rang=2)

	PSNR (dB)		
rang	SPECK	SPECK modifié	SPECK modifié avec bandelettes
1	33.71	33.96	34.31
2			34.41
3			34.27
4			34.15
5			33.96

Tableau 4-11: Variation du PSNR en fonction du rang pour SPECK, SPECK modifié avec ondelettes et SPECK modifié avec bandelettes appliqués sur l'image Cameraman de taille 256x256 (TC=90%, nombre de bits 52428 et T_g=20)

TC (%)	Débit (bpp)	Nombre de bits	PSNR (dB) SPECK	SPECK modifié avec ondelettes	SPECK modifié avec bandelettes
90	0.8	52428	33.71	33.96	34.41
91	0.72	47185	32.73	32.99	33.46
92	0.64	41943	31.86	32.04	32.60
93	0.56	36700	31.11	31.23	31.82
94	0.48	31457	30.33	30.45	31.13
95	0.40	26214	29.24	29.39	29.83

Tableau 4-12: Variation du PSNR en fonction du TC pour SPECK, SPECK modifié avec ondelettes et SPECK modifié avec bandelettes appliqués sur l'image Cameraman de taille 256x256 (T_g=20 et rang=2)

Figure 4.16: Image Cameraman reconstruite 256×256 par SPECK modifié avec bandelettes (TC=94% , T_g=20, rang=2 et PSNR= 31.13 dB)

e- Image Goldhill (taille 256 x 256)

	PSNR (dB)		
Valeur du seuil T_g	SPECK	SPECK modifié	SPECK modifié avec bandelettes
1	29.77	29.85	29.83
2			29.91
3			29.98
4			30.02
5			30.05
6			30.11
7			30.15
8			30.20
9			30.22
10			30.25
11			30.28
12			30.29
13			30.32
14			30.33
15			30.35
16			30.36
17			30.36
18			30.37
19			30.37
20			30.36

Tableau 4-13: Variation du PSNR en fonction de T_g pour SPECK, SPECK modifié avec ondelettes et SPECK modifié avec bandelettes appliqués sur l'image Goldhill de taille 256x256 (TC==92%, nombre de bits est 41943 et rang=2)

	PSNR (dB)		
rang	SPECK	SPECK modifié	SPECK modifié avec bandelettes
1	29.77	29.85	30.23
2			30.37
3			30.21
4			30.01
5			29.94

Tableau 4-14: Variation du PSNR en fonction du rang pour SPECK, SPECK modifié avec ondelettes et SPECK modifié avec bandelettes appliqués sur l'image Goldhill de taille 256x256 (TC=92%, nombre de bits 41943 et T_g=18)

			PSNR (dB)		
TC (%)	Débit (bpp)	Nombre de bits	SPECK	SPECK modifié avec ondelettes	SPECK modifié avec bandelettes
90	0.8	52428	30.82	30.98	31.40
91	0.72	47185	30.25	30.39	30.85
92	0.64	41943	29.77	29.85	30.21
93	0.56	36700	29.36	29.45	29.92
94	0.48	31457	28.83	28.91	29.41
95	0.40	26214	28.12	28.21	28.59

Tableau 4-15: Variation du PSNR en fonction du TC pour SPECK, SPECK modifié avec ondelettes et SPECK modifié avec bandelettes appliqués sur l'image Goldhill de taille 256x256 (T_g=18 et rang=2)

Figure 4.17: Image Goldhill reconstruite 256×256 par SPECK modifié avec bandelettes (TC=91% , T_g=18 , rang=2 et PSNR= 30.85dB)

f- Image Baboon (taille 512 x 512)

	PSNR (dB)		
Valeur du seuil T_g	SPECK	SPECK modifié	SPECK modifié avec bandelettes
1	27.36	27.49	27.49
2			27.56
3			27.62
4			27.67
5			27.71
6			27.75
7			27.79
8			27.82
9			27.84
10			27.87
11			27.95
12			27.94
13			27.97
14			27.99
15			28.01
16			28.03
17			28.05
18			28.06
19			28.08
20			28.09
21			28.10
22			28.10
23			28.11
24			28.10
25			28.10
26			28.09
30			28.04

Tableau 4-16: Variation du PSNR en fonction de T_g pour SPECK, SPECK modifié avec ondelettes et SPECK modifié avec bandelettes appliqués sur l'image Baboon de taille 512x512 (TC=90%, nombre de bits est 209715 et rang=2)

	PSNR (dB)		
rang	SPECK	SPECK modifié	SPECK modifié avec bandelettes
1	27.36	27.49	28.02
2			28.11
3			27.88
4			27.61
5			27.41

Tableau 4-17: Variation du PSNR en fonction du rang pour SPECK, SPECK modifié avec ondelettes et SPECK modifié avec bandelettes appliqués sur l'image Baboon de taille 512x512 (TC=90%, nombre de bits 209715 et T_g=23)

RC (%)	Débit (bpp)	Nombre de bits	PSNR (dB)		
			SPECK	SPECK modifié avec ondelettes	SPECK modifié avec bandelettes
90	0.8	209715	27.36	27.49	28.11
91	0.72	188743	26.77	26.95	27.45
92	0.64	167772	26.12	26.30	26.74
93	0.56	146800	25.22	25.41	25.82
94	0.48	125829	24.60	24.68	25.21
95	0.40	104857	24.07	24.15	24.60

Tableau 4-18: Variation du PSNR en fonction du TC pour SPECK, SPECK modifié avec ondelettes et SPECK modifié avec bandelettes appliqués sur l'image Baboon de taille 512x512 (T_g=22 et rang=2)

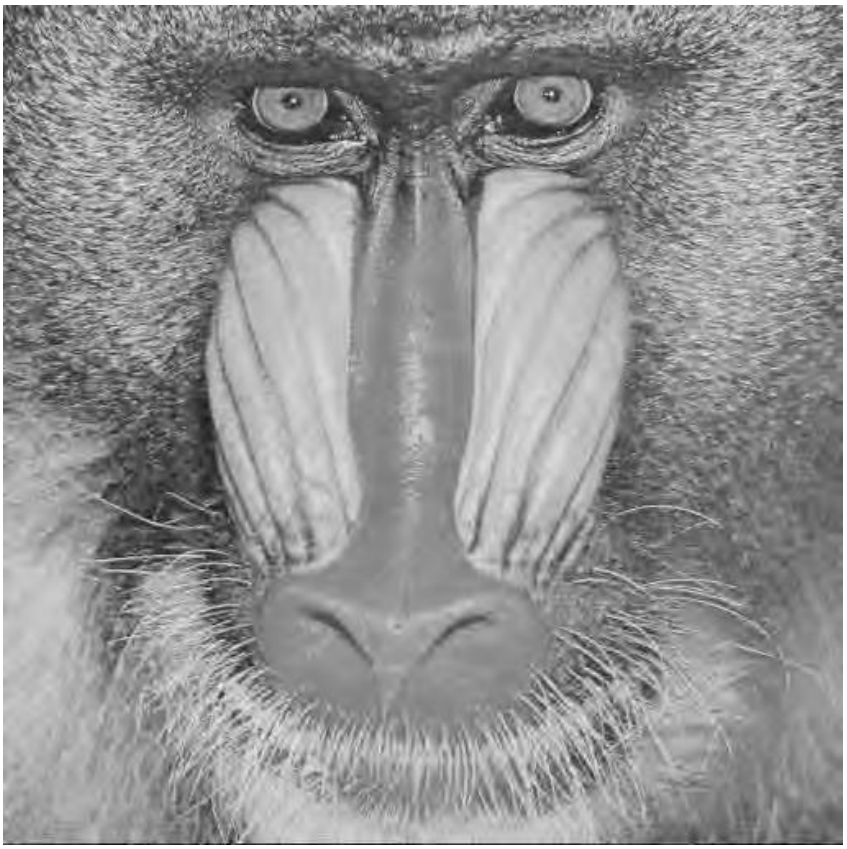


Figure 4.18: Image Baboon reconstruite 512×512 par SPECK modifié avec bandelettes (TC=95% , T_g=22, rang=2 et PSNR= 24.60 dB)

g- Image Barbara (taille 512 x 512)

	PSNR (dB)		
Valeur du seuil T_g	SPECK	SPECK modifié	SPECK modifié avec bandelettes
1	30.47	30.49	30.76
2			30.87
3			30.96
4			31.03
5			31.10
6			31.14
7			31.14
8			31.22
9			31.25
10			31.30
11			31.32
12			31.35
13			31.37
14			31.39
15			31.41
16			31.44
17			31.45
18			31.45
19			31.46
20			31.46
21			31.46
30			31.41

Tableau 4-19: Variation du PSNR en fonction de T_g pour SPECK, SPECK modifié avec ondelettes et SPECK modifié avec bandelettes appliqués sur l'image Barbara de taille 512x512 (TC=95%, nombre de bits est 104857 et rang=2)

	PSNR (dB)		
rang	SPECK	SPECK modifié	SPECK modifié avec bandelettes
1	30.47	30.49	31.39
2			31.46
3			31.39
4			31.20
5			30.99

Tableau 4-20: Variation du PSNR en fonction du rang pour SPECK, SPECK modifié avec ondelettes et SPECK modifié avec bandelettes appliqués sur l'image Barbara de taille 512x512 (TC=95%, nombre de bits 104857 et T_g=19)

RC (%)	Débit (bpp)	Nombre de bits	PSNR (dB)		
			SPECK	SPECK modifié avec ondelettes	SPECK modifié avec bandelettes
90	0.8	209715	35.07	35.10	35.75
91	0.72	188743	34.45	34.50	35.21
92	0.64	167772	33.43	33.52	34.21
93	0.56	146800	32.20	32.24	33.13
94	0.48	125829	31.20	31.24	32.20
95	0.40	104857	30.47	30.49	31.46

Tableau 4-21: Variation du PSNR en fonction du TC pour SPECK, SPECK modifié avec ondelettes et SPECK modifié avec bandelettes appliqués sur l'image Barbara de taille 512x512 (Tg=19 et rang=2)

Figure 4.19: Image Barbara reconstruite 512×512 par SPECK modifié avec bandelettes (TC=94% , T_g=19, rang=2 et PSNR= 32.20 dB)

h-Image Lena (taille 512 x 512)

	PSNR (dB)		
Valeur du seuil T_g	SPECK	SPECK modifié	SPECK modifié avec bandelettes
1	38.86	38.96	39.10
2			39.16
3			39.21
4			39.25
5			39.27
6			39.28
7			39.29
8			39.26
9			39.32
10			39.24
11			39.16
12			39.09
13			39.09

Tableau 4-22: Variation du PSNR en fonction de T_g pour SPECK, SPECK modifié avec ondelettes et SPECK modifié avec bandelettes appliqués sur l'image Lena de taille 512x512 (TC=90%, nombre de bits est 209715 et rang=2)

	PSNR (dB)		
rang (r)	SPECK	SPECK modifié	SPECK modifié avec bandelettes
1	38.86	38.96	39.26
2			39.29
3			39.17
4			39.07
5			39.03

Tableau 4-23: Variation du PSNR en fonction du rang pour SPECK, SPECK modifié avec ondelettes et SPECK modifié avec bandelettes appliqués sur l'image Lena de taille 512x512 (TC=90%, nombre de bits est 209715 et T_g=7)

RC (%)	Débit (bpp)	Nombre de bits	PSNR (dB)		
			SPECK	SPECK modifié avec ondelettes	SPECK modifié avec bandelettes
90	0.8	209715	38.86	38.96	39.29
91	0.72	188743	38.21	38.36	38.66
92	0.64	167772	37.61	37.74	38.05
93	0.56	146800	36.92	37.01	37.40
94	0.48	125829	36.35	36.44	36.87
95	0.40	104857	35.70	35.79	36.21

Tableau 4-24: Variation du PSNR en fonction du TC pour SPECK, SPECK modifié avec ondelettes et SPECK modifié avec bandelettes appliqués sur l'image Lena de taille 512x512 (Tg=9 et rang=2)

Figure 4.20: Image Lena reconstruite 512×512 par SPECK modifié avec bandelettes (TC=95% , T_g=9, rang=2 et PSNR= 36.21 dB)

i-Image Peppers (taille 512 x 512)

	PSNR (dB)		
Valeur du seuil T_g	SPECK	SPECK modifié	SPECK modifié avec bandelettes
1	36.80	36.94	36.94
2			37.01
3			37.07
4			37.10
5			37.15
6			37.17
7			37.19
8			37.19
9			37.25
10			37.26
11			37.22
12			37.16
13			37.11

Tableau 4-25: Variation du PSNR en fonction de T_g pour SPECK, SPECK modifié avec ondelettes et SPECK modifié avec bandelettes appliqués sur l'image Peppers de taille 512x512 (TC=90%, nombre de bits est 209715 et rang=2)

	PSNR (dB)		
rang (r)	SPECK	SPECK modifié	SPECK modifié avec bandelettes
1	36.80	36.94	37.19
2			37.26
3			37.14
4			37.04
5			37.00

Tableau 4-26: Variation du PSNR en fonction du rang pour SPECK, SPECK modifié avec ondelettes et SPECK modifié avec bandelettes appliqués sur l'image Peppers de taille 512x512 (TC=90%, nombre de bits est 209715 et T_g=10)

			PSNR (dB)		
RC (%)	Débit (bpp)	Nombre de bits	SPECK	SPECK modifié avec ondelettes	SPECK modifié avec bandelettes
90	0.8	209715	36.80	36.94	37.26
91	0.72	188743	36.38	36.51	36.80
92	0.64	167772	36.01	36.10	46.39
93	0.56	146800	35.64	35.71	36.01
94	0.48	125829	35.21	35.28	35.57
95	0.40	104857	34.46	34.60	34.76

Tableau 4-27: Variation du PSNR en fonction du TC pour SPECK, SPECK modifié avec ondelettes et SPECK modifié avec bandelettes appliqués sur l'image Peppers de taille 512x512 (Tg=10 et rang=2)

Figure 4.21: Image Peppers reconstruite 512×512 par SPECK modifié avec bandelettes (TC=95% , T_g=9, rang=2 et PSNR= 34.76 dB)

4-8 Discussions

4-8-1 Comparaison entre SPECK et SPECK modifié

Le taux de compression augmente dans le cas du SPECK modifié (figures 4.20 , 4.21 , 4.22, 4.23 et 4.24) vu que l'exploitation des insignifiances de coefficients dans les blocs de quatre coefficients (C_i , C_{i+1}, C_{i+2} , C_{i+3}), tel que si on trouve trois coefficients insignifiants parmi les quatre, on les code par deux bits (représentant la position du coefficient signifiant dans le bloc) au lieu de quatre bits dans le codeur SPECK original (tableau 4.28) .

C_i	C_{i+1}
C_{i+2}	C_{i+3}

(a)

C_i	C_{i+1}	C_{i+2}	C_{i+3}	Bits de sortie	
				Nouveau SPECK	SPECK original
S	I	I	I	00	1000
I	S	I	I	01	0100
I	I	S	I	10	0010
I	I	I	S	11	0001

(b)

Tableau 4.28: (a) : Bloc de quatre coefficients (C_i, C_{i+1}, C_{i+2} , C_{i+3})
(b) : Bits de sortie de test de signifiance selon la position du coefficient signifiant dans le bloc de quatre coefficients dont trois insignifiants.
I: coefficient insignifiant **S**: coefficient signifiant

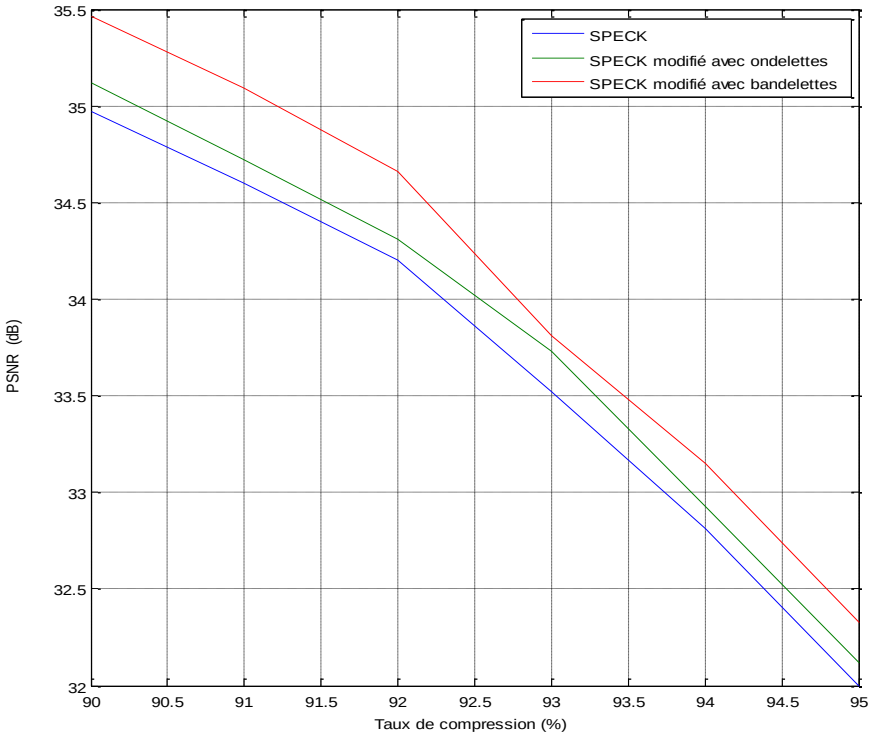


Figure 4.22: Comparaison entre SPECK, SPECK modifié avec ondelettes et SPECK modifié avec bandelettes appliqués sur l'image Woman de taille 256x256 (T_g=10 et rang=2)

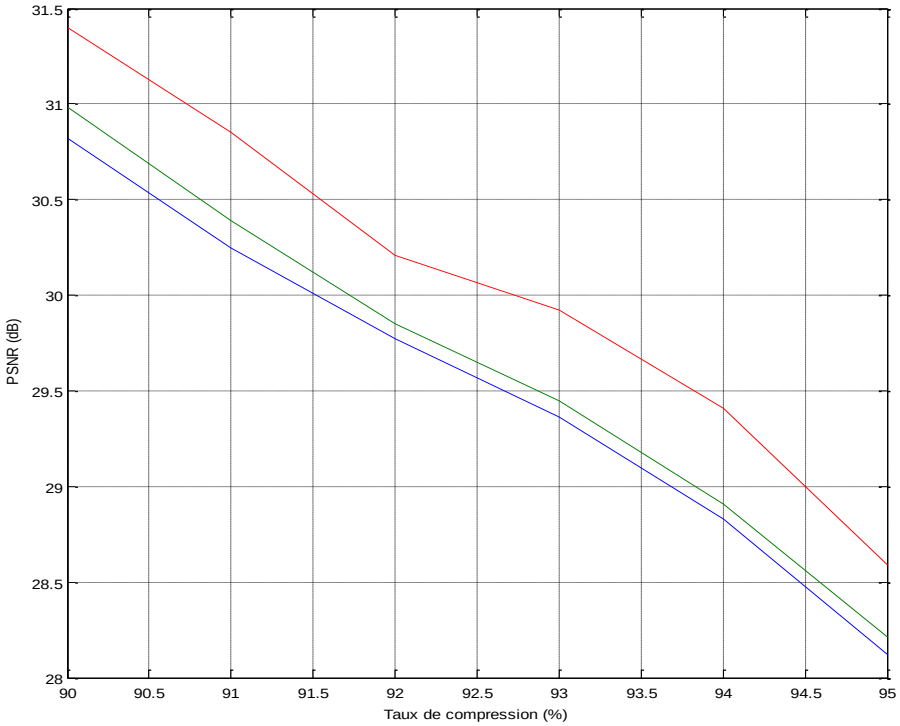


Figure 4.23: Comparaison entre SPECK, SPECK modifié avec ondelettes et SPECK modifié avec bandelettes appliqués sur l'image Goldhill de taille 256x256 (Tg=20 et rang=2)

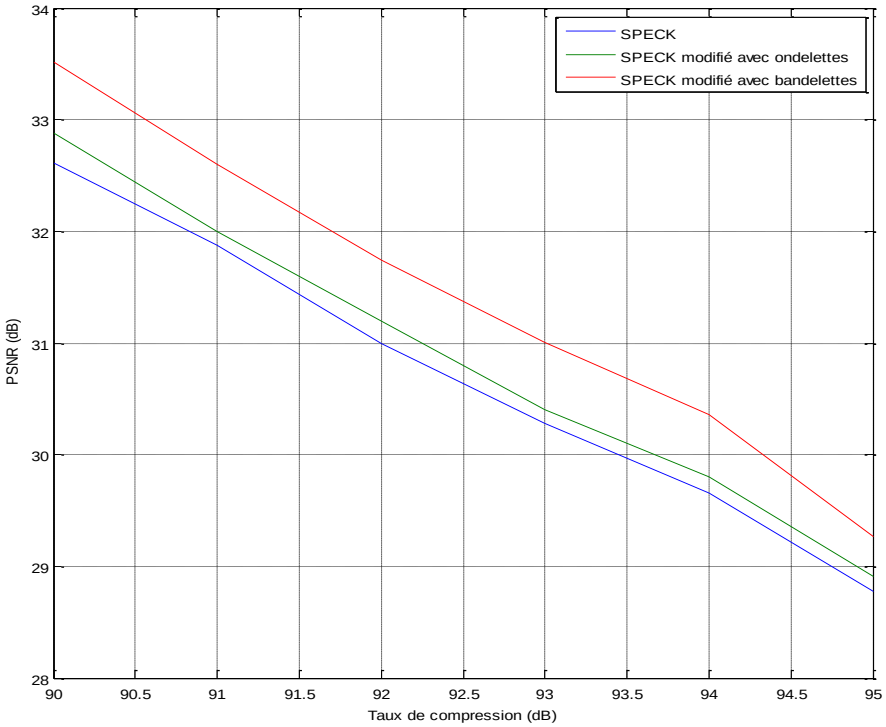


Figure 4.24: Comparaison entre SPECK, SPECK modifié avec ondelettes et SPECK modifié avec bandelettes appliqués sur l'image Hoed de taille 256x256 (T_g=14 et rang=2)

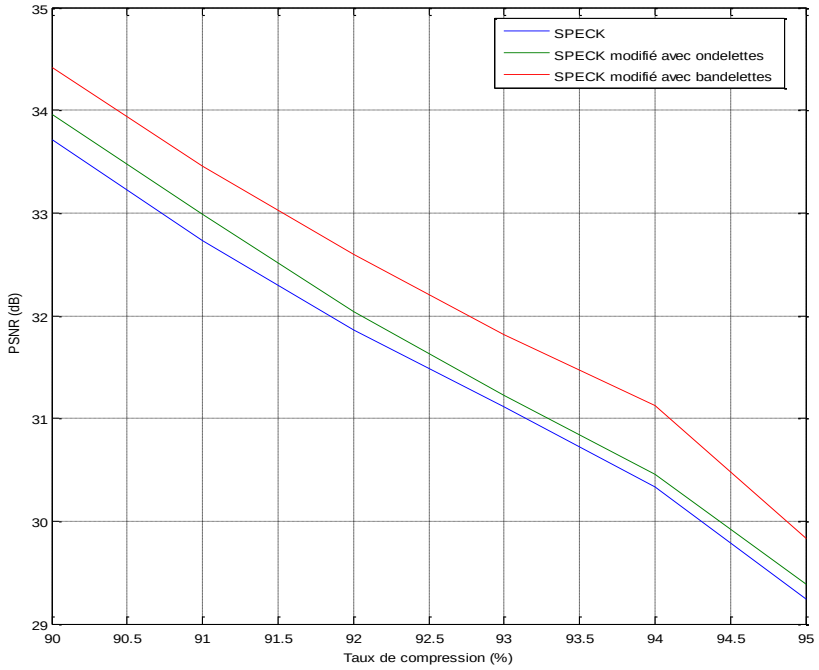


Figure 4.25: Comparaison entre SPECK, SPECK modifié avec ondelettes et SPECK modifié avec bandelettes appliqués sur l'image Cameraman de taille 256x256 (T_g=20 et rang=2)

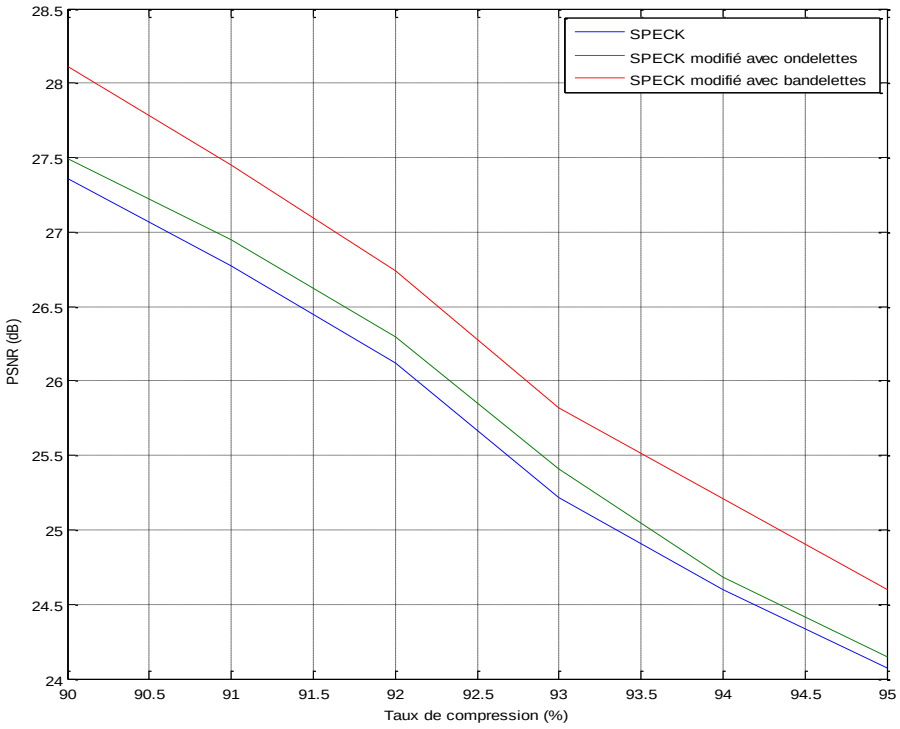


Figure 4.26: Comparaison entre SPECK, SPECK modifié avec ondelettes et SPECK modifié avec bandelettes appliqués sur l'image Baboon de taille 512x512 (T_g=22 et rang=2)

4-8-2 Performances de la transformée en bandelettes en compression

Le nouveau codeur SPECK modifié avec bandelettes donne des meilleurs résultats par rapport à SPECK original et SPECK modifié avec ondelettes car il exploite plus les corrélations entre coefficients de bandelettes (figures 4.20 , 4.21 , 4.22, 4.23 et 4.24).

Les images obtenues avec les bandelettes restent régulière le long de la géométrie utilisée tandis que celle obtenues avec les ondelettes présentent des effets oscillants.

Les résultats de nos tests montrent un gain perceptuel important des bandelettes par rapport aux ondelettes, car la géométrie des images est mieux respectée et les phénomènes d'oscillations près des contours sont diminués.

4-8-3 Etude de l'influence du seuil de géométrie (cas du SPECK modifié avec bandelettes)

L'augmentation de la valeur du seuil T_g implique une amélioration du PSNR (figures 4.25, 4.26, 4.27, 4.28 et 4.29) pour le SPECK modifié avec bandelettes jusqu'à un certain seuil limite où nous verrons à partir de ce dernier une diminution du PSNR. Cela nous justifions par la formule (4.7) : lorsque T_g est grand la valeur de Lagrange devienne plus grand ce qui ne donne pas une meilleure géométrie.

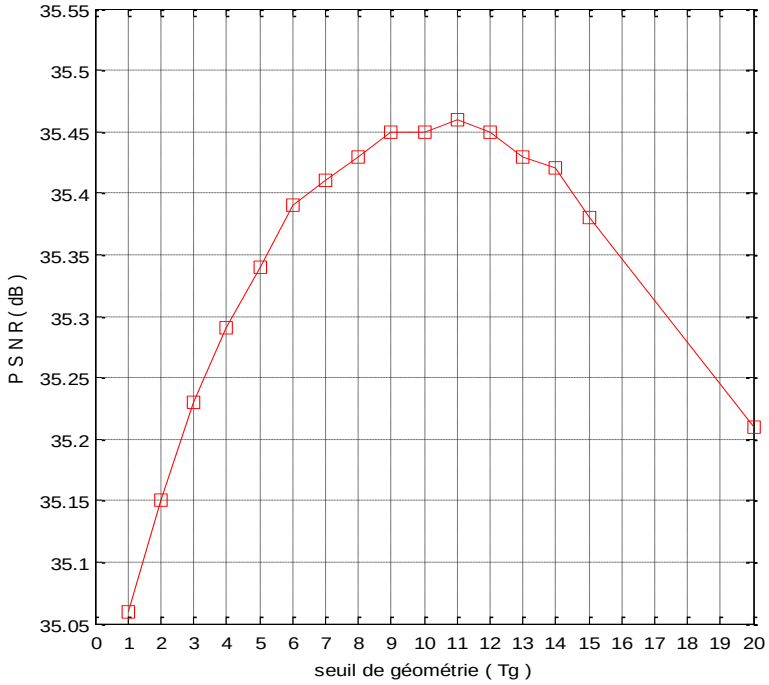


Figure 4.27: Influence du seuil de géométrie pour SPECK modifié avec bandelettes appliqués sur l'image Woman de taille 256x256 (TC==90%, nombre de bits est 52428 et rang=2)

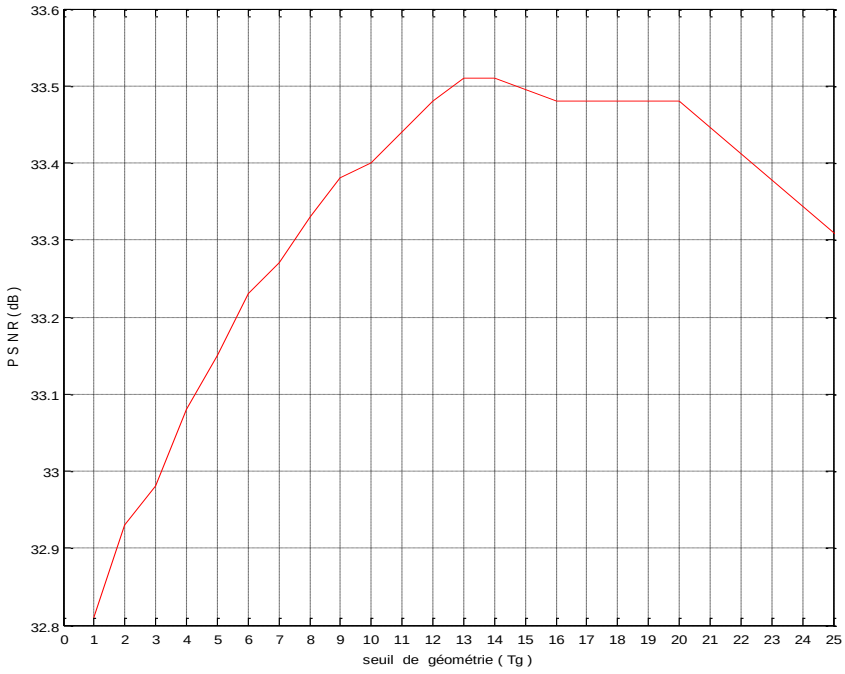


Figure 4-28: Influence du seuil de géométrie pour SPECK modifié avec bandelettes appliqués sur l'image Hoed de taille 256x256 (TC==90%, nombre de bits est 52428 et rang=2)

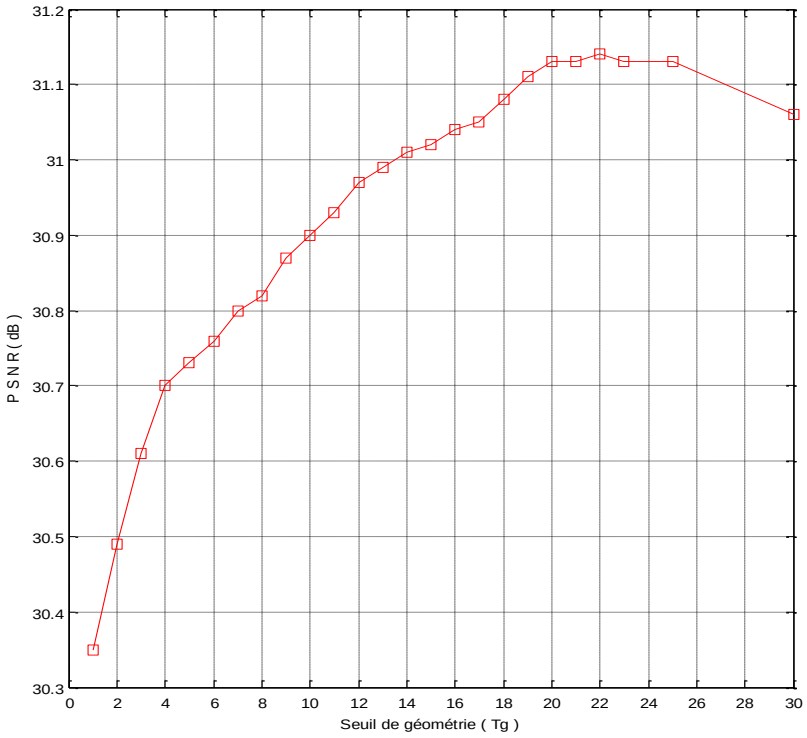


Figure 4-29: Influence du seuil de géométrie pour SPECK modifié avec bandelettes appliqués sur l'image Cameraman de taille 256x256 (TC==94%, nombre de bits est 31457 et rang=2)

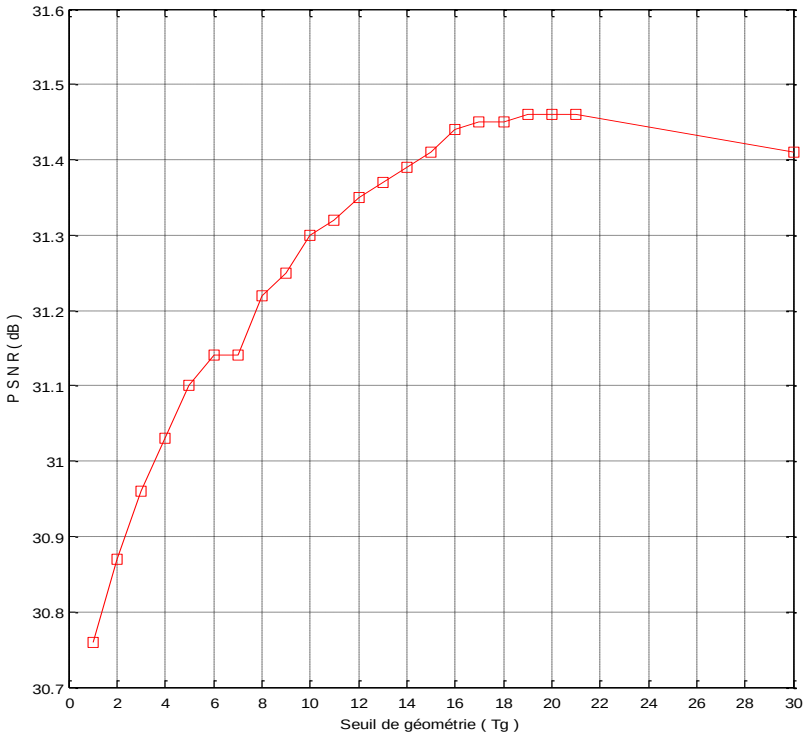


Figure 4-30: Influence du seuil de géométrie pour SPECK modifié avec bandelettes appliqués sur l'image Barbara de taille 512x512 (TC=95%, nombre de bits est 104857 et rang=2)

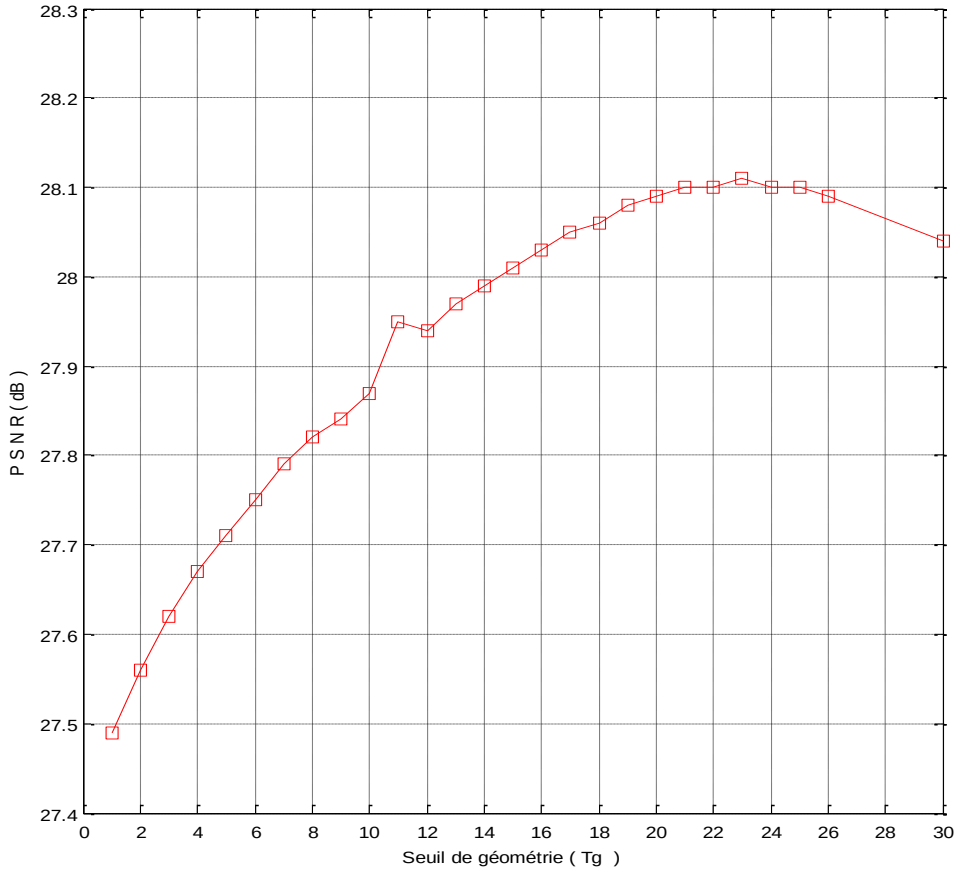


Figure 4-31: Influence du seuil de géométrie pour SPECK modifié avec bandelettes appliqués sur l'image Baboon de taille 512x512 (TC=90%, nombre de bits est 209715 et rang=2)

4-8-4 Comparaison entre le temps de calcul pour SPECK modifié avec ondelettes et SPECK modifié avec bandelettes

- ✓ Le temps de compression est plus supérieur que le temps de décompression car dans le premier nous manipulons un nombre de bits plus grand que dans la décompression, par exemple le nombre de bits codés dans la compression sont 512x512x8 bits =2097152 bits par contre dans la décompression si TC= 90% (débit=0.8 bpp) le nombre de bits décodés sont 209715 bits. On peut conclure que le décodage est rapide que le codage.
- ✓ On remarque que le temps de compression par SPECK modifié avec ondelettes est un peu plus petit que le temps du codeur SPECK original car dans le premier on test les blocs signifiants qui contiennent trois coefficients insignifiants au même temps, mais dans le SPECK le test se fait coefficient par coefficient dans ces blocs particuliers.

- ✓ . Le temps de compression par SPECK modifié avec bandelettes est plus grand que celui par SPECK modifié avec ondelettes car dans la transformée en bandelettes, on a plusieurs étapes supplémentaires. dans l'exploitation de la géométrie de l'image (segmentation en carrés dyadiques - partition des directions - calcul de la meilleure géométrie – Projection orthogonale 1D –une transformation en ondelettes de Haar)

.

TC (%)	Temps de calcul (secondes)					
	SPECK		SPECK modifié avec ondelettes		SPECK modifié avec bandelettes	
	Compression	Décompression	Compression	Décompression	Compression	Décompression
90	12.95	2.21	12.10	1.77	42.45	32.68
91	10.80	2.01	10.49	1.73	41.06	32.49
92	8.81	1.95	8.49	1.60	40.91	32.05
93	8.06	1.91	8.46	1.60	37.74	31.85
94	6.02	1.85	5.72	1.33	34.53	31.49
95	5.53	1.72	4.18	1.28	34.09	30.71

Tableau 4.29 : Temps de calcul pour SPECK, SPECK modifié avec ondelettes et SPECK modifié avec bandelettes sur l'image Barbara 512×512 ($T_{g=}$19 et rang=2)

TC (%)	Temps de calcul (secondes)					
	SPECK		SPECK modifié avec ondelettes		SPECK modifié avec bandelettes	
	Compression	Décompression	Compression	Décompression	Compression	Décompression
90	16.18	2.23	14.28	2.03	51.74	42.52
91	15.09	1.98	13.15	1.90	49.90	38.59
92	11.82	1.92	10.96	1.75	46.39	38.39
93	9.28	1.65	8.91	1.51	44.88	37.71
94	7.51	1.45	7.20	1.35	41.39	35.57
95	5.63	1.28	5.40	1.19	40.16	35.32

Tableau 4.30 : Temps de calcul pour SPECK, SPECK modifié avec ondelettes et SPECK modifié avec bandelettes sur l'image Peppers 512×512 (Tg=10 et rang=2)

4-8-5 Choix de la taille des blocs

Dans cette section, on s'intéresse à la taille de blocs permettant d'obtenir les meilleurs performances pour la transformée en bandelettes. X. Delaunay a étudié dans [28] les dépendances entre coefficients d'ondelettes voisins, il a montré qu'au-delà d'une distance supérieure à 4 pixels entre deux coefficients d'ondelettes, les corrélations sont presque nulles. On s'attend donc à ce que l'exploitation de corrélations sur des blocs de taille supérieure à 4*4 coefficients (rang=2) n'apporte que très peu de gain.

Plusieurs tailles de blocs ont été considérées : sur la figure 4.32, nous avons représenté des blocs de taille 8*8 coefficients (rang=3), avec cette configuration, nous avons constaté une perte de 0.23 dB. Avec des blocs de taille de 16*16 coefficients (rang=4), la perte correspondante à cette configuration est de l'ordre de 0,5 dB. Les résultats de nos tests nous montrent que les meilleures performances sont atteintes avec des blocs de taille de 4*4 coefficients (rang=2).

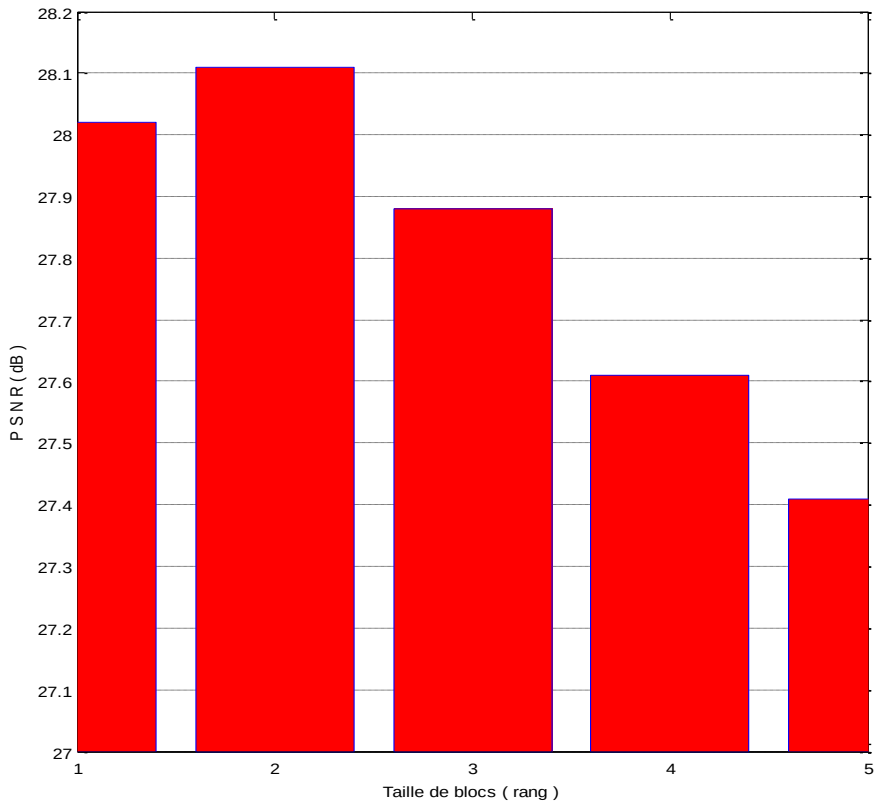


Figure 4-32: Influence de la taille des blocs sur le PSNR pour SPECK modifié avec bandelettes appliqués sur l'image Baboon de taille 512x512 (TC=90%, nombre de bits 209715_et T_g=23)

4-9-Conclusion :

Le SPECK modifié emploie deux bits au lieu de quatre employés dans l'algorithme SPECK original pour coder les blocs signifiants de quatre coefficients dont trois insignifiants. A cet effet, nous avons obtenu une quantité d'informations inférieure par rapport à l'algorithme SPECK original.

L'ajout de la transformée en bandelettes améliore aussi le PSNR parce qu'elle donne des décorrélations supplémentaires dans l'exploitation de la géométrie de l'image.

En se basant sur l'utilisation des pointeurs des coefficients de bandelettes scannés en Morton scan, l'algorithme SPECK modifié devient plus simple du point de vu de la programmation.

Conclusion générale

Nous nous sommes intéressés dans ce travail, par un algorithme de compression d'images fixes basé sur une nouvelle famille de bases, les bandelettes, permettant de capturer les singularités le long de contours. L'algorithme proposé utilise un codage imbriqué qui offre la propriété de la transmission progressive de l'image codée.

Nous proposons dans ce travail une nouvelle approche basée sur une modification de l'algorithme SPECK "Set Partionning Embedded Block" appliquée à une image transformée par bandelettes. L'approche proposée est caractérisée par sa simplicité et donne des meilleurs résultats par rapport au SPECK original car il exploite plus les corrélations entre les coefficients de bandelettes.

.L'utilisation de la transformée en bandelettes améliore le PSNR vu qu'elle donne des décorrélations supplémentaires dans l'exploitation de la géométrie de l'image. D'après les résultats obtenus, les meilleures performances sont atteintes avec des blocs de taille de 4*4 coefficients (rang=2).

L'augmentation de la valeur du seuil de géométrie implique une amélioration du PSNR pour le SPECK modifié avec bandelettes jusqu'à un certain seuil limite.

Le SPECK modifié emploie deux bits au lieu de quatre employés dans l'algorithme SPECK original pour coder les blocs signifiants de quatre coefficients dont trois insignifiants. A cet effet, nous avons obtenu une quantité d'informations inférieure par rapport à l'algorithme SPECK original.

Dans le SPECK original, chaque coefficient insignifiant dans le bloc de quatre coefficients -dont trois insignifiants- est codé indépendamment par un bit '0', mais dans le SPECK modifié, nous codifions les trois coefficients insignifiants dans ce bloc particulier par deux bits seulement.

Le temps de calcul pour SPECK modifié avec ondelettes est un peu plus petit que celui pour SPECK original car dans le premier, nous testons les blocs signifiants qui contiennent trois coefficients insignifiants au même temps, mais dans le deuxième le test se fait coefficient par coefficient dans ces blocs particuliers.

En se basant sur l'utilisation des pointeurs des coefficients scannés en Morton scan, l'algorithme SPECK modifié devient plus simple du point de vu de la programmation.

En perspectives, nous proposons d'associer notre algorithme SPECK modifié avec une transformation qui utilise les curvelete ou les contourlete. Ces dernières représentent l'image par un nombre de coefficients significatifs inférieur par rapport à celui obtenu par les ondelettes classiques. L'association de ces nouvelles transformations avec notre algorithme pourrait réduire en plus le nombre de coefficients à coder.

Bibliographies

[1] V. J. REHNA and M. K. JEYA KUMAR, "An improved algorithm for image compression using geometric image approximation ", International Journal of Electrical, Electronics and Data Communication, ISSN: 2320-2084, Volume-2, Issue-6. June-2014.

[2] Pragya Tiwari, Mohd Ahmed, S.G.Kerhalkarm, "Transmission of Images Using SPECK", IJCSNS International Journal of Computer Science and Network Security, VOL.14 ,No.11, November 2014.

[3] Rehna V. J and Jeya Kumar M. K. " Wavelet based image coding schemes: A recent survey", International Journal on Soft Computing (IJSC) Vol.3, No.3. August 2012.

[4] Maryam Kuchakzadeh, Habibollah Danyali and Sadegh Samadi," Progressive SAR Image Compression Using Low Complexity Bandlet Transform and Modified EZBC". Journal of Electrical Systems and Signals, Vol. 2, No. 1, March 10, 2014.

[5] KADRI Oussama. " Compression d'images fixes par Ondelettes géométriques par utilisation des Curvelets et différents types d'interpolation dans la quantification scalaire". Université Mohamed Khider – Biskra. Mémoire de magister en Electronique. 2014

[6] *R. R. Lakkundi, M. V. Latte, D. K. Deshpande "*Reduced memory listless speck image ". International Conference on Intelligent Signal Processing and Robotics, Allahabad, INDIA .February 2013.

[7] Ping Liu. Guanfeng Li. "An Improved SPIHT Algorithm for Image Compression in Low Bit Rate". Communications and Network.3, 5, 245-248.September 2013.

[8] BENYAHIA Mohamed, "Compression des Images en Couleurs Fixes en Utilisant la DWT", thèse de doctorat, Université Badjimokhtar Annaba. 2013.

[9] ZITOUNI Athmane, "Ondelettes et techniques de compression d'images numérique", thèse de doctorat en électronique. Université Mohamed khider Biskra. 2013

[10] Naimur Rahman Kidwai, M. Alam, Ekram Khan and Rizwan Beg, "A Efficient Memory No List Set Partitioned Embedded Block (NLSK) Wavelet Image Coding Algorithm for Low Memory Devices". International Journal of Signal Processing. Image Processing and Pattern Recognition Vol. 5, No. 4. December. 2012.

[11] Ritu Chourasiya and Ajit Shrivastava. "A study of image compression based transmission algorithm using SPIHT for low bit rate application". Advanced Computing: An International Journal (ACIJ), Vol.3, No.6.November 2012.

[12] AMEL BOUCHEMHA. "Etude et Application des transformées géométriques à la Compression des images hautes résolutions et à la Biométrie (Authentification/Vérification de l'empreinte palmaire". Thèse de doctorat en électronique, Université d'Annaba.. 2016.

[13] MANSOURI Dou El Kefel." La compression des images 3d". Mémoire de Magister. Spécialité : imagerie, vision artificielle et robotique médicale. Université des sciences et de la technologie d'Oran. 2011.

[14] U.S.Ragupathy. A.Tamilarasi." A Novel Method of Image Compression Using Multiwavelets and Set Partitioning Algorithm". Modern Applied Science.Vol.3, No.2. February 2009.

[15] OUAFI Abdelkrim. "Compression d'images avec pertes par codages imbriqués, Proposition d'une optimisation de l'algorithme EZW". Thèse de doctorat en Electronique Université Mohamed khider Biskra, 2009

[16] Sudhakar Radhakrishnan and Jayaraman Subramaniam, "Novel Image Compression Using Multiwavelets with SPECK Algorithm", The International Arab Journal of Information Technology, Vol. 5, No. 1, January 2008

[17] William. A. Pearlman and A. Said. "Set Partition Coding: Part I of Set Partition Coding and Image Wavelet Coding Systems". Foundations and Trends in Signal Processing .Vol. 2, No. 2. 2008.

[18] Xavier DELAUNAY. "Compression d'images satellite par post-transformées dans le domaine ondelettes". Doctorat de l'université de Toulouse. 12 novembre 2008.

[19] Gabriel Peyré, Stéphane Mallat, "Traitement géométrique des images par bandelettes". Journée annuelle de la SMF, pp.36-69. 2006.

[20] M. HETTIRI. "Etude d'algorithmes de codage imbriqué appliques à la compression de séquences d'images". Mémoire de magister en électronique. Université Mohamed khider Biskra. 2006.

[21] Erwan Le Pennec and Stéphane Mallat, "Sparse Geometric Image Representations With Bandelets", Fellow, IEEE, IEEE Transactions on image processing, VOL. 14, NO. 4. APRIL 2005.

[22] PEYRÉ, G and MALLAT, S, "Surface compression with geometric bandelets". *ACM Transactions on Graphics, Vol. 24(3).* Aug 2005.

[23]Vivien Chappelier, " Codage progressif d'images par ondelettes orientées ". Thèse de doctorat en Traitement du Signal. Université de Rennes 1, 2005

[24] Gabriel Peyré. Stéphane Mallat."A Matlab Tour of Second Generation Bandelets". CMAP, Ecole Polytechnique. The Mathworks 2005.

[25] LE PENNEC, Erwan. "Bandelettes et représentation géométrique des images". Thèse de doctorat. École polytechnique. 2002.

[26] Chérif TAOUCHE. "Implémentation d'un environnement parallèle pour la compression d'images à l'aide des fractales". Mémoire de magister en Informatique. Université Mentouri Constantine. 2005

[27] BOUCETTA ALDJIA. "Etude de l'effet des transformées de décorrélation en compression des images couleurs RGB". Mémoire de magister en informatique. Université de Batna. 2010

[28] OUKALI Salim, " Compression d'images par bandelettes : Application à des images de télédétection". Mémoire de magister en électronique, Université Mouloud Mammeri, Tizi-Ouzou, 2011

[29] Gabriel PEYRÉ. "Géométrie multi-échelles pour les images et les textures". Thèse de doctorat en mathématiques appliquées. 2005.

[30] Erwan Le Pennec, Charles Dossal, Gabriel Peyré, Stéphane Mallat. "Débruitage géométrique d'images dans des bases orthonormées de bandelettes". Colloque GRETSI, 11-14 septembre 2007, Troyes.