III.2 Méthodes de compression d'images

On distingue plusieurs méthodes de compression telle que :

III.2.1 Compression physique et compression logique

Dans le cas de compression physique on considère la compression comme un algorithme capable de comprimer énormément de données dans un minimum d'espace mémoire mais la compression logique est un algorithme de compression qui a pour but de recoder les données dans une représentation différente plus compacte contenant la même information physique.

III.2.2 Compression symétrique et asymétrique

Une méthode de compression symétrique utilise le même algorithme, et demande la même capacité de calcul, aussi bien pour la compression que pour la décompression. Par contre les méthodes de compression asymétriques demandent plus de travail dans une direction que dans l'autre. Normalement, l'étape de compression demande beaucoup plus de temps et de ressources systèmes que l'étape de décompression.

III.2.3 Compression avec et sans pertes

III.2.3.1 Méthodes sans pertes

Cela signifie que lorsque des données sont compressées et ensuite décompressées, l'information originale contenue dans les données a été préservée. Aucune donnée n'a été perdue ou oubliée. Les données n'ont pas été modifiées.

La compression sans perte trouve un intérêt particulier en imagerie, par exemple, dans les domaines liés aux applications militaires, médicales…. [18].

Dans le cas de la compression sans perte (lossless) les compromis liés à ce mode de compression sont selon trois axes :

- Efficacité du codage : ceci peut être mesure en bit par pixel (échantillon), elle est limitée par l'entropie de la source. Plus l'entropie de la source est grande plus il est difficile a compresser (exemple un bruit aléatoire).
- Temps de codage : celui-ci est lie a la complexité du processus de codage ou de décodage. Il peut être réduit si on augmente la capacité de calcul du composant de traitement. Pour certaines applications ce temps est contraint, ce qui impose le choix de la technique de codage.
-Complexité du codeur : elle peut être mesurée à l'aide de la quantité de ressources utilisées en termes de mémoire et du nombre d'opérations arithmétiques. [63]

La compression sans pertes peut être réalisée par :

- Le codage par plages (run length encoding) : consiste à coder une suite de valeurs identiques par le couple (valeur, nombre de répétitions).
- Le codage à longueur variable (Huffman par exemple) : utilise des mots binaires de longueurs différentes.
- Les plus courts (resp. longs) codent les valeurs plus (resp. moins) fréquentes.

La compression sans pertes peut également être réalisée par dictionnaire de séquences (Lempev, Zif et Welch, LZW) ou codage arithmétique [56].

III.2.3.2 Méthodes avec pertes

La méthode de compression avec pertes quant à elle "jette", de façon sélective, quelques données d'une image dans le but d'effectuer la compression avec un taux de compression meilleur que la plupart des méthodes de compression sans pertes. Les algorithmes avec pertes s'appliquent généralement aux données ayant de forts taux de redondance, comme les images, ou les sons. Certaines méthodes tirent partis d'algorithmes heuristiques élaborés qui s'ajustent eux-mêmes pour trouver le rapport de compression maximum possible en changeant aussi peu que possible les détails visibles d'une image. Autrement, d'autres algorithmes moins élégants suppriment carrément la portion la moins significative de chaque pixel. L'œil humain est limité dans le nombre de couleurs qu'il est capable de percevoir simultanément particulièrement si ces couleurs ne sont pas adjacentes dans l'image ou sont très contrastées. Un algorithme de compression intelligent peut tenir compte de ces limitations, analyser une image sur ces bases, et effectuer une réduction significative de la taille des données basée sur la suppression de l'information de certaines couleurs difficilement perceptibles par la plupart des gens.

Dans les images en noir et blanc, chaque pixel ne peut prendre que l'une des deux couleurs: noir ou blanc; même dans ces images là, si l'on ne change que quelques pixels, la différence à l'œil nu sera minime.

III.2.4 Mesurer les performances en compression

III.2.4.1 Taux de compression et débit binaire

C'est le but fondamental de la compression, il donne une mesure de performance des méthodes de compression des images fixes, autrement dit il donne une mesure de réduction de la quantité d'information à transmettre.

Le taux de compression se définit par :

$$\text{Taux}=\frac{\textit{nombre de bits dans l'image originale}}{\textit{nombre de bits dans l'image comprimée}}$$

Ce taux de compression peut être relié au débit binaire exprimé en bits par pixel (bpp) :
Le **débit** constitue une mesure alternative souvent utilisée qui donne le nombre moyen de bits nécessaire pour décrire un pixel de l'image comprimée ou encore la résolution numérique de l'image divisée par le taux de compression.

$$\text{Débit}=\frac{nombre\ de\ bits\ par\ pixel\ dans\ l'image\ originale}{taux}\ bits\ par\ pixel\,(bpp)$$

III.2.4.2 Compromis débit-distorsion et critères de qualité

Le taux de compression n'est pas le seul critère de performance d'un système de compression. Dans le cas d'une compression avec pertes, la qualité de l'image reconstruite doit aussi être prise en compte. Il y a donc un compromis à trouver entre le taux de compression et la qualité. On parle de compromis débit-distorsion. La distorsion est une mesure de l'erreur commise entre l'image originale et l'image reconstruite. Pour les mesures de distorsion, on utilisera l'erreur quadratique moyenne EQM entre l'image originale I_0 et l'image compressée I_C de taille $M \times N$:

$$EQM = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}\left(I_0(i,j) - I_C(i,j)\ \right)^2$$

Pour les mesures de qualité: on utilisera le " rapport signal sur bruit " RSB ou SNR (signal to noise Ratio). Une active variante de cette mesure est "le rapport signal sur bruit de crête à crête "*PSNR* (Peak Signal to Noise Ratio) [19] elle se mesure en décibels et défini à partir de l'*EQM* par l'équation :

$$PSNR(db) = 10\log_{10}\left(\frac{(D-1)^2}{EQM}\right)$$

Avec

N, M : longueur et largeur de l'image, respectivement.

D : le nombre de valeurs possibles que peut prendre l'échantillon. Pour une image dont les pixels sont codés sur 8 bits .D est égale à $256=2^8$.

Donc pour les images à 8 bits/pixel, le *PSNR* est donné par :

$$PSNR(db) = 10\log_{10}\left(\frac{(255)^2}{EQM}\right)$$

Il existe d'autres critères de qualité objectifs, comme l'erreur maximale ou l'erreur moyenne absolue mais nous utiliserons le PSNR car c'est une référence pour la mesure des performances en compression.

III.2.4.3 Complexité

Un troisième critère de performance d'un système de compression est la complexité. La complexité calculatoire peut être mesurée par le temps d'exécution du processus de compression ou en nombre d'opérations par pixel : c'est le nombre moyen d'opérations qui sont nécessaires à la compression de l'image.

III.3 Compression d'image par transformée en ondelettes

Nous nous sommes intéressés aux méthodes de compression basées sur la transformée en ondelettes à cause des propriétés de celle-ci. En effet, la transformée en ondelettes associe de bonnes localisations spatiales et fréquentielles.

La transformée en ondelettes est utilisée pour l'analyse des images à comprimer sur différents niveaux de décomposition [11,27]. Ces niveaux de décomposition contiennent un nombre de sous-bandes, composées chacune de coefficients décrivant les caractéristiques (horizontales, verticales et diagonales) de l'image originale.

Aujourd'hui la théorie des ondelettes ou la transformation en sous-bandes regroupe Les méthodes de décomposition en multirésolution [29,55].

III.3.1 Décomposition en ondelettes séparables

La décomposition en ondelettes d'une image se déroule donc de la manière suivante :

- Dans un premier temps, chaque colonne de l'image 2-D est décomposée en utilisant verticalement les filtres 1-D. Cela produit deux *sous-bandes*, l'une correspondant aux basses fréquences verticales, l'autre aux hautes fréquences verticales.
- Dans un second temps, les mêmes filtres 1-D sont appliqués aux lignes des ces deux bandes. Chacune des deux bandes de départ est alors décomposée en deux nouvelles *sous-bandes*, l'une correspondant aux basse fréquences horizontales, l'autre aux hautes fréquences horizontales (figure 3.1).

Au total, quatre *sous-bandes* (LL, HL, LH, HH) sont donc générées à chaque niveau de décomposition. La décomposition suivante effectue le même processus sur la sous-bande *LL*, correspondant aux basses fréquences horizontales et verticales. La figure (3.2) illustre l'effet des différents filtres sur l'image originale.

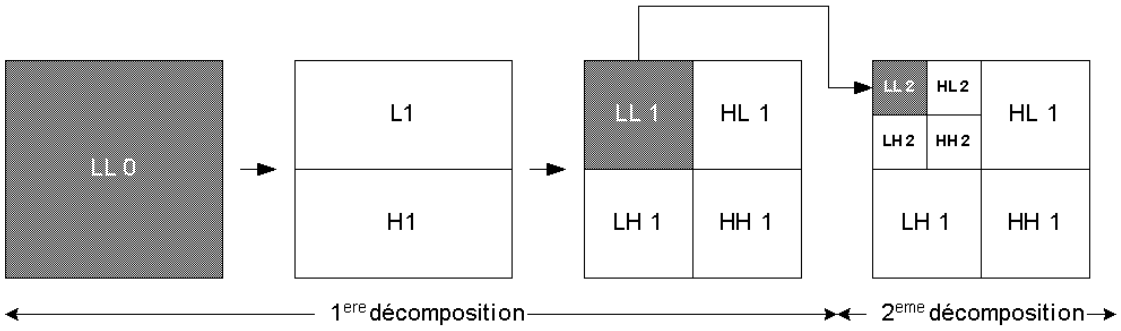


Fig. 3.1 : Décomposition en ondelettes a deux niveaux

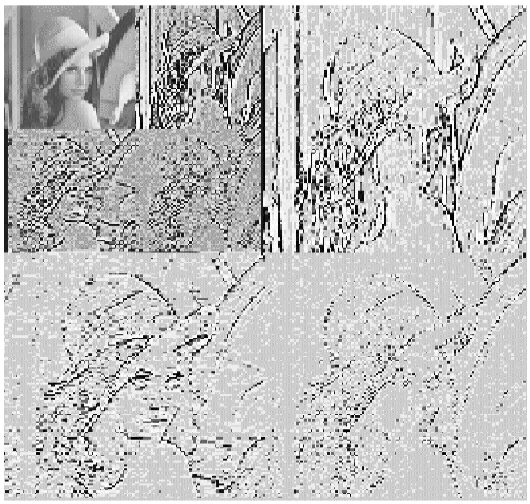


Fig. 3.2 : Exemple de décomposition en ondelettes

III.3.2 Quantification

La quantification est l'opération fondamentale d'un système de conversion analogique/numérique voir figure 3.3

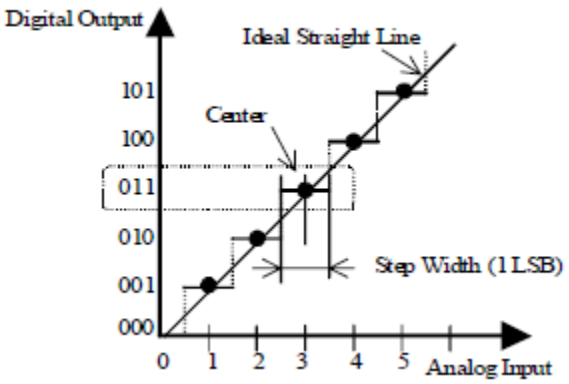


Figure 3.3 Caractéristique entrée - sortie d'un CAN

. Son but est de sélectionner, pour une valeur d'entée donnée, le plus proche voisin appartenant à un ensemble fini prédéterminé de valeurs numériques. [60]

Les coefficients à la sortie de la transformation (ou des filtres d'analyse dans le cas d'un banc de filtres) prennent en général des valeurs réelles qui doivent être quantifiées pour réaliser la compression. La quantification dans sa forme la plus simple consiste à quantifier chaque coefficient séparément, on parle alors de quantification scalaire. Une autre méthode plus complexe consiste à quantifier plusieurs coefficients à la fois, c'est la quantification vectorielle.

III.3.2.1 Quantification scalaire

La transformation en ondelettes est suivie d'une quantification scalaire. Avec une quantification scalaire, la dynamique d'entrée est divisée en une famille finie d'intervalles $P_i = [x_{i-1}, x_i[$ et la valeur de sortie y_i est typiquement choisie dans l'intervalle P_i.

L'opération appelée "codage" consiste à trouver l'intervalle auquel appartient une valeur d'entrée $x(n)$ et à lui associer le numéro de cet intervalle $i(n) \in 1.....,L$ qui sera transmis où stocké. L'opération de "décodage" consiste à associer au numéro $i(n)$ la valeur $y_i(n)$ correspondante dans le dictionnaire :

$$\hat{x}(n) = y_i(n)$$

Ainsi, la quantification est un processus à deux étapes :

$$x(n) \text{ codage } x(n) \xrightarrow{codage} i(n) \xrightarrow{décodage} y_i(n) = \hat{x}(n)$$

et l'erreur de quantification est exprimée par:

$$q(n) = x(n) - \hat{x}(n)$$

où $x(n)$ et $\hat{x}(n)$ sont respectivement l'entrée et la sortie du quantificateur. [61].

La quantification scalaire uniforme avec zone morte (deadzone) figure3.4 permettant de réduire la dynamique des données. Cette opération est destructive, sauf si le pas de quantification est égal à 1 et les coefficients sont des entiers, comme ceux produits par l'ondelette réversible 5/3. Chaque coefficient de la transformation $c_b(u,v)$, de la sous-bande b est quantifié à valeur $q_b(u,v)$ suivant l'équation [35,40,50, 56-58] :

$$q_b(u,v) = signe\,(c_b(u,v)) \left\lfloor \frac{|c_b(u,v)|}{\Delta_b} \right\rfloor$$

Où

Δ_b : est le pas de quantification fonction de la dynamique et du type de la sous-bande, du nombre de niveaux de décomposition.

La dimension du pas de quantification Δ_b est représentée par rapport à la dimension dynamique de la sous-bande *b* .La dimension dynamique dépend du nombre des bits utilisés pour représenter l'image de l'image originale et du choix de la transformée en ondelettes. Les gains des filtres LL, LH, HL et HH n'étant pas identiques, on utilise un pas de quantification différente suivant la sous-bande à laquelle appartiennent les coefficients (seulement dans le cas d'une compression avec pertes). Pour la compression réversible, la taille du pas de quantification doit être égale à 1.

Δ_b sera représenté sous une forme (ε_b, μ_b) correspondant à :

$$\Delta_b = \left(1 + \frac{\mu_b}{2^{11}}\right). 2^{R_b - \varepsilon_b}$$

R_b est la dynamique du signal d'origine (nombre de bits), par exemple R_b= 8 pour un plan codé sur 8 bits [0..255], ε_b est la dynamique voulue des coefficients et μ_b un facteur multiplicatif permettant d'avoir des valeurs de Δ_b différentes des multiples 2^ (N), avec N entier positif. Seules les valeurs ε_b et μ_b sont stockées dans le fichier binaire. Dans le cas d'une compression réversible, seule la valeur $R_b = \varepsilon_b$ est signalée.[62]

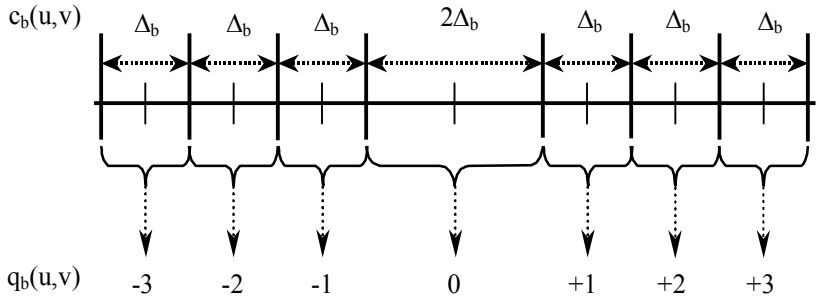


Figure 3.4 : Quantification scalaire uniforme avec zone morte

III.3.2.2 Quantification vectorielle

La quantification vectorielle n'est qu'une généralisation de la quantification scalaire à plusieurs dimensions. Elle permet de prendre en compte directement la corrélation entre échantillons voisins du signal, plutôt que de chercher d'abord à réduire la redondance au

moyen d'une transformation avant de quantifier, comme le fait la quantification scalaire prédictive.

La quantification vectorielle est appliquée à chaque vecteur $x(m)$ de dimension N constitué de N échantillons consécutifs du signal. Les vecteurs $\hat{y}^i = [\hat{y}_0{}^i .....\hat{y}^i_{N-1}]^T$ représentent les vecteurs symboles à la sortie du quantificateur. L'ensemble des L vecteurs $\hat{y}^i$ assimilable à une matrice est le dictionnaire.

L'inconvénient de la quantification vectorielle est sa complexité, qui limite la taille des vecteurs $\hat{y}^i$ utilisés. Pour contourner ce problème le dictionnaire est structuré de façon à simplifier la recherche du représentant, étant donné le vecteur d'entrée. Ceci est réalisable avec la quantification vectorielle arborescente. Une autre approche est d'utiliser des tranformations linéaires (telle que la décomposition en sous-bandes et les transformées en ondelettes) et d'appliquer la quantification vectorielle sur les coefficients.

III.3.3 Codage des sous-bandes

Le but des transformées présentées dans la section précédente est de décorreler les données brutes de l'image représentées par ses pixels. Cette décorrélation n'est cependant pas parfaite et les coefficients obtenus après transformation restent dépendants statistiquement. Ainsi, bien qu'une loi gaussienne généralisée puisse représenter avec fidélité la statistique du premier ordre des sous-bandes, seuls les codeurs exploitant l'information mutuelle résiduelle entre les coefficients ont permis d'obtenir des performances bien meilleures que les codeurs basés sur la quantification vectorielle.

De plus, les transformées en ondelettes offrent naturellement une représentation progressive de l'image, il est intéressant de conserver cette propriété lors du codage des sous-bandes. Ainsi, dans les codeurs emboîtés (embedded), la quantification et le codage sont également réalisés de manière progressive, en commençant par coder partiellement les coefficients de plus forte amplitude, puis en raffinant la quantification de ces derniers en les codant de nouveaux. Nous commençons par présenter les codeurs non progressifs, puis les codeurs emboîtés basés sur des structures d'arbres ou de blocs.

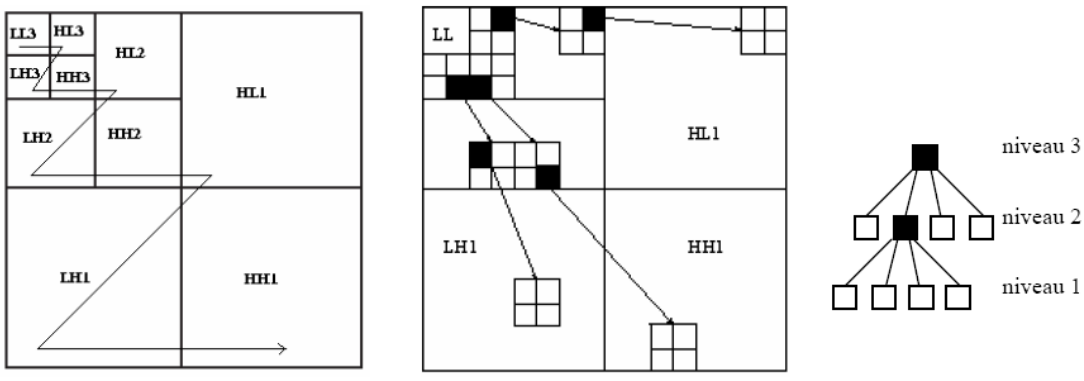
Plusieurs algorithmes de compression utilisant les ondelettes ont été proposés, dont les plus utilisés sont le EZW, le SPIHT, EZBC, EBCOT …

III.3.3.1 L'algorithme de codage EZW

L'idée de base de cet algorithme qui est proposé par Shapiro [41]. Est de trouver le meilleur ordre de transmission des coefficients de représentation en ondelettes.

Il est clair que la transmission des coefficients dans l'ordre décroissant de leur valeur absolue est la meilleure solution, puisque les coefficients les plus signifiants sont ceux dont la valeur absolue est la plus élevée. Shapiro proposa de transmettre les coefficients sous forme d'une suite de bits obtenue par enchaînement progressif des bits des coefficients les plus signifiants en commençant par les bits les plus importants. Cette nouvelle conception offre l'avantage à l'algorithme EZW de faire la transmission progressive d'image puisque le décodeur peut s'arrêter au niveau de n'importe quelle suite de bits.en plus nous aurons une meilleure image reconstruite avec cette suite de bits tronquée. Cet algorithme présente en plus l'avantage de ne nécessiter ni phase d'apprentissage, ni dictionnaire, ni l'information sur l'image source. [57]

Après avoir calculé la transformée en ondelettes de l'image, l'algorithme code les coefficients transformés à l'aide d'une suite décroissante de seuils $T_0.....T_{N-1}$, avec $T_{i+1} = \frac{T_i}{2}$ *et* $T_0 \langle 2|c|$ pour tout coefficient *c* de la représentation en ondelettes. Pour coder les coefficients, l'algorithme effectue récursivement deux passes successives, ne traitant à chaque fois que les coefficients significatifs par rapport au seuil courant : ceux dont la valeur absolue est supérieure au seuil. Dans la première passe, la dominante de l'algorithme parcourt les coefficients de la transformée en ondelettes suivant l'ordre donné par la figure (3.5-a) pour la recherche des coefficients significatifs par rapport au seuil courant, en utilisant la hiérarchie donnée par la figure (3.5-b).



a- ordre de parcours des coefficients b- organisation hiérarchique des coefficients

Figure 3.5 : les relations entre les coefficients d'ondelettes dans différents sous bandes

III.3.3.2 SPIHT

SPIHT (*Set Partioning In Hierarchical Trees*) est un algorithme récemment proposé par Amir Said et William A.Pearlman [42], basé sur une transformation en ondelettes discrètes (*DWT*).

Normalement, la plus grande partie de l'énergie d'une image est concentrée dans les basses fréquences. Par conséquent, plus le niveau de résolution de l'image est grand, plus les variances des images de détails doivent être faibles. De plus, il semble qu'il existe une similarité entre les pixels de même orientation spatiale et de résolutions différentes. En particulier, les amplitudes des pixels de même orientation spatiale semblent être assez bien ordonnées si l'on évolue dans la pyramide. On est alors naturellement conduit à adopter une organisation arborescente des coefficients (figure3.6), où chaque nœud de l'arbre correspond à un pixel et ses descendants directs aux pixels de même orientation spatiale à la résolution suivante.

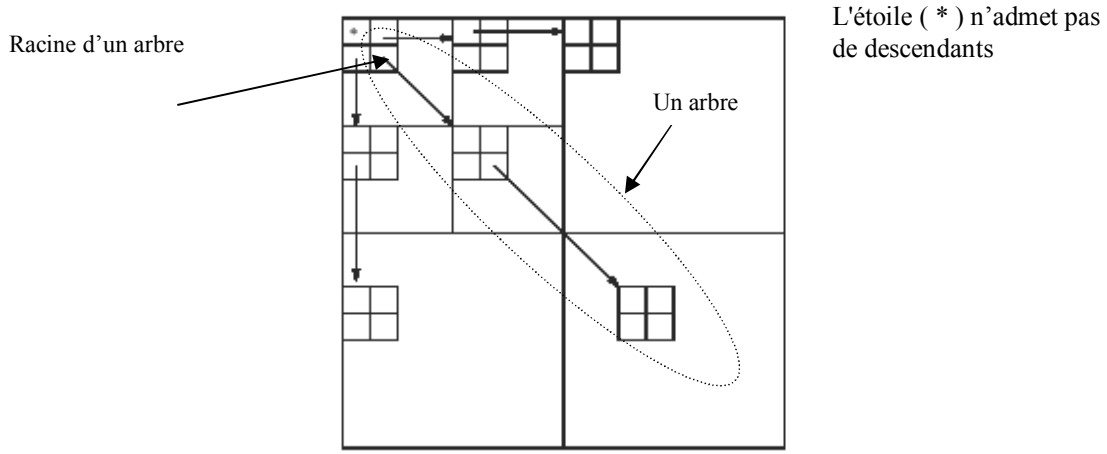


Figure 3.6 : Dépendance de coefficients et arbres d'orientation spatiale dans SPIHT utilisant la décomposition dyadique avec deux résolutions

L'algorithme SPIHT utilise trois listes ordonnées qui contiennent les ensembles de test de la signifiance avec la même méthode pour le codeur et décodeur, la liste des ensembles insignificatifs (**LIS**), la liste des pixels insignificatifs (**LIP**), et la liste des pixels significatifs S (**LSP**). Il y a des listes de coordonnées (i, j) comme dans LIP et LSP qui représentent des coefficients individuels, mais la liste LIS représente des ensembles des

coefficients, chaque ensemble est soit un ensemble D (i, j) (marqué d'une entrée de type A) ou un ensemble L(i,j) (marqué d'une entrée de type B) [42].

III.3.3.3 SPECK

Offrant des performances comparables à SPIHT, l'algorithme SPECK [27] exploite des structures d'ensembles de coefficients non signifiants en blocs plutôt qu'en arbres. Ces structures de blocs permettent de s'affranchir efficacement de la non stationnarité des coefficients en adaptant localement la statistique utilisée pour le codage.

Les coefficients sont initialement séparés en deux ensembles, l'un noté S contenant les coefficients de basses fréquences et l'autre, noté I contenant les autres coefficients. De la même manière que dans SPIHT, deux listes sont tenues à jour, pour représenter les coefficients significatifs (LSP) et les ensembles de coefficients non significatifs (LIS). La liste d'ensembles non significatifs contient des blocs de coefficients de taille variable, y compris les coefficients isolés vus comme des blocs de 1x1 (stockés dans la LIP dans SPIHT). Cette liste est triée des blocs de plus petite taille aux blocs de plus grande taille. Lors du déroulement de l'algorithme, un test de signifiance est réalisé sur chaque ensemble de la LIS à tour de rôle.

Si l'ensemble est signifiant et non réduit à un seul coefficient, il est retiré de la liste et partitionné récursivement en quatre sous blocs sur lesquels ce test est effectué à nouveau. Si le bloc est réduit à un seul coefficient significatif, celui ci est ajouté à la LSP. Dans tous les autres cas, l'ensemble est laissé ou ajouté dans la LIS. Les autres coefficients appartenant à I sont traités ensuite. Si cet ensemble est signifiant, il est séparé en trois blocs de coefficients correspondants aux sous-bandes de plus basses fréquences, et en un nouvel ensemble I contenant le reste des coefficients. Ces trois nouveaux blocs sont traités comme précédemment. Ce processus de séparation de l'ensemble I est répété jusqu'à ce qu'il soit insignifiant. Le codage des bits de raffinement est par ailleurs identique à SPIHT.

Cet algorithme regroupe plusieurs idées développées précédemment. Tout d'abord, le partitionnement adaptatif en quad-tree, développée dans [25] et réalisé lors de la séparation de S, permet de repérer rapidement les régions hautement énergétiques et de les coder avec un nombre minimum d'information de signifiance.

La séparation récursive en sous-bandes réalisée sur l'ensemble I et initialement introduite dans [1] permet d'en exploiter la structure hiérarchique en s'intéressant d'abord aux sous-bandes de plus haute énergie a priori. Combinées avec le codage du partitionnement proposé dans SPIHT, ces techniques donnent un codeur par blocs offrant des résultats

similaires. Le codage entropique est uniquement effectué sur les bits de signifiance dus à la séparation d'un ensemble de type *S* en quatre sous ensembles. A nouveau, un codeur arithmétique contextuel adaptatif est utilisé.

Cette fois-ci, les décisions binaires sont codées une par une (l'alphabet restant donc binaire), en utilisant les signifiances des coefficients précédents comme contexte.

Ainsi, la signifiance du premier coefficient est codée à l'aide d'un contexte unique, celle du second à l'aide de deux contextes, et celle du troisième à l'aide de quatre contextes. Un traitement particulier est réalisé pour le dernier coefficient. Si les trois autres coefficients sont insignifiants aucun bit n'est transmis car il est certain que ce dernier coefficient est signifiant (autrement l'ensemble n'aurait pas été séparé).

Sinon un codage contextuel est réalisé à l'aide des huit contextes formés par la signifiance des trois coefficients précédents.

D'abord aux sous-bandes de plus haute énergie a priori. Combinées avec le codage du partitionnement proposé dans SPIHT, ces techniques donnent un codeur par blocs offrant des résultats similaires. Le codage entropique est uniquement effectué sur les bits de signifiance dus à la séparation d'un ensemble de type *S* en quatre sous ensembles. A nouveau, un codeur arithmétique contextuel adaptatif est utilisé.

Cette fois-ci, les décisions binaires sont codées une par une (l'alphabet restant donc binaire), en utilisant les signifiances des coefficients précédents comme contexte. Ainsi, la signifiance du premier coefficient est codée à l'aide d'un contexte unique, celle du second à l'aide de deux contextes, et celle du troisième à l'aide de quatre contextes. Un traitement particulier est réalisé pour le dernier coefficient. Si les trois autres coefficients sont insignifiants aucun bit n'est transmis car il est certain que ce dernier coefficient est signifiant (autrement l'ensemble n'aurait pas été séparé).

Sinon un codage contextuel est réalisé à l'aide des huit contextes formés par la signifiance des trois coefficients précédents.

III.3.3.4 EBCOT

Le codeur EBCOT (*Embedded Block Coding with Optimized Truncation*) est issu des travaux de Taubman [50] mais n'appartient pas à la famille des schémas de codage à arbre de zéros. Le codeur EBCOT est la référence pour la compression d'images à base de la transformée en ondelettes. [48].

L'algorithme EBCOT, évolution de l'algorithme LZC (Layered Zero Coding) est l'algorithme de codage progressif adopté par le standard de codage d'images fixes JPEG-2000. Il permet un codage emboîté, grâce `a la décomposition en ondelettes d'une image. Son originalité réside dans le réordonnancement du train binaire final, destiné à présenter une progressivité optimale du point de vue débit distorsion.

Description générale

Après la transformée en ondelettes chaque sous-bande est tout d'abord partitionnée en blocs, de taille 64 × 64. Un train binaire indépendant est alors généré pour chacun de ces blocs. Chaque train binaire ainsi formé est progressif en qualité, c'est-à-dire qu'il peut être tronqué à des points, correspondant respectivement à une distorsion. La propriété intéressante de cet ensemble de points de troncature est que la plupart d'entre eux sont situés sur la courbe débit distorsion du bloc considéré.

Afin de générer un train binaire progressif pour représenter l'image entière, EBCOT organise le train binaire final en couches de qualité. Étant donné un débit alloué `a une couche donnée, le train binaire de chacun des blocs est tronqué de fa¸con à minimiser la distorsion globale associée au décodage de la couche considérée.

L'organisation des couches de qualité est illustrée figure 3.7. Dans cet exemple, les blocs 4 et 7 ne contribuent pas à la couche de qualité n°1.

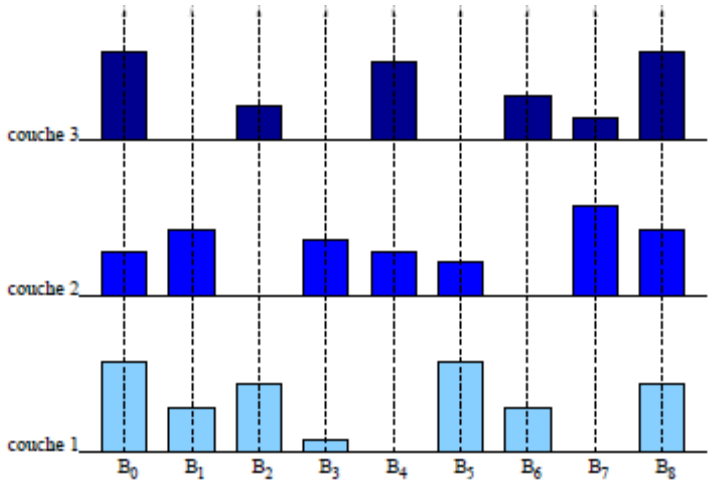


Figure 3.7: Organisation des couches de qualité dans EBCOT.

L'algorithme EBCOT se divise donc en deux étapes, la premiére consistant en l'encodage progressif et indépendant de tous les blocs des différentes sous-bandes, la seconde en le d´ecoupage et le réordonnancement de ces trains binaires élémentaires, comme l'indique la figure 3.8.

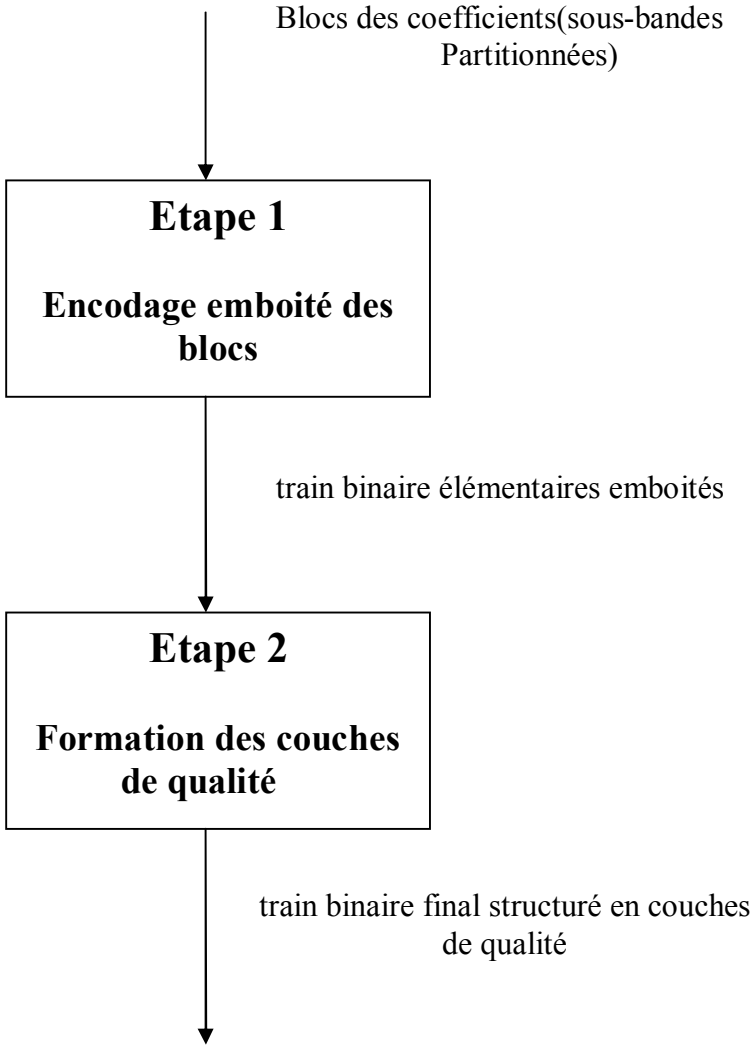


Fig. 3.8: Briques principales de l'algorithme EBCOT.

Codage emboltés des blocs de sous-bandes

Signifiance des sous-blocs . Ce point n'apparait plus dans le standard JPEG-2000, mais fait partie de l'algorithme EBCOT original. L'idée consiste à découper les blocs Bi en sous blocs de manière à définir un arbre (quad-tree). On utilise alors un codage iérarchique pour coder efficacement les sous-blocs non signifiants, c'est-`a-dire qui ne contiennent que

des coefficients d'amplitude plus faible que le pas de quantification courant10.

Signifiance des coefficients: Au sein d'un sous-bloc signifiant, le codage des coefficients est effectué à partir de plusieurs routines de codage. Tout d'abord, la signifiance des coefficients non encore signifiants est cod´ee par codage arithmétique contextuel adaptatif, appelé Zero Coding (ZC) ou par Run-Length Coding (RLC), selon les propriétés de stationnarité du voisinage.

Codage du signe et raffinement de l'amplitude Lorsqu'un coefficient s'est révélé signifiant au regard du pas de quantification courant, on code son signe avec une autre routine de codage arithmétique contextuel adaptatif, appelée Sign Coding (SC). A partir du prochain pas de quantification (c'est-à-dire du prochain plan débit), la valeur du coefficient sera affinée par Magnitude Refinement (MR), troisième primitive de codage arithmétique contextuel adaptatif.

III.3.3.5 EZBC

Le codec EZBC (*Embedded ZeroBlocks with Context modeling*) est basé sur le codec SPIHT en utilisant un codeur arithmétique contextuel, prenant en compte le voisinage du coefficient courant. L'algorithme EZBC est ainsi capable de prendre en compte les relations de dépendance statistique existant entre les coefficients d'une même sous-bande. Il offre une efficacité de codage supérieure au codec SPIHT et est à la base du schéma de codage vidéo MC-EZBC.

Le principe de codage est similaire à SPECK, l'innovation de ce codeur provenant 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.

La première étape de codage consiste à créer un quad-tree de signifiance pour chaque sous bande. Les feuilles de cet arbre représentent les coefficients de la sous-bande en question, tandis que les noeuds internes sont initialisés à la valeur d'amplitude maximale de tous leurs descendants. Ainsi le noeud racine correspond à l'amplitude maximale de la sous-bande. Les différentes couches de l'arbre sont codées du noeud racine aux feuilles en testant la signifiance des noeuds. De même que dans SPIHT et SPECK, ce codage s'effectue très rapidement en s'appuyant sur la gestion de listes de coefficients et 'ensembles.

La dépendance entre les noeuds de l'arbre est exploitée lors du codage entropique contextuel. Pour chaque noeud, un contexte est formé à partir de l'état de signifiance de ses huit voisins situés au même niveau de l'arbre, ainsi que du noeud situé dans l'arbre de la sous-bande parente au niveau inférieur. Le fait de considérer le noeud de la sous-bande parente au niveau inférieur tient compte du changement d'échelle entre les sous-bandes et permet d'exploiter la dépendance inter-échelle. Les 512 états possibles sont réduits à 20 contextes comme dans EBCOT, pour améliorer l'apprentissage et l'adaptation des lois associées.

III.4 Comparaison des différents codeurs

Le succès des approches par ondelettes en compression s'explique en grande partie du fait de l'apparition de codeurs de sous-bandes efficaces. Le codeur EZW est le premier à avoir fourni des performances débit-distorsion remarquables tout en permettant un décodage progressif de l'image. Le principe des zerotrees, ou d'autres structures de partitionnement en ensembles de zéros, permet en effet de tenir compte de dépendance résiduelle des coefficients entre eux. Plus précisément, les coefficients de haute énergie étant groupés spatialement, leur position est codée efficacement par complémentarité en indiquant la position des ensembles de coefficients peu énergétiques. Après séparation des coefficients peu énergétiques et des coefficients énergétiques, ces derniers sont relativement indépendants et peuvent être codés efficacement à l'aide de techniques de quantification et de codage entropique simples.

Les codeurs successifs comme SPIHT et SPECK ont amélioré le principe de EZW enproposant un codage plus efficace de l'information de signifiance qui code la position des coefficients énergétiques. Comme cette information représente une grande part du débit de l'image, son codage efficace est primordial. Ce même principe est encore utilisé dans les codeurs les plus récents que sont EBCOT et EZBC, offrant des performances supérieures à SPIHT du fait de l'utilisation de contextes de codages de plus grande dimension, au prix d'une complexité accrue. Notons cependant que l'information de signifiance est fragile et qu'il est essentiel, dans un environnement bruité, de la protéger pour obtenir une image décodée correcte

Parmi les codeurs progressifs, les codeurs EZW, SPIHT et SPECK supposent que l'énergie des coefficients d'ondelettes décroît des échelles grossières aux échelles fines, et que les coefficients signifiants ont probablement des fils signifiants. Les coefficients sont codés dans un ordre de parcours défini par l'algorithme et fixe pour toutes les images.

En revanche, les codeurs EZBC et EBCOT permettent de créer des flux binaires progressifs indépendants. En particulier, le codeur EBCOT permet de coder chaque bloc de 32x32 coefficients indépendamment. En enregistrant les points de troncature de chaque flux binaire et les distorsions correspondantes, une optimisation débit distorsion est alors réalisable a posteriori. Ainsi, il est à la fois possible d'obtenir une représentation non progressive de l'image en ne réalisant l'allocation de débit que pour un seul débit de codage (EBCOT SL), et une représentation progressive en réalisant l'allocation de débit pour plusieurs débits cibles. Cette information sur la répartition du débit entre les différents blocs doit toutefois être représentée dans le flux binaire final, ce qui explique les performances légèrement inférieures du mode de codage progressif (EBCOT GS).

III.5 Conclusion

Le but de la compression d'image est donc de modifier la représentation initiale des données, pour qu'elles occupent moins de place. Cette nouvelle représentation sera décodée durant une procédure de décompression pour reconstruire l'image.

Plusieurs algorithmes de compression utilisant les ondelettes ont été proposés, dont les plus utilisés sont le EZW, le SPIHT, EZBC, EBCOT …

En remarque que chaque codeur a des inconvénients et des avantages, ainsi il n'ya pas des codeurs universel utilisé dans l'opération de compression donc on peut déduire que le fais de changer l'image on doit changer l'algorithme de compression.