République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Ecole Nationale Polytechnique
Département d'Electronique
Laboratoire Signal & Communication

Thèse de Doctorat
en Electronique

Option : Traitement du Signal et Communications

Présentée par :
MEKHALFA Faiza
Magistère en Electronique de l'ENP

Intitulé

Compression d'images à base de la géométrie fractale

Soutenue publiquement le 01/06/2016 devant le jury composé de :

Président :	M^{me} HAMAMI Latifa	Professeur	ENP
Directeur :	Mr. BERKANI Daoud	Professeur	ENP
Examinateurs :	Mr. GUESSOUM Abderrezak	Professeur	U. de Blida
	Mr. AISSAT Abdelkader	Professeur	U. de Blida
	Mr. SMARA Youcef	Professeur	USTHB
	Mr. BOUSBIA-SALAH Hicham	Maître de conferences A	ENP

ENP 2016

ملخص :

تقليص الصور الرقمية بطريقة الهندسة الكسيرية و أنظمة الدوال المتكررة، ترتكز على إنشاء دالة خاصة للصورة الأصلية، تدعى بالرمز الكسيري. يتم تكرار هذا الرمز على أي صورة إبتدائية لإعطاء سلسلة من الصور، تتقارب نحو التمثيل الكسيري للصورة الإبتدائية. الجزء الأول من هذه الأطروحة يتضمن ثلاث طرق تعتمد على الكسيريات، وتستعمل عدة أصناف من التقسيمات: التقسيم المربع الثابت، تقسيم على شكل شجرة، تقسيم بواسطة مثلثات Delaunay. في الجزء الثاني قدمنا خواريزمية لتقليص الصور باستعمال الهندسة الكسيرية ونظريات الموجات الصغيرة. ثم إقترحنا طريقة جديدة تسمح بتحسين فعالية الخوارزمية السابقة بتهجين الكسيريات و الموجات الصغيرة مع طريقة تقليص عكسية.

الكلمات المفتاحية: أنظمة الدوال المتكررة، تقليص الصور بالكسيريات، تحويل الموجات الصغيرة، تقليص مهجن بالكسيريات و الموجات الصغيرة، خوارزمية Huffman، تقليص عكسي بالكسيريات و الموجات الصغيرة، صور إشعاعية لعيوب الإلحام

Abstract:

Fractal image compression based on concept of iterated functions systems, consists of partitioning blocks and approximates each block by a transformed codebook block derived from image itself. In this work, we present three fractal compression techniques employing different partitioning schemes: fixed size square block, quadtree decomposition and Delaunay triangulation. In the second part, we develop a wavelet-fractal coder (WFC), which combines fractal compression and wavelet transform. Then we propose a new compression scheme by incorporating the combined concept of fractal coding and wavelet theory with a lossless compression method.

Key words: Iterated functions systems, Image fractal compression, discrete wavelet transform, hybrid wavelet fractal-coder, Huffman algorithm, lossless wavelet-fractal compression, radiographic images of weld defects.

Résumé :

La compression des images par fractales fondée sur la théorie des systèmes de fonctions itérées, consiste à approximer chaque bloc d'une partition à l'aide d'une transformation locale contractante, appliquée sur une autre partie de l'image. Dans le cadre de cette thèse, nous présentons trois méthodes de compression fractale basées sur différents modèles de partitionnement géométrique : le partitionnement carré, le partitionnement en arbre quaternaire et la triangulation de Delaunay. Dans la deuxième partie nous développons un algorithme de compression basé sur la théorie des ondelettes et le formalisme mathématique des fractales. Nous proposons ensuite un schéma de compression sans perte en combinant le codage hybride fractale-ondelette avec une autre méthode de compression sans perte.

Mots clefs : Systèmes de fonctions itérées, Compression fractale d'images, Transformée en ondelette discrète, codage hybride fractale-ondelette, algorithme de Huffman, Compression fractale-ondelette sans perte, Images radiographiques des défauts de soudures

A la mémoire de mon Père

Remerciements :

Je tiens à remercier le Professeur BERKANI Daoud d'avoir dirigé mes travaux de thèse et de son soutien tout au long des années de préparation de cette thèse.

Mes reconnaissances sont destinées à :

Mme HAMAMI Latifa, Professeur à l'ENP qui me fait l'honneur de présider le Jury de cette thèse.

Mr. GUESSOUM Abderrezak professeur à l'U de Blida, Mr. AISSAT Abdelkader Professeur à l'U de Blida, Mr. SMARA Youcef professeur à l'USTHB et Mr. BOUSBIA-SALAH Hicham Maître de conférences A à l'ENP, d'avoir accepté de juger ce travail.

Je tiens à exprimer mes remerciements à Mr. Mohammad reza nasiri Avanaki Docteur à Wayne State University (WSA) – USA pour son aide précieux.

Table des matières:

Liste des tableaux :

Liste des figures :

Liste des abréviations :

LZ : Lempel Ziv

LZW : Lempel Ziv Welch

RLE : Run Length Encoding

DPCM : Differential Pulse Code Modulation

ADPCM : Adaptive Differential Pulse Code Modulation

KLT : Karhunen-Loeve Transform

DFT : Discrete Fourier Transform

DCT : Discrete cosine transform

DWT : Discrete Wavelet Transform

FFT: Fast Fourier Transform

QV: Quantification vectorielle

JPEG : Joint Photographic Expert Group

ISO : International Standards Organisation

CEI : Commission Electronic International

RGB : Red Green Blue

YCbCr : Luminance Chrominance

DC : Direct Current

AC : Alternative Current

JPEG-LS : Joint Photographic Expert Group Lossless

LOCO-I : Low Complexity Lossless Compression method

MED : Median Edge Detection

MPEG : Moving Pictures Experts Group

IFS : Iterated Functions System

L-IFS : Local Iterated Functions System

QD : Quadtree Decomposition

QR : Quadtree Recomposition

H-V : Horizontal Vertical

LBG : Linde Buzo Gray

EZW : Embedded Zerotree Wavelet coder

SPIHT : Set Partitionning In Hierarchchal Trees

EBCOT : Embedded block coding with optimized truncation

PPC: Predictive Pyramid Coder

SQS : Self Quantization of Subtree

SVH : Système de Vision Humain

PSNR: Peak Signal to Noise Ratio

FIC: Fractal Image Coder

WFC: Wavelet Fractal Coder

WFC-LS : Wavelet Fractal Coder Lossless

Introduction générale :

La qualité visuelle des images numériques s'améliore de plus en plus avec le développement de nouvelles techniques d'affichage et technologies d'acquisition. Les appareils photo numériques atteignent une qualité méga pixel aujourd'hui. Cependant la taille de ces images augmente proportionnellement avec leur qualité. La compression d'image est devenue donc une tâche essentielle pour optimiser l'utilisation de ces grands volumes d'information dans le stockage et la transmission.

La compression d'images associe plusieurs disciplines et connaissances à la fois en imagerie, en analyse de données, qu'en théorie de l'information. Il existe deux types de compression : la compression sans perte et avec pertes. Dans la compression sans perte, l'image reconstruite après décompression est identique à l'image originale. Par contre, la compression avec pertes introduit une certaine distorsion plus ou moins perceptibles à l'œil mais permet une compression bien plus importante que celle obtenue par des méthodes sans perte [1].

La compression fractale fait partie des méthodes de compression avec pertes. Elle est devenue l'un des champs privilégiés d'investigation en compression d'images. Le procédé de cette méthode est basé sur des appariements entre des portions de l'image, qui correspondent au même motif pour des échelles et des orientations différentes, à l'aide de quelques transformations contractantes définissant un système de fonctions itérées (IFS) [2]. Jacquin [3] a proposé un schéma pratique de codage fractale par bloc, qui consiste à rechercher les autosimilarités locales ou partielles dans l'image et de les représenter sous forme de transformations affines contractantes. Les principaux avantages de cette technique sont : des taux de compression élevés et une procédure de décompression particulièrement simple et rapide. L'inconvénient majeur de cette méthode est que la procédure de compression est très lente et que la qualité d'image est mauvaise à faible débit. Depuis que Jacquin [3] a proposé ce schéma, de nombreuses recherches ont été entamées visant à accélérer la phase du codage et améliorer la qualité de l'image reconstruite. A cet effet, plusieurs travaux se sont penchés sur deux axes de recherche: l'axe de partitionnement de l'image [4] et l'axe d'accélération du codage avec l'élaboration des schémas hybrides basés sur la transformation en cosinus discrète ou en ondelettes [5][6][7].

La théorie des ondelettes a connu un réel essor depuis quelques années concrétisé par un grand nombre d'applications. La transformée en ondelettes est une transformation qui est bien localisée à la fois en espace et en fréquence et qui admet la non-stationnarité du signal. De plus, la décomposition en ondelettes de l'image permet d'obtenir un ensemble de sous-bandes

ayant de meilleures caractéristiques que l'image d'origine : réorganisation de l'information, plus faible entropie et contours orientés. L'intérêt de l'introduction de la théorie des ondelettes en compression d'images par fractales est justifié par l'existence des autosimilarités dans la décomposition multirésolution. Dans un codeur hybride fractale-ondelette, l'image est d'abord décomposée par la transformation en ondelette et le codage par fractales est ensuite appliqué aux coefficients résultants.

Nous étudions dans le cadre de cette thèse plusieurs méthodes de compression d'images selon l'approche fractale opérant sur trois modèles de partitionnement géométriques (carré, quadtree, triangulaire). Nous développons par la suite le codeur hybride fractale-ondelette et nous montrons sa supériorité par rapport à l'algorithme standard de compression fractale. A base de cet algorithme nous proposons un codeur hybride-fractale ondelette sans perte d'information. Le schéma proposé permet de restituer l'image originale sans aucune distorsion, en combinant le formalisme mathématique des fractales et la théorie des ondelettes avec l'algorithme de Huffman. Les résultats de simulation sur des images standards et des images radiographiques des défauts de soudures montrent que le schéma proposé surpasse l'algorithme de Huffman.

L'organisation générale de la thèse est décrite ci-dessous :

Le premier chapitre présente des généralités sur la théorie de l'information et le codage en imagerie où nous exposons les différentes méthodes de compression des images numériques.

Le second chapitre est réservé à la compression fractale d'images. Dans un premier temps, nous allons aborder la théorie des fractales, puis nous présentons une revue des principales techniques de compression basées sur la géométrie fractale. Nous détaillons la méthode originale de compression des images naturelles proposée par Jacquin [3]. Nous continuons par une description de la méthode de Fisher [4] qui est basée sur le partitionnement quadtree. Nous terminons en détaillant une méthode de compression fractale qui adapte la triangulation de Delaunay comme modèle de partitionnement géométrique.

Dans le troisième chapitre nous introduisons les notions nécessaires à la compréhension de la théorie des ondelettes et nous présentons quelques méthodes de compression d'images par ondelettes. Puis nous décrivons notre schéma de compression sans perte qui combine le codage hybride fractal-ondelette avec l'algorithme de Huffman.

Le dernier chapitre est consacré aux discussions et interprétations des résultats obtenus.

Enfin, nous terminons par une conclusion en dégageant les points importants, et en discutant les perspectives de recherche sur la compression des images par fractales.

CHAPITRE 1

Chapitre1 : Etat de l'art sur la Compression d'images

La compression d'images est un codage de source dont le but est de réduire le nombre de bits par pixels. De nos jours cette opération s'avère indispensable pour la transmission d'information sur différents canaux de communication (web, téléphonie mobile, télévision, etc). De nombreuses méthodes ont été proposées pour la compression d'images fixes et animées. Nous présentons dans ce chapitre les principales méthodes.

1.1 Outils de la théorie de l'information [8][9][10]:

La théorie de l'information développée par Claude Shannon [10] en 1948 sert à décrire les aspects fondamentaux des systèmes de communication à l'aide de la théorie des probabilités. Elle est devenue aujourd'hui incontournable dans la conception de tout système de communication.

1.1.1 Entropie d'une Source:

Considérons une source discrète sans mémoire X : Chaque échantillon temporel $x(i)$ de la source est indépendant des autres et ne peut prendre que N-1 valeurs différentes dans l'ensemble $A = \{a_0, a_1, \ldots\ldots a_{N-1}\}$. Chaque échantillon temporel est associé à une probabilité $P(x(i) = a_i) = P(a_i)$. On définit ainsi les probabilités : $P(a_0)$, $P(a_1)$, ……., $P(a_{N-1})$.

L'entropie de X est définie comme suit :

$$H(X) = -E\left[\log_2 P(X)\right] = -\sum_{i=0}^{N-1} P(a_i)\log_2 P(a_i) \tag{1.1}$$

$H(X)$ ainsi définie s'exprime en bits, elle mesure l'information moyenne d'une source et vérifie l'inégalité suivante:

$$0 < H(X) < \log_2 N \tag{1.2}$$

De cette double inégalité on peut souligner les remarques suivantes :

- Lorsque $H(X)$ est nulle, cela se traduit par une seule probabilité non nulle d'un élément de l'alphabet. Dans ce cas la source est totalement prédictible.
- On obtient une probabilité uniforme $P_x = 1/N$ dans le cas où $H(X) = \log_2 N$, la source serait totalement non prédictible.
- L'entropie d'une source discrète est toujours positive.

On définit aussi la redondance de la source $R(X)$ par :

$$R(X) = \log_2 N - H(X) \tag{1.3}$$

La redondance pouvant être considérée comme la partie inutile d'un signal.

1.1.2 Codage d'une source :

Le codage d'une source X consiste à associer à chaque symbole $x(i)$ de la source un mot de code c_i de longueur l_i. Les mots de codes c_i apparaissent avec la même probabilité $P(a_i)$ que les symboles $x(i)$ de la source X. La valeur moyenne L_m de longueur des mots de code d'une source X, est égale à la somme des longueurs l_i pondérées par les probabilités de leur apparition $P(a_i)$:

$$L_m = \sum_{i=0}^{N-1} P(a_i).l_i \tag{1.4}$$

On constate que l'entropie $H(X)$ d'une source X représente la longueur moyenne minimale des mots de code. Le codage d'une source X dont la longueur moyenne L_m est égale à l'entropie $H(X)$ de la source est celui qui comprime le mieux l'information. Ce code est appelé code entropique (sans distorsion).
Un code doit satisfaire à priori les conditions suivantes :

- Unicité : deux messages différents ne doivent pas être codés de la même façon.
- Déchiffrablité : deux mots codes successifs doivent être distingués.

1.1.3 Fonction débit/distorsion :

Le codage entropique qui conserve la qualité initiale du signal n'est pas toujours nécessaire. En effet, le système visuel humain accepte une certaine distorsion entre le signal original et le signal décodé.

La théorie du codage nous apprend qu'il est possible de tenir compte de ces distorsions. La courbe débit/distorsion $R(D)$ donne pour une distorsion donnée, un débit minimal R et réciproquement pour un débit donné la plus faible distorsion possible D. Cette courbe est une limite théorique des performances des systèmes de codage. L'objectif de toute méthode de codage est d'atteindre une performance assez proche de cette limite.

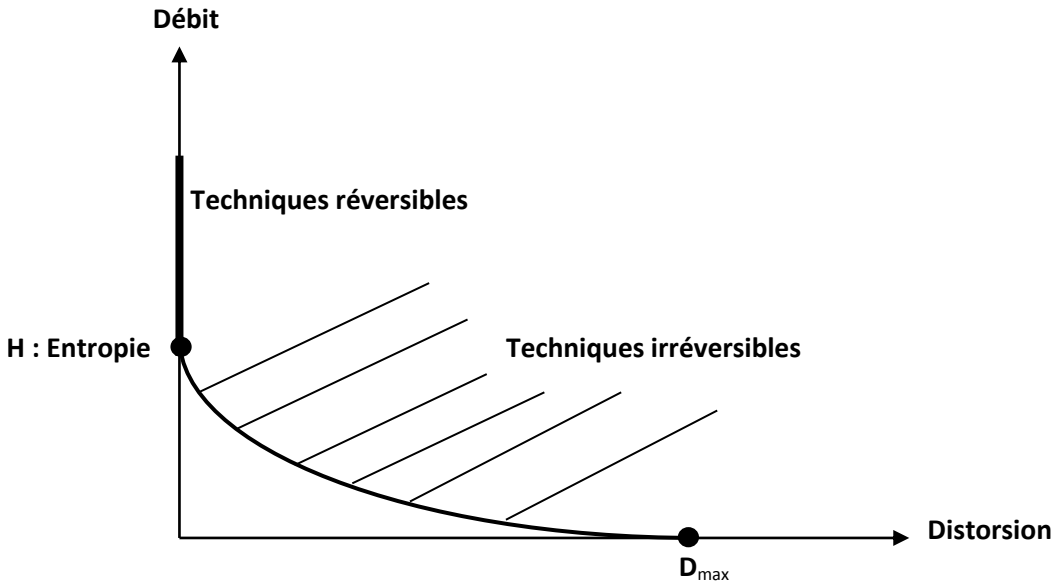


Figure 1.1: La courbe R(D) pour une source discrète à alphabet fini.

1.1.4 Codage d'images :

Lorsque l'on numérise une image sur B bits, chaque pixel ne peut prendre que 2^B valeurs distinctes de niveaux de gris comprises entre 0 et $2^B - 1$. On peut considérer une image comme étant une source discrète *X* de 2^B valeurs de niveaux de gris. Le champ de probabilités *P* associé à la source *X* représente les fréquences d'apparition de chaque niveau de gris [11].

Si les niveaux de gris sont indépendant et équiprobables $P(a_i) = 2^{-B}$, l'entropie *H* de l'image est égale à $log_2 2^B = B$ bits. Chaque pixel porte la même quantité d'information et l'image est incompressible dans ce cas. Si les niveaux de gris ne sont pas équiprobables (densité de probabilité non uniforme), l'entropie *H* est inférieure à B.

1.2 Classification des méthodes de compression d'images :

L'idée de base de la compression d'images est de réduire le nombre moyen de bits par pixels nécessaire à leur représentation, en exploitant la redondance informationnelle de l'image. Plusieurs types de redondance en termes de corrélation peuvent être considérés :

- la redondance spatiale entre pixels ou blocs voisins dans l'image.
- la redondance spectrale entre plans de couleur ou bandes spectrales.
- la redondance temporelle entre images successives dans une séquence vidéo.

Il existe une multitude de techniques de compression qui varient suivant les types d'images (naturelles, médicales, satellitaires) et les applications destinées (transmission, stockage). Elles peuvent se regrouper en deux classes [1][11][12]:

- Les méthodes sans perte d'information (sans distorsion ou réversible).
- Les méthodes avec pertes d'information (avec distorsion ou irréversible).

1.2.1 Compression sans perte:

La compression sans perte ou réversible permet de retrouver exactement les pixels de l'image originale.

1.2.1.1 Méthodes statistiques ou entropiques :

Le processus de codage entropique s'appuie sur des informations statistiques de l'image et permet de s'approcher au mieux de l'entropie. Le but de codage entropique est de coder les pixels de l'image à l'aide de mots code de longueurs variables, égales à $-\log_2 P(a_i)$ bits, de manière à ce que le nombre moyen de bits par pixel soit égale à l'entropie *H* bits (*H*<B). Il s'agit donc d'associer à chaque pixel de l'image un mot de code dont la longueur dépend de la probabilité d'apparition du niveau de gris correspondant. Pour obtenir un codage efficace, il suffit d'associer les mots de code les plus courts aux niveaux de gris les plus probables et inversement pour les niveaux possédant une faible probabilité, afin de s'approcher le plus possible à l'entropie *H* de l'image.

1.2.1.1.1 Codage de Schanon-Fano :

C. Shannon [10] et R.M. Fano [13] ont développé une méthode de codage basée sur de simples connaissances de probabilités de chaque symbole.

Le principe est le suivant [14] :

1- Classer les symboles (valeurs des niveaux de gris) par ordre de probabilités décroissantes.

2- Partager l'ensemble des symboles en deux sous-ensembles de probabilités aussi proches que possible.

3- Attribuer à chaque sous-ensemble l'état 0 ou 1.

4- Refaire les étapes 2 et 3 pour chaque partie résultante jusqu'à ce que toutes les probabilités soient isolées.

1.2.1.1.2 Codage de Huffman :

D. A. Huffman [15] a développé en 1952 un algorithme de compression capable, à partir d'une analyse statistique de données, d'associer à celles les plus souvent présentes les codes les plus courts et les données les plus rares se verront attribuer les codes les plus longs.

L'algorithme de Huffman procède de la manière suivante [16]:

1- Classer les symboles dans l'ordre décroissant des probabilités d'occurrence et les considérer comme les feuilles d'un arbre.

2- Tant qu'il reste plus d'un nœud.

- Regrouper les deux nœuds avec la plus faible probabilité pour former un nouveau nœud ayant pour probabilité la somme des deux nœuds regroupés.

- Assigner arbitrairement 0 et 1 aux deux branches conduisant au nouveau nœud formé.

3- Parcourir l'arbre depuis la racine pour trouver le code affecté à chaque symbole.

Le codage de Huffman est efficace et sa complexité est faible mais il a deux limitations:

- La difficulté d'évaluer les probabilités lorsque la taille des symboles augmente ou lorsque la loi de probabilité change.
- Le codage d'un symbole se fait sur un nombre entier de bits alors que la valeur optimale pourrait être fractionnaire.

1.2.1.1.3 Codage arithmétique :

Le codage arithmétique est un codage qui utilise un modèle statistique comme le codeur de Huffman. Mais contrairement à ce dernier, il traite la séquence de symboles tout entière, en lui associant un unique nombre décimal appartenant à l'intervalle$[0,1[$. Ces chiffres décimaux dépendent non seulement des symboles dans l'ordre où ils apparaissent, mais aussi de leur distribution [17] [18].

La procédure de codage arithmétique est la suivante :

1. Calculer la probabilité associée à chaque symbole dans la chaîne à coder.
2. Associer à chaque symbole un sous intervalle proportionnel à sa probabilité, dans l'intervalle $[0,1[$ (l'ordre de rangement des intervalles sera mémorisé car il est nécessaire au décodeur)
3. Initialiser la limite inférieure de l'intervalle de travail à la valeur 0 et la limite supérieure à la valeur 1.
4. Tant qu'il reste un symbole dans la chaîne à coder:
   - largeur = limite supérieure - limite inférieure.
   - limite inférieure = limite inférieure + largeur × limite basse du sous intervalle du symbole.
   - limite supérieure = limite inférieure + largeur × limite haute du sous intervalle du symbole.
5. La limite inférieure code la chaîne de manière unique.

1.2.1.2 Méthodes à base de dictionnaire :

Les méthodes de compression par dictionnaire ont été proposées pour prendre en compte les dépendances d'un symbole à l'autre de la source. Ces algorithmes permettent de coder une source quelconque sans connaître à priori ses statistiques [19].

1.2.1.2.1 Algorithme de Lempel Ziv LZ77

L'algorithme LZ77 proposé par Abraham Lempel et Jacob Ziv en 1977 [20] est à la base de tous les algorithmes à dictionnaires utilisés actuellement. Le dictionnaire est un tableau, de taille variable, contenant les séquences de symbole repérées par leur adresse. Le principe est basé sur l'utilisation d'une fenêtre de texte, divisée en deux parties. La première partie de grande taille contient le texte déjà codé. La deuxième, de taille plus réduite, est un tampon de lecture. Si le début du texte dans le tampon est déjà dans la première partie, le code correspondant est composé de la position du bloc dans la première partie, la longueur du bloc ainsi que le symbole suivant. Le codeur émet le code puis fait entrer "longueur+1" symboles supplémentaires dans le tampon. Le processus se répète ainsi jusqu'à la fin du fichier à compresser. De plus, s'il n'y a pas de correspondance, les symboles doivent être transmis individuellement avec une longueur de séquence égale à zéro. Cette méthode est souffre du temps de calcul important pour coder un fichier. Ceci est dû aux nombreuses comparaisons des blocs entre le tampon de pré-lecture et ceux de la première partie de la fenêtre.

1.2.1.2.3 Algorithme LZ78 :

Ziv et Lempel [21] ont proposé en 1978 un algorithme de codage connu sous le nom LZ78, où le dictionnaire est construit dynamiquement à partir de tous les symboles précédents rencontrés et non par une fenêtre glissante. Chaque signe transmis par le codeur comprend un index dans le dictionnaire, ainsi que le caractère suivant à coder. La longueur de la chaîne n'a plus à être transmise puisqu'elle est conservée dans le dictionnaire. La concaténation de la chaîne répétée avec le caractère suivant est également placée dans le dictionnaire.

Le procédé de l'algorithme LZ78 est le suivant :

1. chaîne courante = vide
2. dictionnaire = chaîne courante
3. répéter
   - lire le caractère et l'ajouter à la chaîne courante
   - Si chaîne courante est dans l'arbre constituant le dictionnaire

   Alors continuer

   Sinon
     - ajouter la chaîne courante dans le dictionnaire et étiqueter la chaîne
     - émettre l'étiquette de la chaîne courante sans le dernier caractère
     - émettre le dernier caractère
     - chaîne courante = vide
4. jusqu'à ce qu'il n'y ait plus de caractère à coder

L’algorithme LZ78 permet de surmonter les limites de LZ77 et améliore considérablement les performances du codage. Cependant, la taille du dictionnaire utilisé par le LZ78 est limitée et la compression de fichier trop volumineux ou contenant de nombreuses répétitions n'est pas très performante.

1.2.1.2.3 Algorithme LZW :

Terry Welsh [22] a proposé en 1984 d’utiliser un dictionnaire qui comprend au départ le code ASCII, repéré de 0 à 255.

Le procédé de l’algorithme est le suivant :

1. Initialiser le dictionnaire en y plaçant les codes des caractères ASCII,
2. Lire le caractère à transmettre :
    - S’il existe déjà dans la table, on passe au caractère suivant,
    - Si le groupe des deux existe également, on regarde le suivant, etc.
3. Lorsqu’un nouveau groupe est découvert, on le définit en l’insérant dans le dictionnaire. Dans un premier temps, on transmet les codes des morceaux qui le composent. La prochaine fois qu’on le rencontrera, on ne transmettra que son code propre.
4. On procède de la sorte jusqu’à la fin de la transmission.
5. Lorsque le dictionnaire est plein, soit on procède à son extension, soit on se borne à utiliser les codes déjà existants.

1.2.1.3 Méthodes de compression par plages (Run Length Encoding) :

Une plage est une succession de symboles ayant la même valeur. La méthode de compression par plages, connue sous le nom anglais "Run Length Encoding" (RLE), code des séquences de pixels identiques sur une ligne d’image par leurs longueurs et leurs valeurs. Le procédé "Run Length" ne relève pas d’une théorie mathématique très complexe, il s’agit simplement de remplacer des éléments significatifs successifs identiques par un seul d’entre eux et du nombre de répétition [23].

RLE semble bien adapté à la compression d’image grâce à la corrélation entre pixels voisins dans l’image. Soit par exemple une image en 256 niveaux de gris commençant comme ceci : 110, 110, 110, 113, 25, 25, 25, 25,25, …

RLE donne la représentation (nombre d’occurrence, symbole) comme suit : 3, 110, 113, 5, 25

Cette méthode est efficace seulement pour de nombreuses et longues plages constantes. C’est un codage rapide et simple à mettre en œuvre.

1.2.1.4 Méthodes différentielles et prédictives sans pertes [24][25]:

Ces méthodes exploitent la redondance spatiale entre un pixel et ses voisins. Dans les méthodes différentielles on transmet non plus la valeur du pixel, mais celle de la différence avec le pixel précédent, qui est statistiquement inférieure et peut donc être codée sur un nombre de bits plus réduit. On peut alors appliquer les méthodes de Huffman ou LZ aux données obtenues. Ce codage, largement utilisé par le passé, suppose néanmoins des traitements assez lourds et se révèle de moins en moins performant lorsque le nombre de couleurs augmente [24].

Les algorithmes qui utilisent le codage par prédiction [25] permettent de prédire la valeur d'un pixel en fonction de la valeur des pixels voisins et de ne coder que l'erreur de prédiction. Le voisinage peut être défini selon sa connexité (4-connexité ou 8-connexité) ou selon l'ordre du parcours choisi pour accéder aux pixels voisins. L'une des techniques de prédiction la plus simple est la DPCM (Differential Pulse Code Modulation) [26]. Cette technique effectue une prédiction à base d'une combinaison linéaire des valeurs des pixels voisins. Une version adaptative, ADPCM (Adaptive Differential Pulse Code Modulation), qui utilise différentes formes de prédiction et de voisinage selon le contexte et le contenu de l'image a été présentée par Kyung et al [27]. Les méthodes prédictives permettent une mise en œuvre facile et elles sont efficaces pour les images dont les spatiales sont petites.

1.2.1.5 Le codage par plans de bits:

Une image codée sur 2^B niveaux de gris (B bits par pixel) peut être considérée comme une superposition de B plans chacun de hauteur 1 bit. Chaque plan de bits (image binaire) peut être codé séparément en utilisant la méthode RLE. Le code de Gray [8] est dans ce cas utilisé pour augmenter la cohérence au sein des différents plans de bits. Les plans composés des bits de poids fort contiennent la majeure partie de l'information visuelle de l'image.

1.2.2 Compression avec pertes :

Ce sont des méthodes irréversibles qui permettent de retrouver une approximation de l'image numérique. Ces techniques sont les plus implémentées dans les systèmes pratiques car elles permettent une réduction considérable de la quantité d'information.

1.2.2.1 Les étapes de compression avec pertes :

La compression se fait en trois grandes étapes : la décorrélation, la quantification et le codage des symboles issus de la quantification des données [12].

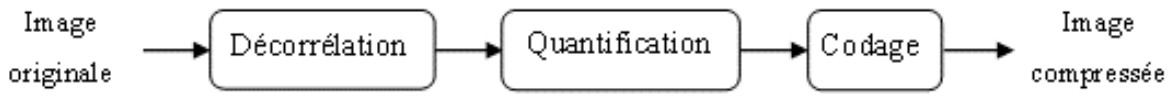


Figure 1.2 Etapes de la compression avec pertes.

1.2.2.1.1 La Décorrélation :

Il existe deux principales méthodes de décorrélations des données :

Décorrélation spatiale:

Les algorithmes qui exploitent la redondance spatiale ont comme objectif de rechercher un modèle de représentation le plus adéquat de l'information à coder. La méthode prédictive est une méthode décorrélative, qui consiste à représente chaque pixel en fonction de ses voisins. L'idée est de coder l'erreur de prédiction au-dessus d'un seuil. Ce seuil peut être défini par rapport à la qualité de l'image ou le niveau de compression espéré [28].

Décorrélation par transformation :

Les méthodes qui utilisent cette technique utilisent des transformations pour produire une décorrélation des redondances spatiales [29][30]. Les pixels passent d'un espace où ils sont fortement corrélés dans un autre espace de représentation d'énergie fortement décorrélée. Cette classe d'algorithme de compression est déclinée suivant plusieurs types de transformations fréquentielles, dont les principales sont la transformation de Karhunen-Loeve (KLT) [31], la transformation de Fourrier discrète (DFT) [32], la transformation en cosinus discrète (DCT) [33] et la transformation en ondelettes discrète (DWT) [34].

Nous évoquerons dans ce qui suit la transformation de Karhunen-Loeve et la Transformation en cosinus discrète, la transformation de Fourrier discrète et la transformation en ondelettes discrète sera traitée dans le chapitre 3.

Transformation de Karhunen-Loeve (KLT) :

La transformation de Karhunen-Loeve (KLT) est optimale au sens où tous les coefficients obtenus sont décorrélés et que la quasi-totalité de l'énergie est conservée par un minimum de coefficients, mais les éléments de la transformation dépendent de l'image originale. Par ailleurs, il n'existe pas d'algorithme rapide pour le calcul de la transformation de Karhunen-Loeve. Toutes ces raisons font que cette transformation soit très peu utilisée dans la pratique malgré sa supériorité théorique [35].

Transformation en Cosinus Discrète (DCT) :

La DCT est l'une des transformations les plus utilisée pour la compression d'images, C'est une transformation mathématique orthogonale qui transforme un ensemble de données d'un domaine spatial en un spectre de fréquence et inversement (IDCT) [36].

La DCT est effectuée sur une matrice carrée $N \times N$ de valeurs de pixels $f(x,y)$ et donne une matrice carrée $N \times N$ de coefficients de fréquence $F(u,v)$:

$$F(u,v) = \frac{1}{\sqrt{2N}} C(u)C(v) \sum_{y=0}^{N-1} \sum_{x=0}^{N-1} f(x,y) \cos\left(\frac{(2x+1)u\pi}{2N}\right) \cos\left(\frac{(2y+1)v\pi}{2N}\right) \quad (1.5)$$

avec $C(u)$ et $C(v)$ les facteurs d'orthogonalité de la transformée:

$$C(u) = \begin{cases} \frac{1}{\sqrt{2}} & \text{si } u = 0 \\ 1 & \text{si } u \neq 0 \end{cases}$$

où les variables x et y représentent les indices de ligne et de colonne dans la matrice des données. Les u et v représentent les indices de ligne et de colonne des coefficients transformés.

Le coefficient DCT, $F(0,0)$ est appelé composante continue ou composante DC (Direct Current). Il est proportionnel à la moyenne des intensités des pixels sur le bloc considéré. Les autres coefficients sont appelés AC (Alternating Current). Sur les coefficients AC, les énergies sont groupées en basse (zone homogène de l'image originale), moyenne et haute (zone texturée ou zone de contour) fréquences.

La transformée inverse qui permet de revenir au domaine spatial à partir des coefficients DCT s'écrit :

$$f(x,y) = \frac{1}{\sqrt{2N}} \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} C(u)C(v)F(u,v) \cos\left(\frac{(2x+1)u\pi}{2N}\right) \cos\left(\frac{(2y+1)v\pi}{2N}\right) \quad (1.6)$$

L'efficacité de la DCT en termes de compactage d'énergie est proche de celle de la KLT, pour les images ayant une forte corrélation inter-pixels [37]. Il existe de nombreux algorithmes rapides de calculs de la DCT, qui diminuent le nombre d'opérations nécessaires, souvent en passant par la FFT [38]. La DCT peut être calculée sur des blocs de l'image (transformation par bloc) ou sur l'image tout entière (transformation Full-Frame).

1.2.2.1.2 La Quantification :

Dans le schéma de compression avec pertes, l'étape de quantification dégrade de manière irréversible l'image, mais permet une réduction importante du débit binaire. On distingue en général deux types de quantification : la quantification scalaire et la quantification vectorielle.

Quantification scalaire :

Un quantificateur scalaire est un opérateur qui associe à une variable continue x d'un ensemble U ($\mathbb{R}$ par exemple) une variable discrète $q(x)$ pouvant prendre un nombre plus faible, et fini de valeurs y_i d'un ensemble $Y = \{y_i, i = 1, \cdots L\}$ correspondant au dictionnaire. L'application est ainsi définie de la manière suivante :

$$U \rightarrow Y = \{y_1, y_2, \cdots y_L\}$$

$$x \rightarrow y_i = q(x)$$

Les valeurs y_i sont appelées valeurs de quantification.

Les performances d'un quantificateur se mesurent en termes de débit binaire et de distorsion [39]. Le nombre L d'éléments du dictionnaire détermine le débit binaire maximal R_{max} :

$$R_{\max} = \lceil \log_2(L) \rceil \text{ bits/echantillon}$$

Quand une valeur d'entrée x est quantifiée par $q(x) = y_i$, on introduit une erreur de quantification $e = x - q(x)$. Généralement on évalue la distorsion moyenne comme l'erreur quadratique moyenne :

$$D = E\left[(X - q(X))^2\right] = \sum_{i=1}^{L} \int_{-\infty}^{+\infty} (x - y_i)^2 f_X(x) dx \tag{1.7}$$

Où $f_X(x)$ est la distribution de probabilité de la source X.

Il existe plusieurs stratégies de quantification scalaire que nous allons brièvement présenter :

Quantification uniforme :

La quantification uniforme est la plus répandue et la plus simple. Il s'agit d'un quantificateur régulier pour lequel les niveaux de décision x_i sont également espacés et les représentants y_i se situent au milieu des intervalles $]x_{i-1}, x_i]$, en d'autres termes :

$$\begin{aligned} x_i - x_{i-1} &= \Delta, \quad \forall i = 1, \cdots, L \\ y_i &= \frac{x_{i-1} + x_i}{2}, \quad \forall i = 1 \cdots, L \end{aligned} \tag{1.8}$$

Lorsque une source X à valeurs uniformément réparties est quantifiée par une quantification scalaire uniforme, la distorsion introduite au sens de l'erreur quadratique moyenne est donnée par :

$$D = E\left[(X - q(X))^2\right] = \frac{\Delta^2}{12} \tag{1.9}$$

Lorsque la source est uniformément distribuée, le quantificateur uniforme est optimal et permet d'obtenir le minimum de l'erreur quadratique moyenne. En revanche, lorsque la distribution du signal n'est pas uniforme, le quantificateur optimal n'est plus uniforme.

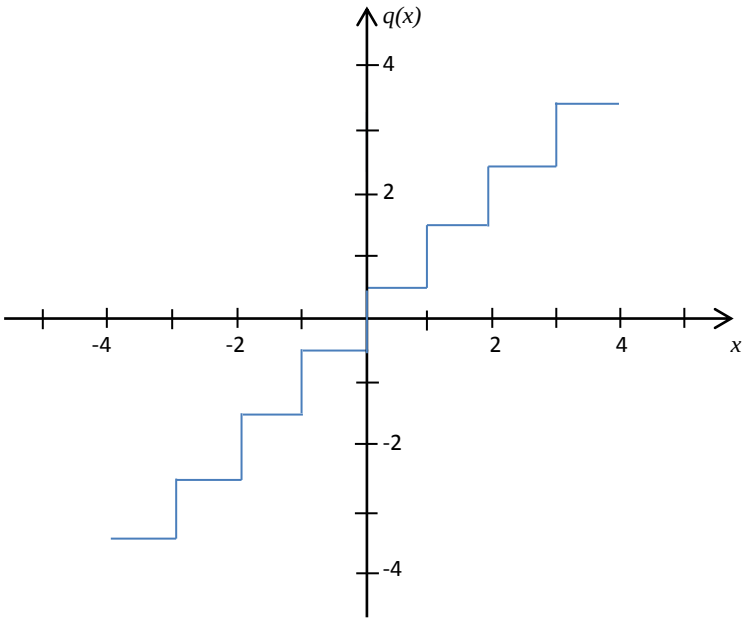


Figure 1.3 : Quantification scalaire uniforme.

Quantificateur scalaire optimal de Lloyd-Max :

Le quantificateur de Lloyd-Max minimise l'erreur quadratique moyenne entre l'entrée et la sortie, pour un nombre de niveaux de quantification donné L. La minimisation de l'erreur quadratique de quantification conduit aux valeurs optimales suivantes [40][41][42]:

$$\begin{cases} x_i = \dfrac{y_i + y_{i-1}}{2} \\ y_i = \dfrac{\int_{x_i}^{x_{i+1}} x f_X(x)dx}{\int_{x_i}^{x_{i+1}} f_X(x)dx} \end{cases} \tag{1.10}$$

Quantification vectorielle (QV):

Le quantificateur vectoriel (QV), développé par Gersho et Gray [43][44][45], peut être vu comme une application Q associant à chaque vecteur d'entrée $x = (x_i,\ i = 1...K)$ un vecteur de sortie $y = (y_i,\ i = 1...K) = Q(x)$, choisi parmi un dictionnaire $Y = (\ y_j,\ \ j = 1...N)$ de taille finie N. En pratique, dans plusieurs systèmes de communication, seul l'indice j de la séquence choisie y_j est transmis au décodeur qui choisit alors la séquence correspondante à cette valeur de l'indice. Le dictionnaire qui influe considérablement sur les performances techniques d'un QV est caractérisé par sa taille N et sa dimension K, ces paramètres permettent de déterminer le débit binaire $R = \log_2 N/K$ bits par échantillon. La figure 1.4 schématise un exemple

d'organisation structurelle d'un quantificateur vectoriel souvent utilisé dans les systèmes de communication numérique [46].

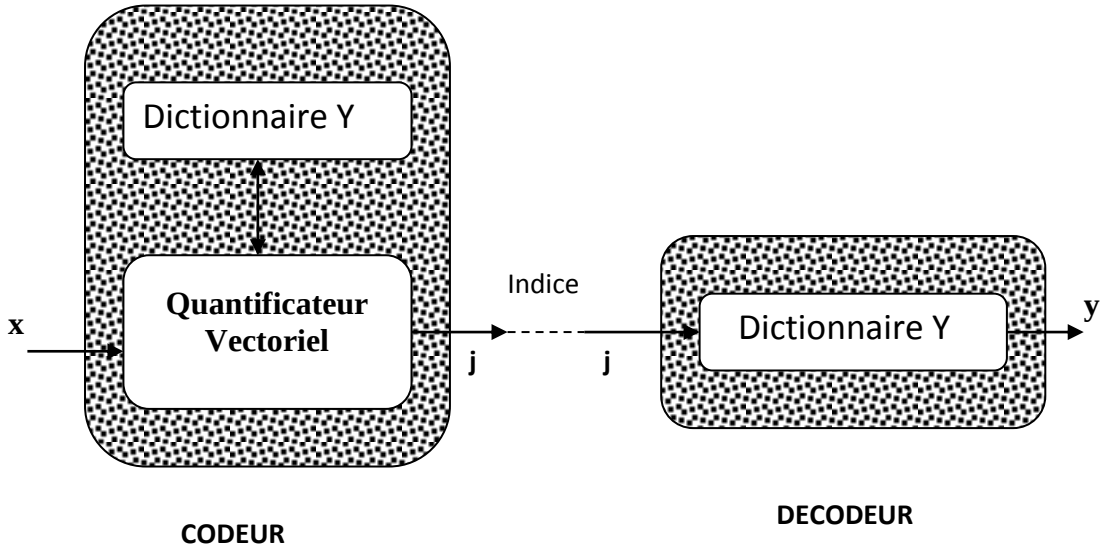


Figure 1.4 : Schéma d'un quantificateur vectoriel.

1.2.2.1.3 Le Codage :

Un codage sans perte est effectué sur les coefficients quantifiés afin d'exploiter les redondances présentes à la sortie du quantificateur.

1.2.2.2 Autres méthodes de compression avec pertes :

1.2.2.2.1 Codage sous bandes :

Le codage sous-bandes utilise également une représentation assimilable à une représentation par transformation de l'image. Il consiste à décomposer le signal ou l'image en différentes bandes de fréquences spatiales et à coder chacune d'entre elles avec des stratégies adaptées à leur contenu énergétique. Dans les schémas classiques, la bande spectrale du signal original est d'abord divisée en deux parties à l'aide de deux filtres numériques (passe bas et passe haut). La procédure est ensuite répétée dans chacune des sous bandes fréquentielles. Tout comme les méthodes par transformation, on tend à privilégier les basses fréquences qui sont riches en énergie, et à coder plus grossièrement les hautes fréquences en prenant en compte la sensibilité de l'œil humain. Les méthodes de codage couramment employées sur les sous-bandes sont la méthode prédictive pour la sous-bande basse fréquence, et la quantification vectorielle pour les autres sous-bandes [47].

Un avantage du codage sous-bandes réside dans la possibilité de transmission progressive, qui permet de reconstruire l'image comprimée en basse résolution (la sous bande basse fréquence) et d'ajouter progressivement les sous-images de détail si l'utilisateur désire plus de finesse.

1.2.2.2.2 Méthodes par contour :

La méthode par contour consiste à séparer l'image en deux images complémentaires: une image de contours (contenant les hautes fréquences) et une image de fond (contenant les basses fréquences). Une stratégie de codage appropriée est appliquée sur chacune de ces deux images [48].

1.2.2.2.3 Méthodes texturales :

La méthode texturale consiste à repérer dans l'image des zones ayant des caractéristiques voisines, c'est à dire des textures semblables. On code le type de texture et ses paramètres caractéristiques. On reconstruit l'image en régénérant synthétiquement les textures [48].

1.2.3 La norme de compression JPEG :

La norme JPEG (Joint Photographic Experts Group) est conçue par le groupe ISO (International Standards Organisation) et le groupe CEI (Commission Electronic International) [49]. Elle est destinée à la compression des images fixes en couleurs et à niveaux de gris.

Différentes contraintes ont été imposées au standard :

- L'algorithme JPEG doit être implémentable sur une grande variété de type de CPU et sur des cartes spécialisées (appareil photo numérique, téléphone portable).
- La norme JPEG doit compresser n'importe quels types d'images réelles (images photographiques, médicales,…) de tailles différentes et multi-composantes.
- Il possède quatre modes de fonctionnement : le mode séquentiel (avec pertes), le mode sans perte, le mode progressif qui permet de parcourir plusieurs fois l'image et ajouter des détails au fil des parcours, et le mode hiérarchique qui traite l'image comme une série d'images à plusieurs niveaux de résolution.

JPEG avec pertes :

Entre les 4 modes de compression de la norme JPEG, le séquentiel (avec pertes) est le mode principal le plus répandu et le plus performant du point de vue de l'efficacité de codage. La figure 1.5 représente les étapes de ce mode.

L'image originale est soumise à un changement d'espace couleur YCbCr (YUV). Les informations de chrominance (les plans Cb et Cr) sont sous échantillonnées. Après cette opération, les deux plans auront deux fois moins de lignes et de colonnes. Le sous échantillonnage est fondé sur le principe que le système visuel humain ne peut discerner des différences de chrominance au sein d'un carré de 2×2 pixels. Après le sous échantillonnage, chaque plan Y, Cb, Cr est décomposé séquentiellement en blocs de 8×8 pixels, pour subir le même traitement selon le schéma fonctionnel présenté sur la figure 1.5.

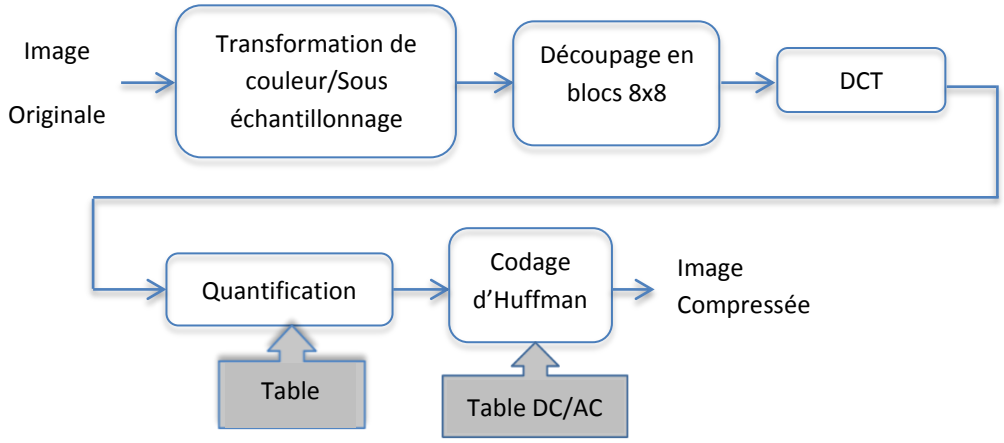


Figure 1.5 : Schéma typique d'un codeur JPEG avec pertes.

Une transformée en cosinus discrète bidimensionnelle (DCT) est réalisée sur chaque bloc. Une fois la DCT calculée sur un bloc, nous obtenons une matrice carrée des valeurs, pour chacune des fréquences, qui seront quantifiées. L'étape de quantification est réalisée à l'aide d'une matrice de quantification Q de 8×8 éléments. Chaque coefficient DCT est divisé par la valeur correspondante dans Q et le résultat est arrondi à l'entier le plus proche. Quelques tables standards, pour la quantification, ont été générées grâce à une série de caractéristiques psychovisuelles. La sensibilité du système visuel humain est plus faible à des hautes fréquences et plus sensible aux basses fréquences. Les valeurs prennent donc en compte cette caractéristique et introduisent majoritairement de la distorsion dans les hautes fréquences.

La dernière étape de la compression JPEG est le codage de la matrice DCT quantifiée. Après la quantification un grand nombre de coefficients sont nuls ou très proche de zéro. Le coefficient DC (Direct Current) est codé séparément par rapport aux AC (Alternative Current). Les coefficients AC sont parcourus en zigzag, et sont codés par des couples (HEAD), (AMPLITUDE). L'entête HEAD contient des contrôleurs qui seront utilisés pour accéder aux tables d'Huffman. Le paramètre AMPLITUDE est un entier signé correspondant à l'amplitude du coefficient AC non nul. La structure HEAD varie en fonction du type de coefficient. Pour les AC, elle est composée de (RUNLENGTH , SIZE), alors que pour les DC elle est composée seulement de la taille SIZE. Puisque les blocs contigus sont très corrélés, on peut coder le coefficient continu DC du bloc courant en codant simplement la différence avec le coefficient continu du bloc précédent. Cela produira en général un très petit nombre.

JPEG sans perte :

La norme JPEG sans perte de 1993 (JPEG lossless) [50], propose un algorithme de prédiction à la présence d'un contour horizontal ou vertical. Huit cas de figures sont répertoriés qui conduisent à huit décisions différentes. Le codage du résidu se fait par Huffman. Des taux de compression pouvant aller jusqu'à 3 sont obtenus en imagerie

médicale, mais ils sont souvent plus faibles en imagerie naturelle où JPEG sans perte n'a jamais eu de grand succès.

Un second codeur sans perte pour JPEG a ensuite été proposé [51]: le JPEG-LS est basé sur la méthode LOCO-I (Low Complexity Lossless Compression method) [52]. Dans JPEG-LS le procédé de compression est composé de trois parties : **la prédiction** de la valeur du pixel qui est faite par rapport aux pixels voisins en utilisant l'approche de prédiction MED (Median Edge Detection), **la détermination d'un contexte** : Ce contexte représente l'environnement du pixel à coder à ses voisins. L'idée est de prendre le meilleur environnement qui affine la prédiction avant le codage et de réduire le nombre de paramètres de l'erreur de prédiction, **le codage** de l'erreur de prédiction dont l'approche est de réinsérer l'erreur de prédiction dans le système puis de la comparer à d'autres mesures d'erreur. Il a des performances supérieures à JPEG lossless et peut atteindre 3 sur des images naturelles. Il a fait l'objet d'une normalisation ISO-14495-1 [53].

1.3 Compression des images animées:

Dans une séquence animée, on exploite deux sortes de redondances : la redondance spatiale présente dans les images, et la redondance temporelle (le fait que des images successives aient des parties communes) [54] [55]. Lorsqu'on parle de compression vidéo, la première idée est de coder chaque image individuellement par compression JPEG. Mais puisque dans les séquences vidéos, de nombreuses scènes sont fixes ou bien changent très peu, il suffit donc de décrire seulement le changement d'une image à l'autre. Le groupe MPEG (Moving Pictures Experts Group) a été établit en 1988 dans le but de développer des standards internationaux de codage d'image animées.

Il existe plusieurs standards MPEG :

MPEG-1 : développé en 1988 est un standard pour la compression des données vidéos et des canaux audio associés. Il permet le stockage de vidéos à un débit de 1.5 Mbps. C'est la norme de stockage de vidéos sur CD-ROM au format VCD (Vidéo CD).

MPEG-2 : un standard dédié originalement à la télévision numérique (HDTV) offrant une qualité élevée pour un débit pouvant aller jusqu'à 40 Mbps. Il s'agit du format utilisé par les DVD vidéo.

MPEG-4 : un standard destiné à coder les données multimédia sous formes d'objets numériques, afin d'obtenir une plus grande interactivité, ce qui rend son usage particulièrement adapté au WEB et aux périphériques mobiles.

MPEG-7, MPEG-21 et MPEG-x (x prenant les valeurs de A à V) sont des développements qui visent les applications multimédias des séquences vidéos et définissent des fonctionnalités spécifiques à l'archivage, la recherche dans les bases de données, la production, l'interaction, la diffusion et la protection des contenus.

Le MPEG encode une vidéo grâce à plusieurs techniques :

Le codage intra-image (Frames I) : Les images sont codées séparément en utilisant le codage JPEG, sans faire référence aux images précédentes. De telles images sont nécessaires dans une vidéo MPEG car ce sont elles qui assurent la cohésion de la séquence (puisque les autres sont décrites par rapport aux images qui les entourent), elles sont utiles notamment pour les flux vidéo qui peuvent être pris en cours de route (télévision), et sont indispensables en cas d'erreur dans la réception. Il y en a donc une ou deux par seconde dans une vidéo MPEG.

Le codage inter-images : Il se fait selon 2 modes :

Le mode prédictif (Frames P) : Les images sont décrites par différence avec les images précédentes. L'encodeur cherche les différences de l'image par rapport à la précédente et définit des blocs, appelés macroblocs (16×16 pixels) qui superposent à l'image précédente. L'algorithme compare les deux images bloc par bloc et à partir d'un certain seuil de différence, il considère le bloc de l'image précédente différent de celui de l'image en cours et lui applique une compression JPEG. C'est la recherche des macroblocs qui détermine la vitesse de l'encodage. Par rapport aux frames I, les frames P demandent d'avoir toujours en mémoire l'image précédente.

Le mode interpolé bi-directionnellement (Frames B) : ce mode permet d'interpoler le flot d'images en utilisant à la fois une image antérieure I et une image postérieure P pour reconstruire des images intermédiaires notées B. Ce mode de fonctionnement présente néanmoins l'inconvénient d'introduire un délai dans la reconstruction de l'image et oblige à garder en mémoire trois images (la précédente, l'actuelle et la suivante).
Afin d'optimiser le codage MPEG, les séquences d'images sont dans la pratique codées suivant une suite d'images I, P et B. La première image de la séquence sera une image intra, les suivantes étant des images inter (P et B). La succession des images intra et inter dépendra de l'algorithme utilisé. En raison des prédictions, des pertes sont souvent générées. Il est donc nécessaire de régulièrement encoder une frame intra (I). Ainsi, comme illustré sur la figure 1.6 en sortie du codeur, la frame I est codée, suivie de la frame P. On trouve ensuite trois images B codées à partir des frames I et P. Dans le flux en sortie du décodeur, on remarque que l'ordre est modifié.

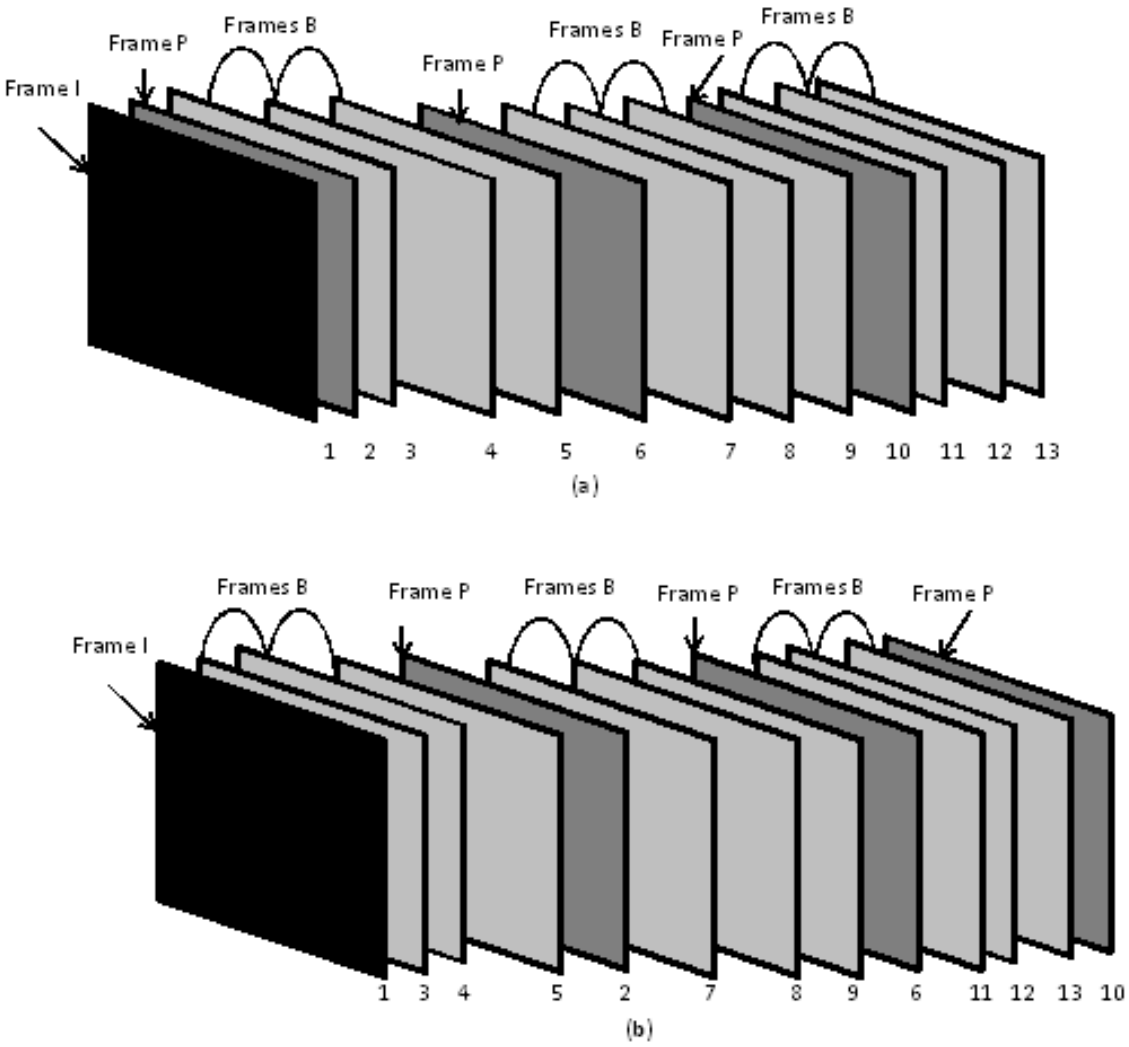


Figure 1.6 : Codage des images animées (a) Codeur, (b) Décodeur.

1.4 Conclusion :

Nous avons introduit dans ce chapitre différentes méthodes de codage et de compression réversible et irréversible des images numériques. D'une manière générale toutes les approches énumérées précédemment utilisent les corrélations entre pixels voisins dans l'image. Ces corrélations sont le fondement de la compression car elles sont liées à la notion de redondance. Pour obtenir des taux de compression importants des méthodes irréversibles sont souvent utilisées mais elles induisent une distorsion ou des erreurs dans la sortie décodée.

Le codage étant plus ou moins complexe, il nécessite un certain temps d'exécution avant qu'il soit complété. Il faut donc opter pour un faible débit, une bonne qualité de décodage et un temps de codage minimal. La compression de futur semble être la compression par fractales et ondelettes. Le chapitre suivant est consacré à la compression d'images selon l'approche fractale.

CHAPITRE 2

Chapitre 2 : La compression fractale d'images

Nous rappelons dans ce chapitre les principaux aspects de la géométrie fractale et la théorie des systèmes de fonctions itérées IFS permettant le codage et la synthèse d'images fractales. Nous donnons par la suite un état de l'art de la compression d'images par fractales, en précisant les critères liés à la conception du codeur fractal. Puis nous étudions les principales méthodes de compression fractale fondées sur trois modèles de partitionnement géométriques (carré, quadtree et triangulaire).

2.1 Généralités sur les fractales :

Mandelbrot [56], l'initiateur du concept de fractale, définit une fractale comme une figure géométrique ou un objet naturel ayant les caractéristiques suivantes :

- Ses parties ont la même forme ou la même structure que le tout.
- Sa forme est, soit extrêmement irrégulière, soit extrêmement interrompue ou fragmentée quelle que soit l'échelle d'examen.
- Il contient des éléments distincts dont les échelles sont variées et couvrent une très large gamme.

2.1.1 Classification des fractales :

Il existe deux grandes classes d'objets fractales : Les fractales géométriques et les fractales stochastiques ou naturelles.

2.1.1.1 Les fractales géométriques :

Ces fractales sont construites d'une façon déterministe sans faire intervenir le hasard. Partant d'une figure de base, qui est l'ordre zéro de la fractale (Initiateur) et d'une certaine figure que nous appellerons la graine de la fractale (Générateur). On ajoute ou l'on remplace certaines parties de l'initiateur par une image semblable au générateur. On obtient alors l'ordre un de la fractale. Il suffit de recommencer avec l'ordre un pour obtenir l'ordre deux et ainsi de suite. La figure obtenue à l'ordre infini, si elle existe, est une fractale.

Selon la manière dont les fractales géométriques sont construites, nous distinguons trois types [56]:

Les fractales particulaires :

Les fractales particulaires sont constituées entièrement de points, elles sont obtenues à partir de suites récurrentes de forme : $x_{n+1} = F(x_n, y_n)$ et $y_{n+1} = G(x_n, y_n)$.

Si par exemple, on pose :

$$\begin{cases} (x_0, y_0) = (0,0) \\ x_{n+1} = \sigma\left(r_n + (x_n - \alpha)/2\right)^{1/2} \\ y_{n+1} = \pm\sigma\left(r_n + (x_n - \alpha)/2\right)^{1/2} \end{cases}$$

avec : $r_n = \left((x_n - \alpha)^2 + (y_n - \beta)^2\right)^{1/2}$; α et β sont des constantes fixes.

± désigne le signe de $(y_n - \beta)$

σ est aléatoirement égale à +1 ou –1.

Au bout de quelques itérations, la suite ainsi obtenue donne des points situés à la frontière d'un certain ensemble appelé ensemble de JULIA. Cette frontière est appelée la courbe de JULIA [57].

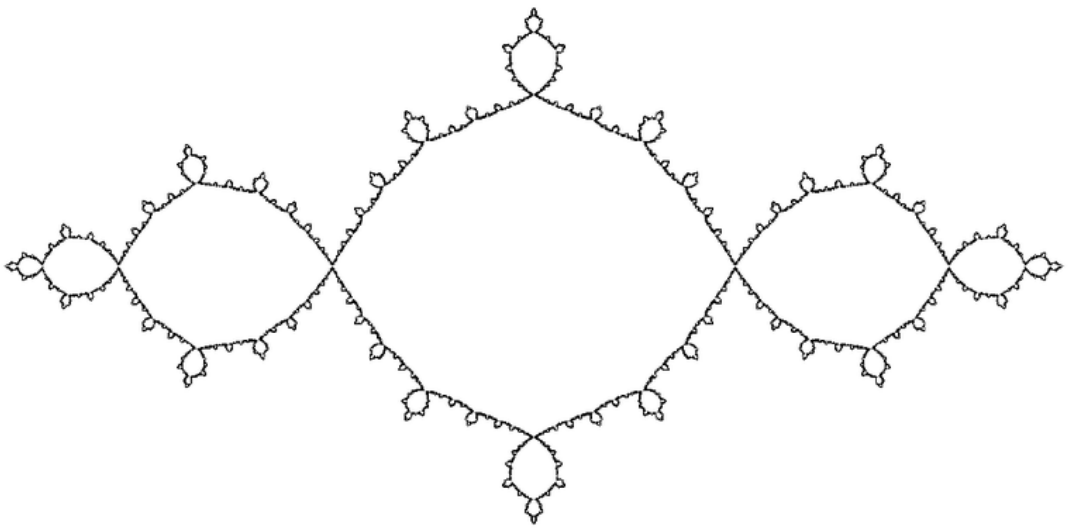


Figure 2.1 : Courbe de JULIA avec α = -1 et β = 0 [57].

Les fractales en cascades :

Ces fractales sont déduites d'une forme géométrique simple, dessinée un nombre de plus en plus important de fois entre deux ordres successifs, mais de plus en plus petite. Donnons un exemple basé sur des triangles équilatéraux. Partant d'un tel triangle, on dessine centré sur chacune de ses pointes, un triangle semblable tourné vers le centre du triangle initial et deux

fois plus petit. Puis on recommence pour chacun des trois triangles ainsi obtenus. A l'ordre *n*, le nombre de triangles tracés au total est : $T_n = 1+3+3^2+\ldots\ldots 3^n$.

Les fractales linéaires :

Elles sont composées à tout ordre fini de segments de droite. Chaque ordre se déduisant du précédent en remplaçant tout ou partie de ces segments de droite, par une ligne polygonale semblable à une ligne fixée au départ. La courbe de Von KOCH [58] constitue un bon exemple de ces fractales.

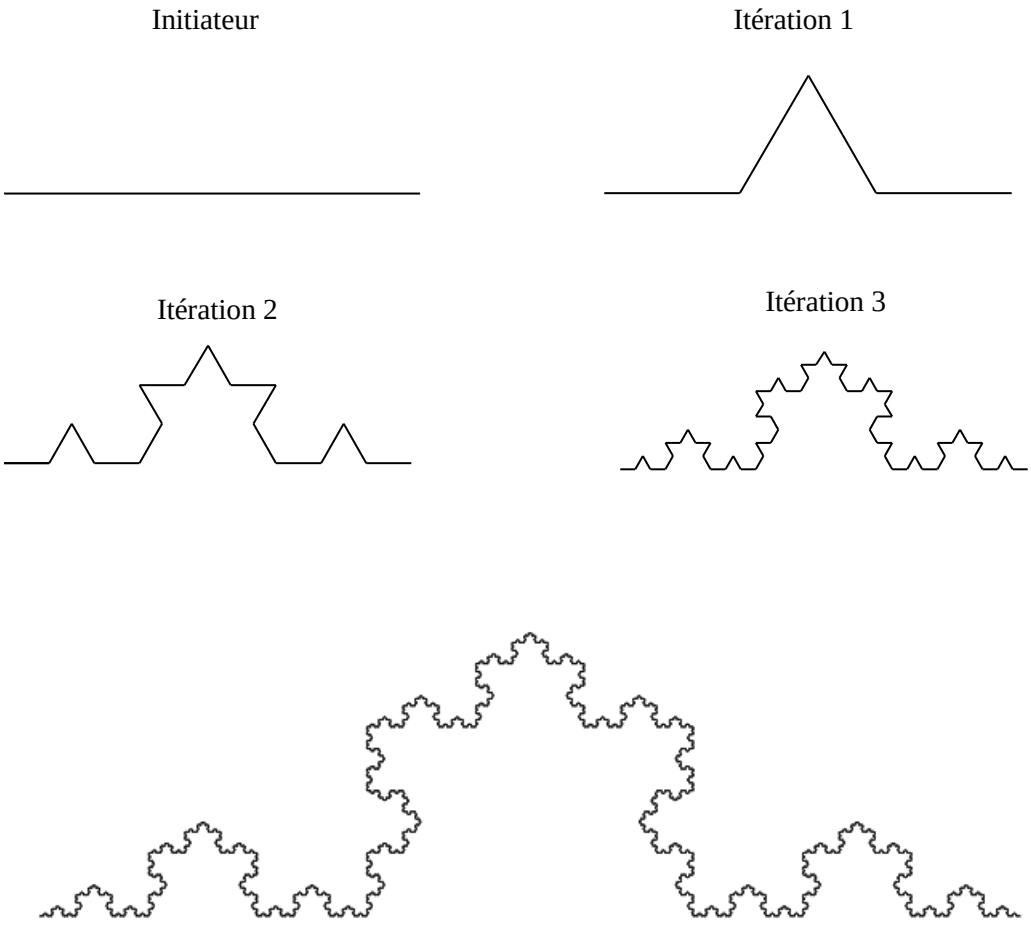


Figure 2.2 : Courbe de Von Koch [58].

Une autre variation possible dans les structures des fractales géométriques réside dans la présence simultanée de plusieurs échelles de dilatation. La figure 2.3 est un exemple d'une telle structure obtenue par itération déterministe contenant les facteurs 1/ 4 et 1/ 2. Sa géométrie est évidemment fractale et sa structure est en réalité plus complexe dans la distribution de son support que les fractales décrites plus haut : elle est en fait multi-fractale.

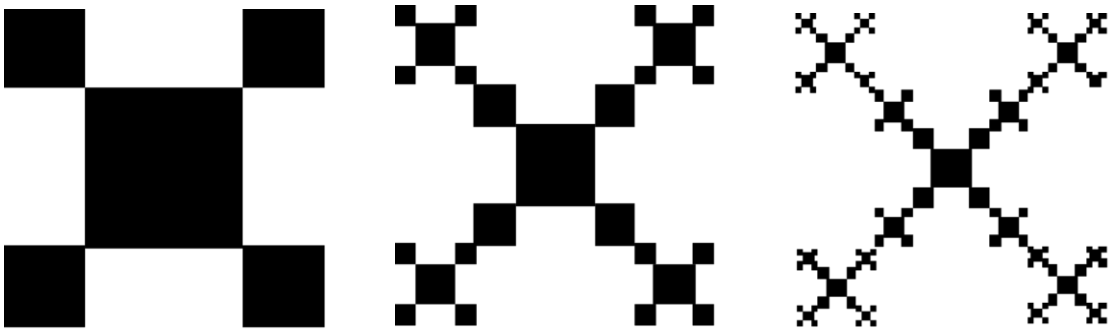


Figure 2.3 : Fractale non uniforme.

2.1.1.2 Les fractales stochastiques:

Ces fractales se distinguent des fractales géométriques par leur forme qui rappelle certains éléments de la nature. Dans les structures stochastiques la récurrence, définissant la hiérarchie, est régie par une ou plusieurs lois de probabilités précisant le choix de l'application de tel ou tel générateur à chaque itération. Ces fractales stochastiques peuvent être soit homogènes, soit hétérogènes. Les fractales stochastiques homogènes se caractérisent par la masse de la structure qui est répartie de façon uniforme à chaque niveau de hiérarchie, c'est à dire que les divers générateurs, servants à construire la fractale conservent le rapport de masse d'un niveau au suivant.

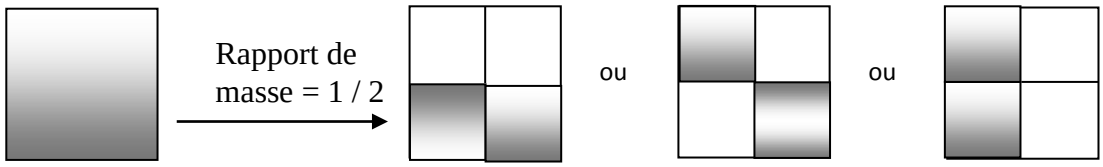


Figure 2.4 : Générateur d'une fractale stochastique homogène.

Les fractales stochastiques hétérogènes quant à elles se caractérisent par un rapport de masse qui fluctue au sein même d'une hiérarchie. Les fractales stochastiques sont à quelques exceptions remarquables prés, les seules rencontrées dans la nature. La génération d'une montagne est un exemple typique et particulièrement simple de fractales naturelles.

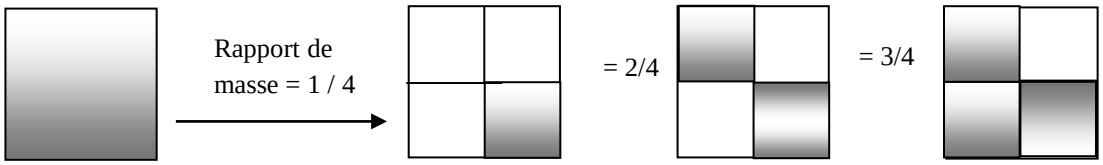


Figure 2.5 : Générateur d'une fractale stochastique hétérogène.

2.1.2 Notion d'auto- similarité :

La théorie des fractales modélise la nature de certaines courbes, et plus généralement de certains objets en y reconnaissant entre autre, une auto- similarité. Un objet est auto- similaire lorsqu'il est constitué d'objets identiques à lui-même après d'éventuelles transformations simples et cela quelle que soit l'échelle d'observation.

De nombreux exemples peuvent être donnés, l'arbre en est un. En effet, à mesure que l'on s'en approche, on peut distinguer des branches maîtresses qui globalement ressemblent à l'arbre lui-même, mais plus petites et disposées dans une direction principale qui n'est plus nécessairement verticale. Si l'on s'approche davantage, on constate qu'une branche maîtresse est elle-même constituée de branches qui ressemblent à nouveau à l'arbre mais en dimensions réduites, et selon des directions variées et ainsi de suite.

2.2 Théorie des systèmes de fonctions itérées IFS:

La théorie des IFS est entièrement basée sur la propriété d'invariance par changement d'échelle. Elle permet de générer des objets fractales à l'aide d'un ensemble de fonctions contractantes formant un système de fonctions itérées ou IFS. Elle fut étudiée par Hutchinson dans le cadre mathématique de l'autosimilarité [59], puis par Barnsley dans le cadre de la géométrie fractale [60], avec des applications en compression d'images [2].

2.2.1 Définitions :

2.2.1.1 Transformation contractante:

Soit $w : \mathbb{R}^2 \to \mathbb{R}^2$ une transformation définie sur l'espace métrique ($\mathbb{R}^2$, d) muni d'une fonction de distance à valeurs réelles d.
La transformation w est dite contractante, de facteur de contraction le réel s, $0 < s < 1$, si elle vérifie :

$$d(w(x), w(y)) \leq s.d(x, y) \qquad \forall\, x, y \in \mathbb{R}^2 \tag{2.1}$$

Comme l'évoque son appellation, c'est une transformation qui réduit la distance entre deux points images par rapport à leurs antécédents.

2.2.1.2 Point fixe :

Soit une transformation contractante $w : \mathbb{R}^2 \to \mathbb{R}^2$, Les itérations successive de w sont les transformations $w^{\circ n} : \mathbb{R}^2 \to \mathbb{R}^2$ définies par :

$$w^{\circ 0}(x) = x,\ w^{\circ 1} = w(x), \ldots\ldots w^{\circ n}(x) = w \circ w^{\circ(n-1)}(x) = w(w^{\circ(n-1)}(x)),\ n = 1, \ldots \infty$$

Une transformation contractante w: $\mathbb{R}^2 \to \mathbb{R}^2$ possède un unique point fixe, $x_f \in \mathbb{R}^2$, tel que : $w(x_f) = x_f$.

Pour tout point x élément de $\mathbb{R}^2$, la suite de nombre réels $\{w^{\circ n}(x), n = 0,1,2,\ldots\}$ converge vers x_f :

$$\lim_{n\to\infty} w^{\circ n}(x) = x_f \tag{2.2}$$

2.2.1.3 Espace métrique des fractales :

Nous devons tout d'abord définir l'espace qui supportera les fractales. A celui-ci il faudra ajouter une métrique.
Soit ($\mathbb{R}^2$, d) un espace métrique complet. Nous définissons H($\mathbb{R}^2$) comme l'espace dont les éléments sont les sous ensemble compacts non vides de $\mathbb{R}^2$.
Désormais nous appellerons fractale n'importe quel élément de H($\mathbb{R}^2$).

2.2.1.4 Distance de Hausdorff :

Considérons l'espace métrique ($\mathbb{R}^2$, d). La distance de Hausdorff entre deux ensemble A et B éléments de H($\mathbb{R}^2$), notée h_d (A, B) est définie par :

$$h_d(A, B) = \max\{d(A, B), d(B, A)\} \tag{2.3}$$

La distance de Hausdorff h_d est une métrique sur l'espace H($\mathbb{R}^2$) et (H($\mathbb{R}^2$), h_d) est un espace complet.

2.2.1.5 Transformation contractante sur l'espace H($\mathbb{R}^2$) :

Soit w une transformation contractante définie sur l'espace métrique ($\mathbb{R}^2$, d) avec s son facteur de contraction alors la transformation w : H($\mathbb{R}^2$) $\rightarrow$ H($\mathbb{R}^2$) définie par : $w(B) = \{w(x) : x \in B\}$, $\forall B \in$ H($\mathbb{R}^2$) est contractante sur (H($\mathbb{R}^2$), h_d), avec s son facteur de contraction.

2.2.1.6 Système de fonctions itérées (IFS) [60][61]:

Un IFS (Iterated Functions System) défini dans l'espace métrique complet ($\mathbb{R}^2$, d) est constitué d'un ensemble de N transformations contractantes: w_i : $\mathbb{R}^2 \rightarrow \mathbb{R}^2$ (i = 1,…..,N) dotées respectivement d'un facteur de contraction s_i .
La notation pour cet IFS est {$\mathbb{R}^2$, w_i, i=1,…..,N}, son facteur de contraction noté s est égal à : max { s_i i = 1,….., N}.

2.2.1.7 Attracteur d'un IFS :

Soit un IFS {$\mathbb{R}^2$, w_i, i = 1,….., N} avec un facteur de contraction s, et W : H($\mathbb{R}^2$)$\rightarrow$ H($\mathbb{R}^2$) un opérateur défini par : $W(B) = \bigcup_{i=1}^{N} w_i(B), \forall B \in$ H($\mathbb{R}^2$), W est appelé l'opérateur d'Hutchinson.

Il a été démontré que W est contractant, et que son facteur de contraction est celui de l'IFS. Il possède un unique point fixe $A_t \in \mathrm{H}(\mathbb{R}^2)$ qui vérifie :

$$A_t = W(A_t) = \lim_{n\to\infty} W^{\circ n}(X), \forall X \in \mathrm{H}(\mathbb{R}^2) \tag{2.4}$$

A_t est aussi appelé **attracteur de l'IFS**. Il est invariant par W car quel que soit l'ensemble de départ X, ainsi en appliquant les transformations w_i à X jusqu'à un ordre n, on finit par obtenir le même objet attracteur A_t. L'objet est également dit auto-similaire car il est composé de l'union de transformations contractantes de lui-même.

2.2.1.8 Système de fonctions itérées locales (L-IFS):

Une transformation w_i, définie dans un espace métrique complet (X,d) est dite localement contractante si : $\exists E \subset X$ telle que $w_i : E \to X$ est contractante.

Un L-IFS (Local Iterated Functions System) W est un ensemble fini de transformations localement contractantes dans un espace métrique complet (X,d): $W = \{w_i : E_i \to X, E_i \subseteq X, i = 1,\ldots N\}$ avec: $W\left(\bigcup_{i=1}^{N} E_i\right) = \bigcup_{i=1}^{N} w_i(E_i)$

2.2.2 Les IFSs et le problème inverse:

Les IFSs permettent la génération d'images totalement auto-similaires, en utilisant un nombre restreint de transformations. Des méthodes simples pour générer des fractales définies par des codes de système de fonctions itérées (IFS) ont été développées par différents chercheurs (voir Annexe).

Le problème inverse de la théorie des IFS consiste à trouver l'opérateur W codant une image, pour cela M. Barnesley [2][60] à démontrer un théorème important appelé théorème de collage. Ce théorème prouve la possibilité de reconstruire l'image compressée à partir d'un certain nombre de transformations affines et contractantes appliquées à elle-même. Il fournit une borne supérieure à la distance de Hausdorff h_d entre un point A inclus dans $\mathrm{H}(\mathbb{R}^2)$ et l'attracteur A_t d'un IFS. Le théorème indique l'inégalité suivante:

$$h_d(A, A_t) \le \frac{1}{1-s} . h_d\left(A, \bigcup_{i=1}^{N} \omega_i(A)\right) \tag{2.5}$$

Le facteur de contraction s contrôle la vitesse de convergence. Cette inégalité nous permettra de remarquer que s'il est possible de transformer un objet A de manière à vérifier $A \approx W(A)$ tout en s'assurant que W est contractant, alors l'attracteur A_t de l'opérateur W est proche de A.

2.3 Compression fractale d'images:

2.3.1. Etat de l'art :

La compression d'images fixes par fractales est une méthode qui permet d'exploiter les autosimilarités dans les images. En effet au lieu de rechercher la corrélation entre pixels adjacents, on s'intéresse à des corrélations entre parties plus au moins espacées dans l'image.

Le système de codage/décodage fractal est basé sur la construction du code IFS. Le code IFS trouvé pour une image donnée est utilisé dans la construction de l'attracteur (la fractale), qui devra ressembler à l'image initiale. La compression d'images par IFS est performante du point de vue taux de compression, car les transformations nécessitent un taux d'information assez faible comparé à celui des images originales. C'est une méthode de compression avec pertes, du fait que l'attracteur ne constitue qu'une approximation de l'image originale. Toutefois, cette méthode est applicable seulement pour des images purement fractales, vis-à-vis celles qui présentent des auto- similarités globales.

Pour les images naturelles, il s'agit de trouver les similarités permettant de les représenter sous forme de transformations contractantes. Les premières recherches tentaient de trouver des transformations globales dans le sens qu'elles agissent sur toute l'image, mais elles n'ont pas abouti vu que les images naturelles ne possèdent pas forcément la propriété d'autosimilarité globale. La solution proposée par Jacquin [3] consistant à rechercher les autosimilarités locales ou partielles a conduit au premier algorithme de compression par fractales. L'algorithme cherche à trouver des transformations localement contractantes w_i et leurs domaines de départ (blocs source) qui sont des parties de l'image, de façon à ce que l'image de ces domaines par les transformations soit assez proche de l'image de départ (blocs destination). La construction des transformations s'inspire du théorème de collage, avec la minimisation de la distance entre l'image originale et sa transformée par LIFS.

Suite au travail de Jacquin, Diverses études ont tenté d'améliorer le schéma standard à plusieurs niveaux :

- Construction d'une partition optimale pour calculer la transformation fractale [62] [63][64].
- Composition de l'ensemble des blocs domaine (global ou local) [65][66].
- Accélération de la phase de codage [67].
- Introduction de vecteurs fixes pour approximer les blocs destination à partir des blocs source [68].
- Recherches de fonctions nouvelles (affines ou autres) pour coder la redondance spatiale de l'image [69].
- Accélération du décodage : itératif, non-itératif, hiérarchique, orthogonalisation des vecteurs (blocs) source par rapport au vecteur unitaire [65][70][71][72].
- Etude théorique de la convergence du décodeur [73][74].

- Compression par fractales des signaux réels 1D [75] et extension de la méthode au codage des images vidéo [76][77].

- Utilisation des fractales dans les schémas de codages-décodage hybrides (Fractal-DCT, Fractal-DWT,…) [5] [6][7].

Pour plus de précisions, le lecteur peut se référer aux articles de synthèse de Saupe [78], de Wohlberg [79] de Koli [80] ou l'ouvrage de Barlaud [81].

2.3.2 Principe du codage fractale :

Le principe du codage fractal consiste à partitionner l'image I à coder en des blocs R_i, usuellement des carrés, qui constituent les domaines d'arrivée des transformations et qui seront appelés destination. On définit en suite un dictionnaire de domaines de départ de blocs D_j de l'image, on les appellera blocs source. Puis on définit un dictionnaire des transformations w_i.

Pour chaque bloc destination R_i, il faut donc trouver une transformation w_i et un bloc D_j qui minimise la distorsion $d(R_i, w_i(D_j))$. La transformation W de l'image I est formulée à l'aide de l'équation suivante [4] :

$$W(I) = \bigcup_{i=1}^{N} w_i(D_j) \tag{2.6}$$

Le code de l'image compressée est alors composé de l'ensemble de N transformations affines contractantes w_i, N étant déterminé par le type de partitionnement.

La phase de décompression de l'image correspond à l'application répétée des transformations composant le LIFS. L'image initiale doit avoir les mêmes dimensions que l'image compressée mais son contenu peut être quelconque. A chaque itération, l'image est transformée et se rapproche visuellement de l'image compressée (convergence due à l'ensemble des transformations). On stoppe les itérations quand on estime que l'image reconstruite est suffisamment bonne ou qu'il n'y a plus de changement d'une itération à la suivante.

Mathématiquement, elle consiste simplement à itérer la transformation W sur n'importe quelle image initiale I_0, jusqu'à ce que la convergence vers une image finale stable soit observée :

$$I_{reconstruite} = \lim_{n \to \infty} W^n(I_0) \tag{2.7}$$

Une itération consiste à parcourir les blocs destination (dans le même ordre que celui issu du codage). Les collages des blocs D_j sur les blocs R_i sont réalisés en utilisant les transformations affines contractantes w_i. Les détails à l'intérieur des blocs R_i vont devenir de plus en plus petits au cours des itérations.

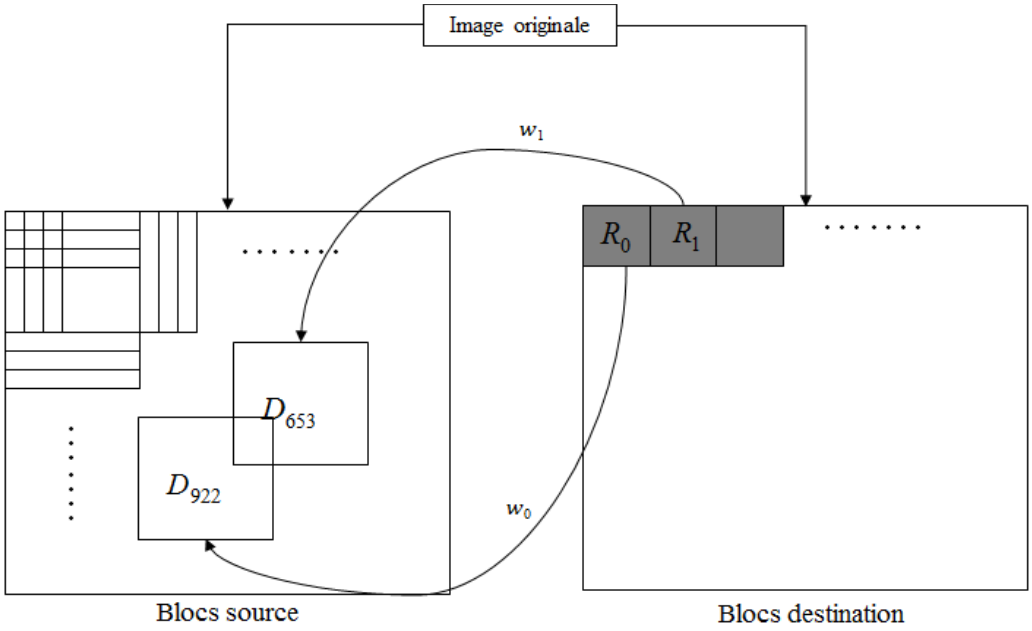


Figure 2.6 : Illustration de la procédure de codage fractale

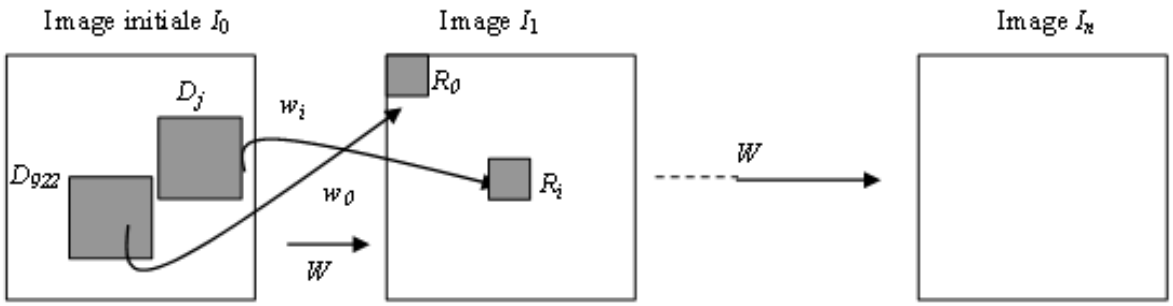


Figure 2.7 : Décodage fractale

2.3.3 Analogie avec la quantification vectorielle :

Le codage par fractales est une représentation d'un vecteur (une matrice de pixels dans une image) par un autre vecteur qui lui ressemble, mais qui a possiblement un niveau de contraste, un niveau d'intensité et une taille différente.

Les méthodes à base de quantification vectorielle comprennent une phase d'apprentissage. C'est-à-dire plusieurs images sont analysées pour générer un dictionnaire de vecteurs, qui serviront à représenter une image divisée en sous-blocs avec le moins d'erreur possible.

Pour la méthode fractale cette phase n'est pas nécessaire. Le codage fractal consiste à quantifier une section de l'image à partir d'une autre section similaire, possiblement de taille différente, ailleurs dans l'image. Une recherche de l'autosimilarité parmi une image est donc effectuée. Chaque bloc destination de l'image sera quantifié à partir d'autres blocs source de la même image et non à partir d'un dictionnaire généré à partir d'une banque d'images.

2.4 Conception du codeur fractale :

Pour concevoir un codeur d'images basé sur la géométrie fractale, on doit prendre en considération trois critères essentiels :

- Le partitionnement de l'image.
- Le calcul de la transformation fractale
- Le choix de la distance

2.4.1 Partitionnements de l'image :

Le but de la compression par fractales est de saisir la redondance visuelle locale à l'intérieur de l'image à l'aide de transformations contractantes. La transformation fractale est calculée sur une partition de l'image. Le support de l'image peut être partitionné de plusieurs façons et les partitions peuvent avoir plusieurs formes, l'essentiel c'est que l'ensemble des partitions couvre toute l'image. Un partitionnement est choisi s'il exploite mieux les autosimilarités entre les parties de l'image et réduit le nombre de transformations locales.
Différents modèles de partitionnement d'images pour la compression fractale ont été étudiés, nous citerons :

2.4.1.1 Partitionnement carré :

Le partitionnement carré est très utilisé dans la majorité des schémas de codage par fractales de fait qu'il est facile à réaliser. Il a été adopté dans la méthode de Jacquin [61] qui sera présentée en détail dans la section 2.5.1.
Si la taille des blocs est choisie ≤ 4*4, alors ces blocs sont :

-Faciles à analyser et à les classer.
-Faciles à coder et avec précision.
-Permettent une évaluation rapide pour le calcul de la distorsion inter blocs.
-Rends le système plus efficace.

Si la taille des blocs est choisie≥ 5*5, alors :

-Ils permettent une grande exploitation de la redondance.
-Ils guident théoriquement vers des taux de compression élevés.

2.4.1.2 Partitionnement Quadtree :

Une généralisation du partitionnement à taille fixe est l'utilisation d'un partitionnement quadtree. Un quadtree est une représentation de l'image sous forme d'un arbre quaternaire, où chaque nœud possède quatre fils [82]. Ce partitionnement est rigide car il est guidé par un découpage récursif en blocs carrés. Il n'est pas adapté aux formes des objets de l'image même s'il s'adapte bien au contenu de celle-ci. Pour réaliser un partitionnement de l'image en quadtree en peux suivre deux approches :

Approche de division (Quadtree Decomposition QD) :

L'arbre quaternaire est construit du haut vers le bas (top-down). La construction du quadtree consiste à diviser récursivement un bloc carré non homogène selon un prédicat donné en quatre sous blocs de même dimension. A chaque nouvelle division, les attributs des quatre blocs crées sont recalculés pour être à nouveau soumis au prédicat d'homogénéité [4].

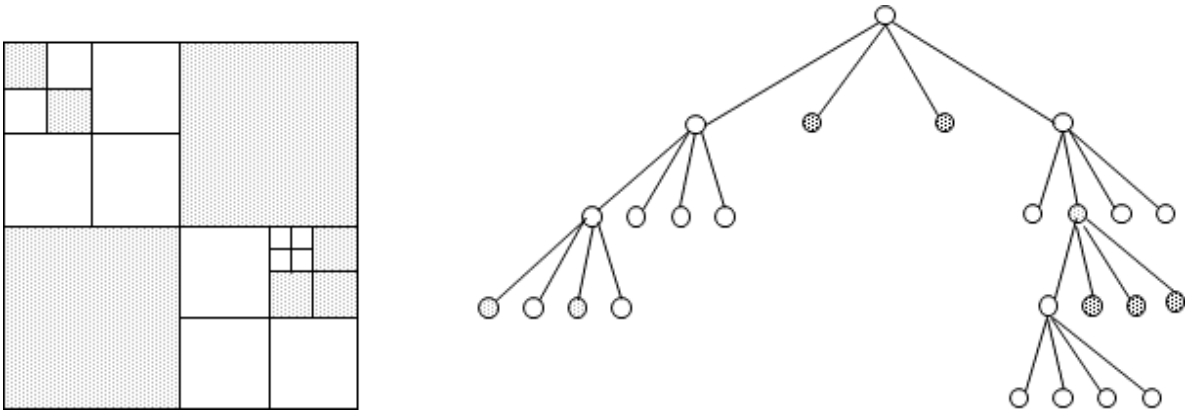


Figure 2.8 : Approche de division (Quadtree decomposition)

Approche de fusion (Quadtree Recomposition QR) :

Dans cette approche la construction du quadtree se fait du bas vers le haut. Elle consiste à regrouper les blocs adjacents égaux selon un prédicat d'homogénéité considéré. Son intérêt est de réduire le nombre total de régions. Cette approche n'est cependant pas utilisée dans le cas des images complexes, car elle exhibe des performances supérieures au niveau du temps de calcul.

Nous présentons dans la section 2.5.2 l'utilisation du partitionnement quadtree dans la compression des images par fractales.

2.4.1.3 Partitionnement horizontal-vertical (H-V)

Fisher et Menlove [83] proposent un partitionnement rectangulaire couramment appelé partition H-V pour le codage des images selon une approche fractale. Dans la partition H-V, l'image est découpée récursivement en deux rectangles, pas nécessairement de même taille, soit horizontalement soit verticalement, ce qui conduit à une partition adaptée au contenu de l'image. Le découpage est répété jusqu'à ce que toutes les parties de l'image soient couvertes. Ce partitionnement s'adapte au contenu de l'image et fait apparaître les autosimilarités entre régions de l'image. La partition n'est pas rigide comme le quadtree puisque la division d'un bloc, qui tient compte de sa texture, ne crée pas nécessairement deux blocs d'égales surfaces.

2.4.1.4 Partitionnement triangulaire adaptatif :

Fisher [83] a proposé une autre manière de partitionner l'image basée sur des triangles. Il suggère de diviser diagonalement le support rectangulaire de l'image, puis de subdiviser récursivement chacun des deux triangles parents en quatre sous-triangles enfants. La division est réalisée en joignant trois points judicieusement choisis sur les arêtes du triangle père. Ce partitionnement qui est plus flexible que le partitionnement H-V, à l'inconvénient d'être difficile à implanter et à coder.

La triangulation de Delaunay manipule aussi des figures triangulaires pour partitionner l'image. Elle utilise un pavage formé par les triangles de Delaunay en se basant sur le diagramme de Voronoï [84]. Nous verrons dans la section 2.5.3 le codage par fractales calculé sur la triangulation de Delaunay [85] [86].

2.4.1.5 Partitionnement à base de Polygones :

L'image est partitionnée adaptativement en polygones. Cette technique permet une meilleure représentativité des objets formant l'image. Elle est flexible et permet de représenter les régions avec beaucoup de détails. Dans [62] Reusens propose un partitionnement en polygones irréguliers, basé sur un mode de construction par division-fusion de la partition de Voronoi. La structure non-figée des blocs, permet d'accroître l'adaptativité de la partition au contenu de l'image. Ceci assure un contrôle de la richesse des blocs et augmente la qualité des collages.

Une autre manière pour avoir un partitionnement en polygones consiste à modifier la triangulation de Delaunay, en regroupant les triangles voisins formant des quadrilatères. Ce regroupement se fait à condition que les deux triangles candidats ne soient pas sur un contour de l'image, et que leur union génère un bloc convexe [87][88].

2.4.1.6 Partitionnement irrégulier :

Ce partitionnement permet de générer des blocs destination de formes irrégulières, en utilisant des méthodes de partitionnement usuelles. Thomas et Deravis [89], emploient la partition uniforme dans leur algorithme. Le quadtree est introduit dans le partitionnement irrégulier par Chang [90] et Ochata et Saupe [91].

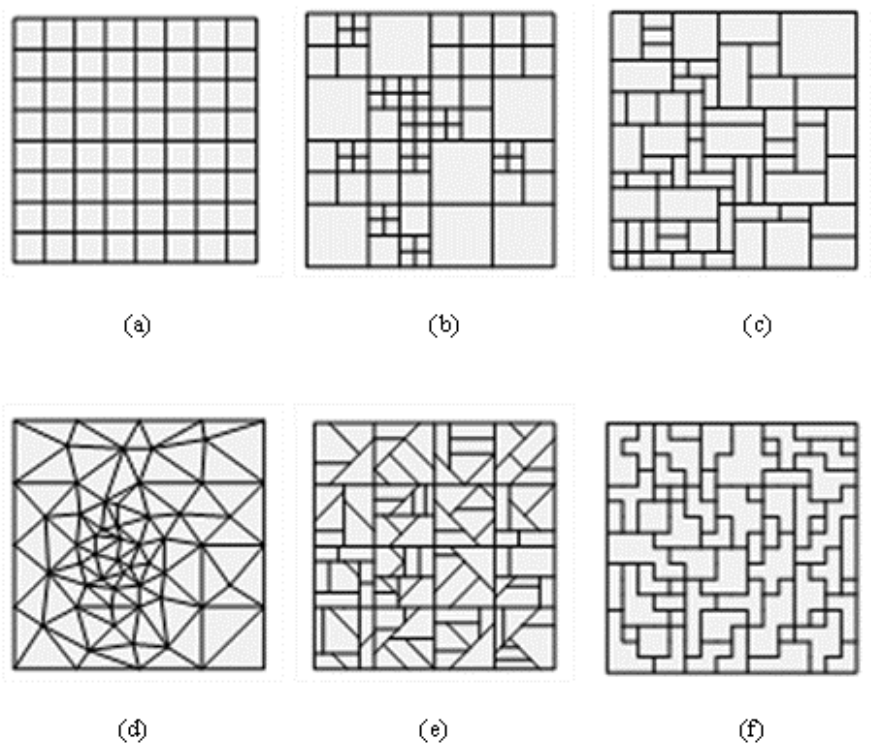


Figure 2.9 : Différents types de partitionnement : (a) Carré (b) Quadtree. (c) Horizontal-Vertical (H-V) (d) Triangulation de Delaunay (e) Polygonale (f) irrégulier [79].

2.4.2 Calcul des transformations fractales:

Les transformations fractales w_i qui réalisent l'opération de collage d'un bloc source D_j, sur un bloc destination R_i sont dites itératives et se décomposent en deux parties, une partie géométrique et une partie massique. La forme générale d'une transformation fractale est donnée par [3] :

$$W = \sum_{i=1}^{N} w_i = \sum_{i=1}^{N} M_i \circ G_i \tag{2.8}$$

où :

N : représente le nombre de blocs destination dans l'image originale.

G_i : est la transformation géométrique. Elle agit linéairement sur un bloc de l'image (contraction, rotations, translations)

M_i : est la transformation massique. Elle agit sur la luminance et le contraste du bloc déformé pour approximer le bloc destination R_i.

Les transformations fractales paramètrent la qualité de l'image compressée. Plus elles sont nombreuses et riches plus elles permettent de faire correspondre des blocs image entre eux. Etablir un ensemble riche de transformations pour permettre de coder au mieux un bloc destination par un bloc source donné n'est pas forcément bénéfique à tous points de

vue : Premièrement, un ensemble de transformations avec un cardinal élevé fait baisser le taux de compression de l'image. En effet, pour chacune d'elles il faut stocker les paramètres pour chaque bloc destination de l'image, ce qui augmente la taille de mémoire unitaire occupée par un bloc destination. Deuxièmement, dans le cas où les transformations deviennent particulièrement souples et nombreuses, nous nous trouvons dans une situation où n'importe quel blocs source peut coder n'importe quel bloc destination.

2.4.3 Choix de la distorsion :

On a vu précédemment la métrique de Haussdorf, qui est une mesure de distance entre deux sous-ensembles de $\mathbb{R}^n$. Dans le cas des images, on définit la distance comme une mesure de degré de proximité ou de ressemblance entre deux images. La construction d'une fonction de distance ou une mesure de distorsion entre les images numériques, revient à celui d'une fonction de distance entre les blocs des images. Pour pouvoir comparer un bloc source transformé à un bloc destination, il existe plusieurs métriques qui peuvent représenter un certain type d'erreur.

Les équations (2.9) et (2.10) sont celles de distance absolue moyenne et de la valeur supérieure des distances entre chaque pixel.

$$d_{abs} = \frac{1}{MN}\sum_{m=1}^{M}\sum_{n=1}^{N}\left|R_i(m,n) - w_i(D_j(m,n))\right| \tag{2.9}$$

$$d_{\sup} = \sup_{(m,n)\in I}\left|R_i(m,n) - w_i(D_j(m,n))\right| \tag{2.10}$$

Où R_i est le bloc destination et $w_i(D_j)$ est le bloc source transformé. Les deux blocs ont une taille M par N et proviennent de la même image I.

La métrique erreur quadratique moyenne (EQM) est la plus utilisée. Elle est donnée par [3] :

$$EQM = \frac{1}{MN}\sum_{m=1}^{M}\sum_{n=1}^{N}\left[R_i(m,n) - w_i(D_j(m,n))\right]^2 \tag{2.11}$$

2.5 Principales méthodes de compression d'images par fractales:

L'étude est consacrée à la méthode originale de Jacquin (algorithme standard) [3], la méthode de Fisher [4] et la méthode basée sur la triangulation de Delaunay [92]. Nous présentons aussi les diverses améliorations apportées à la méthode originale de Jacquin, pour accélérer la phase du codage.

2.5.1 Méthode de Jacquin:

L'approche de Jacquin [3] [61] est fondée sur une partition régulière à géométrie carrée. L'image est partitionnée en blocs destination carrés R_i, de taille fixe égale à $B \times B$. L'algorithme recherche pour chacun des blocs destination R_i, le bloc source carré D_j de taille $D = 2B$ qui minimise la distorsion $d(R_i, \hat{R}_i)$ où $\hat{R}_i$ est l'approximation de R_i calculée à partir du bloc source D_j par la transformation w_i qui se décompose en deux parties :

- **Une transformation géométrique ou spatiale** :

Elle permet de ramener la taille du bloc D_j à l'échelle du bloc destination R_i. Le bloc transformé, noté $G_i(D_j)$ est obtenu par décimation des pixels du bloc source. Un pixel de coordonnées (x_m, y_n) dans $G_i(D_j)$ est donné par l'équation suivante :

$$G_i(D_j)(x_m, x_n) = \frac{1}{4}\left[d_j(x_k, y_l) + d_j(x_k, y_{l+1}) + d_j(x_{k+1}, y_l) + d_j(x_{k+1}, y_{l+1})\right] \quad (2.12)$$

Où (x_k, y_l) sont les coordonnées d'un pixel de niveau de gris noté d_j à l'intérieur du bloc D_j.

D'autres transformations applicables aux blocs décimés ne font que déplacer les pixels dans les blocs, sont appelées isométries. Il existe huit isométries appliquées au bloc carré :

- Identité: Les positions des pixels du bloc ne changent pas.
- Réflexion par rapport à l'axe horizontal: (j = (B-1)/ 2)
- Réflexion par rapport à l'axe vertical: (i = (B-1)/ 2)
- Réflexion par rapport à la première diagonale: (i = j)
- Réflexion par rapport à la deuxième diagonale: (i + j = B-1)
- Rotation de 90° par rapport au centre de bloc
- Rotation de180° par rapport au centre de bloc
- Rotation de -90° par rapport au centre de bloc

- **Une transformation massique** :

Elle agit sur le bloc source décimé (après application des isométries) pour approximer le bloc destination R_i. La transformation massique proposée par Jacquin est une transformation linéaire donnée par :

$$\hat{R}_i = M_i(D_j) = s_i G_i(D_j) + o_i 1 \quad (2.13)$$

où s_i est le coefficient d'échelle et o_i est le coefficient de décalage. s_i et o_i contrôlent respectivement la variance et la moyennes des niveaux de gris des pixels.

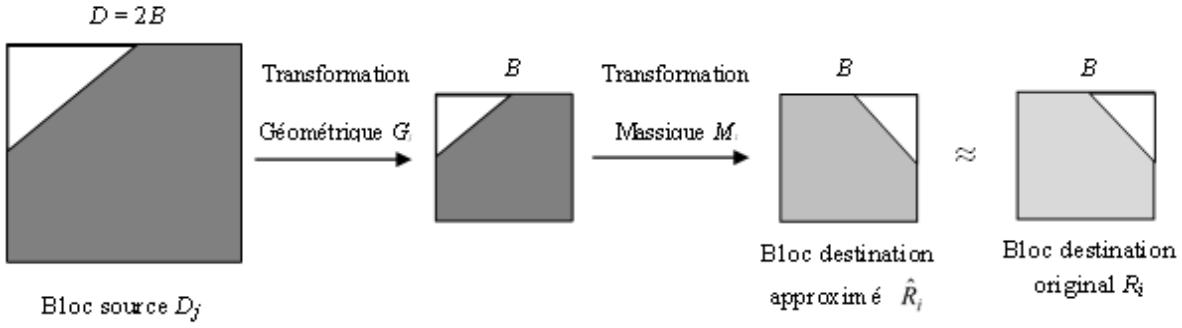


Figure 2.10 : Effet des deux transformations G_i et M_i sur un bloc source.

La complexité de la transformation massique dépend de la nature du bloc R_i considéré. Jacquin propose pour cela de classifier les blocs carrés à l'aide de la méthode développée par Ramamurthi and Gersho [93]: trois classes regroupent les blocs homogènes, les blocs texturés et les blocs avec contours. Selon la classe à laquelle appartient le bloc destination R_i, une transformation lui est associée de la manière suivante:

Si *Ri* est un bloc homogène:

Aucune recherche à faire, On approxime le bloc R_i par un bloc uniforme avec un niveau de gris égal à la moyenne des niveaux de gris de R_i, dénoté par $\overline{R}_i$, et la transformation massique M_i, qui génère ce type de bloc est l'absorption par o_i :

$$\hat{R}_i = \overline{R}_i \,. \tag{2.14}$$

Si R_i est un bloc texturé:

Pour les blocs texturés, on restreint notre attention aux transformations massiques qui sont composées d'une modification du contraste et d'un décalage de la luminance de la forme suivante:

$$w_i(D_j) = M_i(G_i(D_j)) = s_i \times (G_i(D_j)) + o_i \tag{2.15}$$

où s_i appartient à l'ensemble {0.7, 0.8, 0.9, 1.0} et l'entier o_i est compris entre -255 et 255.

Si R_i est un bloc avec contours:

La transformation massique utilisée, dans ce cas-là, est la composition d'une modification du contraste, d'un décalage de luminance et d'une isométrie i_n de la forme:

$$w_i(D_j) = M_i(G_i(D_j)) = i_n(s_i \times (G_i(D_j)) + o_i) \tag{2.16}$$

où s_i appartient à l'ensemble {0.5, 0.6, 0.7, 0.8, 0.9, 1.0} et o_i est compris entre -255 et 255.

Lorsque le bloc destination est texturé ou contient des contours, le coefficient d'échelle s_i est calculé de manière que les écarts types des blocs R_i et $G_i(D_j)$ sont égaux. Le coefficient de

décalage o_i est calculé d'une façon à ce que les moyennes des pixels des deux blocs soient égales.

Jacquin [94] propose dans un deuxième temps de rediviser les blocs parents collés en quatre sous blocs destination de taille $B/2 \times B/2$ pixels appelés blocs enfants. Les blocs obtenus sont comparés à leurs correspondants dans l'image originale, au sens d'une mesure d'erreur. Si l'erreur est supérieure à un seuil donné ils sont codés séparément en recherchant dans l'image le meilleur bloc source de taille $B \times B$. Le processus de collage est appelé dans ce cas collage enfant. Si pour un bloc parent, trois ou quatre collages enfants sont nécessaires, on code seulement les quatre collages enfants. Si un ou deux collages enfants sont nécessaires, le bloc parent est codé par le collage parent suivi des collages enfants.

Pour le contrôle de la contraction de la transformation fractale, Jacquin montre que le facteur de contraction d'une transformation massique dépend du facteur d'échelle s_i et est égale à s_i^2. Le facteur de contraction des autres transformations massiques (décalage, absorption) est égal à l'unité. Il montre aussi que le facteur de contraction de la transformation géométrique est égale à un. Il assure ainsi la contraction de la transformation fractale en imposant la condition : $s_i^2 < 1$.

2.5.2 Méthode de Fisher :

La procédure de codage proposée par Fisher [4] se base sur la construction d'un arbre quadtree en fixant sa profondeur maximale à l'avance, ce qui impose la taille minimale $B_{\min}$ des blocs destination. La taille $B_{\max}$ est aussi imposée. L'algorithme de codage cherche, pour chaque bloc destination de taille B, un bloc source de taille $2B$, centré sur l'un des points d'un réseau régulier. Fisher à définit trois types de réseaux à utiliser en fonction du pas de chevauchement de l'image à compresser à savoir :

1. Un réseau pour lequel le pas de chevauchement est constant pout toute profondeur du quadtree. Le dictionnaire ainsi obtenu par ce réseau contient en moyenne autant de petits blocs que de grands blocs. Il offre les meilleurs résultats en général. Son inconvénient est que le temps nécessaire pour effectuer la génération est très élevé à cause du grand nombre de blocs.
2. Un réseau dans lequel il n'y a pas de chevauchement, dans ce cas le dictionnaire contient plus de petits blocs que de grands.
3. Un réseau où le pas de chevauchement est fonction de $B_{\min}$ et $B_{\max}$, on aura plus de grands blocs que de petits blocs, habituellement utilisés dans les régions uniformes.

Le codage d'un bloc destination se fait par recherche du bloc source approprié, qui permet de minimiser l'erreur entre le bloc destination et son approximation. Si la distance minimale est supérieure à un seuil prédéfini (tolérance), et si le niveau du bloc destination dans le quadtree est inférieur à la profondeur maximale de celui-ci, alors le bloc destination est redivisé et la procédure de codage est relancée sur chacun des quatre sous-blocs créés. Si la

distance minimale est inférieure au seuil fixé, on considère le bloc source similaire au bloc destination et les paramètres de transformation sont stockés.

Fisher décrit l'opération de collage d'un bloc source sur un bloc destination en utilisant la forme suivante:

$$w_i\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{bmatrix} a_i & b_i & 0 \\ c_i & d_i & 0 \\ 0 & 0 & s_i \end{bmatrix}\begin{pmatrix} x \\ y \\ z \end{pmatrix} + \begin{pmatrix} e_i \\ f_i \\ o_i \end{pmatrix} \tag{2.17}$$

où :

(x,y) : sont les coordonnées d'un pixel intérieur au bloc source D_j

z : est le niveau de gris du pixel.

a_i, b_i, c_i, d_i, e_i et f_i : sont les coefficients de la transformation géométrique ou spatiale qui ramène les pixels du bloc source D_j à l'intérieur du bloc destination R_i en remplaçant les groupes de 2×2 pixels par leur moyenne. La transformation spatiale est contractante si le module de déterminant $\begin{vmatrix} a_i & b_i \\ c_i & d_i \end{vmatrix}$ est inférieur à 1.

o_i et s_i : contrôlent respectivement une différence de moyenne des niveaux de gris et une variation du contraste entre les blocs destination et les blocs source.

Le calcul de la transformation w_i, se fait par minimisation de l'erreur quadratique (EQM) entre R_i et $w_i(D_j)$, cette erreur s'écrit:

$$\varepsilon = \frac{1}{M}\sum_{k=1}^{M}(s_i D'_j(k) + o_i - R_i(k))^2 \tag{2.18}$$

M étant le nombre total de pixels dans le bloc destination et $D'_j = G_i(D_j)$ est le bloc source décimé. La minimisation de cette erreur en fonction du facteur d'échelle s_i (contraste) et du coefficient de décalage o_i (offset), est immédiate puisqu'il suffit d'annuler les dérivées partielles en s_i et o_i.

Les valeurs optimales de s_i et o_i sont données par les formules suivantes :

$$\begin{aligned} s_i &= \frac{M\sum_{k=1}^{M} D'_j(k)R_i(k) - \sum_{k=1}^{M} D'_j(k)\sum_{k=1}^{M} R_i(k)}{M\sum_{k=1}^{N} D'_j(k)^2 - (\sum_{k=1}^{N} D'_j(k))^2} \\ o_i &= \frac{1}{M}(\sum_{k=1}^{M} R_i(k) - s_i\sum_{k=1}^{M} D'_j(k)) \end{aligned} \tag{2.19}$$

où les $R_i(k)$ (k=1,…M) sont les valeurs des pixels du bloc destination R_i, et $D'_j(k)$ (k =1,…M) celles des pixels du bloc source décimé.

L'erreur se calcule donc par l'équation suivante :

$$\varepsilon = \frac{1}{M}\left[\sum_{k=1}^{M} D'_j(k)^2 + s_i\left(s_i \sum_{k=1}^{M} R_i(k)^2 - 2\sum_{k=1}^{M} R_i(k)D'_j(k) + 2o_i \sum_{k=1}^{M} R_i(k)\right) + o_i\left(Mo_i - 2\sum_{i=1}^{M} D'_j(k)\right)\right] \quad (2.20)$$

Pour accélérer la phase de codage, Jacobs et al. [95] proposent de classifier les blocs source et destination. Il s'agit de diviser un bloc carré en quatre quadrants, puis calculer la moyenne A_i et la variance V_i pour chaque bloc. La méthode de classification repose sur le fait qu'il est toujours possible d'orienter le bloc de manière à ce que la suite des valeurs $A_i$vérifie l'une des trois inégalités suivantes :

1. $A_1 \geq A_2 \geq A_3 \geq A_4$
2. $A_1 \geq A_2 \geq A_4 \geq A_3$
3. $A_1 \geq A_4 \geq A_2 \geq A_3$

Tout bloc carré peut donc être associé à l'une des trois classes. Selon les variances V_i, chacune des classes peut être subdivisée en 24 sous classes, ce qui nous donne en tout 72 classes.

2.5.3 Méthode de Davoine basée sur la triangulation de Delaunay :

2.5.3.1 Triangulation de Delaunay :

Définitions et propriétés [84] [92]:

Région de Voronoï :

On désigne par P un ensemble composé de n points p_i de l'espace IR^2 appelés aussi sites ou germes :

$$P = \left\{p_i \in \mathrm{IR}^2, i = 1, \cdots, n\right\} \quad (2.21)$$

Soit p un élément de P. On appelle polygone de Voronoï associé au site p la région $Vor_P(p)$ composée de l'ensemble de points de IR^2 plus proches, au sens de la distance euclidienne d, de p que tous les autres points de P :

$$Vor_P(p) = \left\{x \in \mathrm{IR}^2, d(x, p) \leq d(x, q), \forall q \in P - p\right\} \quad (2.22)$$

Diagramme de Voronoï :

On décrit le diagramme de Voronoï comme l'union de tous les polygones de Voronoï de P :

$$Vor_n(P) = \bigcup_{p \in P} Vor_P(p) \tag{2.23}$$

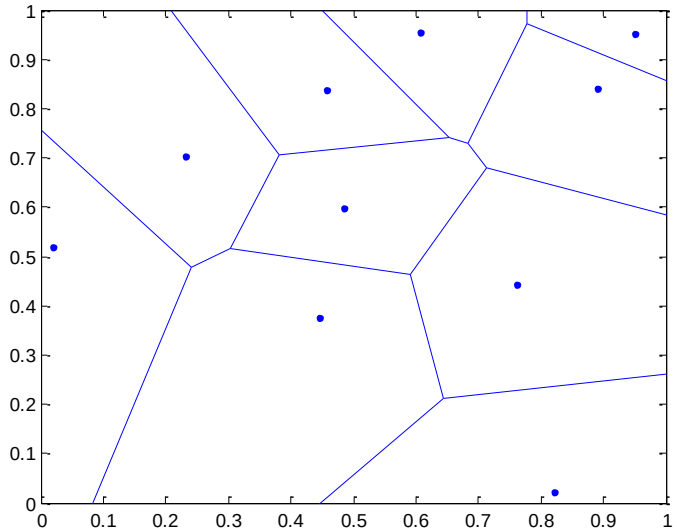


Figure 2.11 : Diagramme de Voronoï de l'ensemble *P* formé de *n* points.

Le diagramme de Voronoï possède les propriétés suivantes :

- Les polygones de Voronoï sont convexes.
- Le diagramme couvre toute l'image
- Une arête de Voronoï, séparant deux cellules C_i et C_j, est portée par la médiatrice du segment $[p_i, p_j]$.
- Le sommet de Voronoï séparant trois cellules C_i, C_j et C_k est le centre du cercle circonscrit au triangle de sommets p_i, p_j et p_k.
- Pour chaque sommet *S* du diagramme de Voronoï, le cercle passant par les trois points voisins à ce sommet, ne contient aucun autre point de *P*.

La triangulation de Delaunay :

On appelle triangulation de Delaunay de l'ensemble *P*, et on note *Del*(*P*), le dual du diagramme de Voronoï de *P*. Les sommets de la triangulation de Delaunay sont les points $p_i \in P$. Deux sommets sont reliés par une arête si les polygones de Voronoï correspondantes sont voisines.

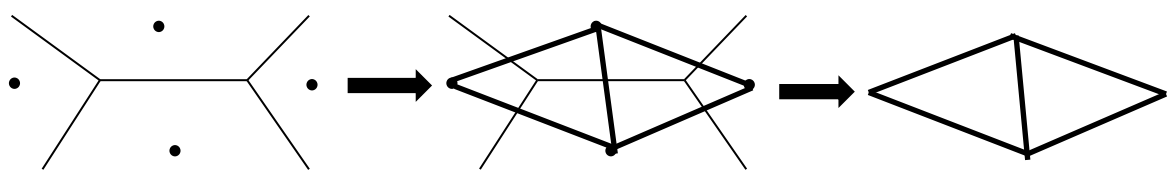


Figure 2.12 : Construction du dual du diagramme de Voronoï.

Notons que seul le couple de points (point à gauche, point à droite) ne possède pas d'arrête de Voronoï, ils ne sont donc pas reliés. Voici une illustration du Diagramme de Voronoï et de sa triangulation réciproque.

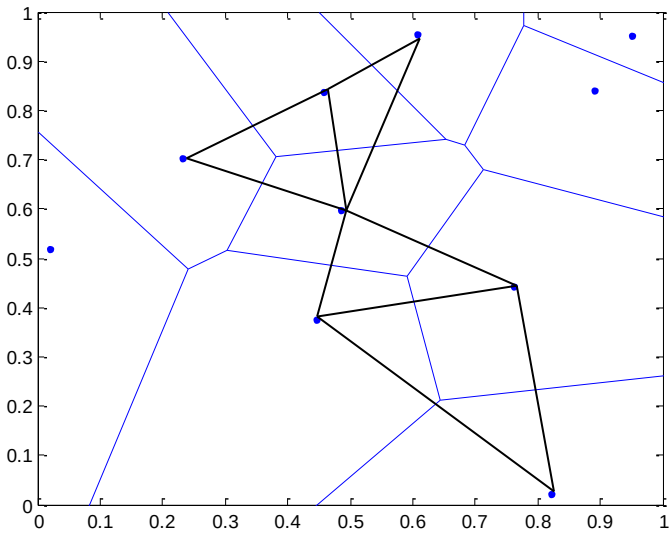


Figure 2.13 : La triangulation de Delaunay, duale du diagramme de Voronoï.

On peut par la dualité déduire les propriétés suivantes :

- La triangulation de Delaunay est unique.
- La triangulation de Delaunay est une triangulation complète.
- Les cercles passants par les trois sommets de chaque triangle ne contiennent aucun autre site en leur intérieur.

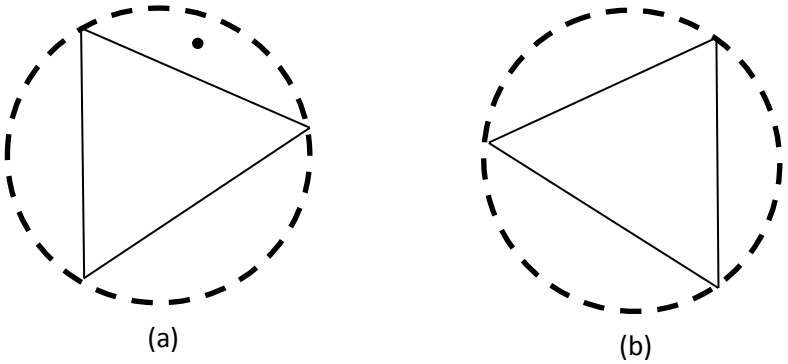


Figure 2.14 : (a) Triangle non Delaunay (b) Triangle Delaunay

Cette dernière propriété est essentielle, et elle va être utilisée pour caractériser la triangulation de Delaunay sans avoir à recourir à la dualité avec le diagramme de Voronoï. Elle va aussi être utilisée comme critère de choix des triangles à construire, lors de l'exécution d'une triangulation.

2.5.3.2 Utilisation de la triangulation de Delaunay pour la compression par fractales :

La triangulation de Delaunay livre une partition de l'image sous forme triangulaire. Il est possible dès lors entreprendre une compression. Le processus de compression reste identique à celui évoqué dans la méthode de Jacquin, en considérant les blocs triangulaires L'algorithme nécessite une partition en triangle de Delaunay *R*. Les triangles destination R_i sont recherchés parmi une triangulation régulière de l'image contenant les blocs sources D_j appelée partition *D*. Il s'agit d'associer à chacun des blocs R_i le bloc D_j qui minimise l'erreur entre le bloc R_i et le bloc D_j transformé par w_i [92].

Partitionnement Adapté au Contenu de l'image

Afin d'avoir un maximum de ressemblance entre les différents blocs source et destination, Davoine [92] a proposé un processus de division et fusion appliqué sur la partition *R*. La procédure est initialisée avec une distribution régulière de points sur le support de l'image, donnant un nombre limité de triangles. La division consiste à rajouter des points sur le barycentre des triangles dont le niveau de gris n'est pas homogène (critère de la variance). La phase de division se poursuit jusqu'à convergence, elle s'arrête si la surface des triangles est inférieure à une valeur limite [85]. La phase de fusion, quant à elle supprime les triangles inutiles, pour lesquels les voisins ont une valeur moyenne de niveaux de gris très proche. Elle permet d'obtenir des plus grands triangles dans les zones de l'image où le niveau de gris est constant.

Calcul de la transformation fractale :

La transformation w_i transformant un triangle source D_j en un triangle destination R_i est donnée par:

$$\begin{aligned} w_i(D_j) &= w_i(x_m, y_m, f(x_m, y_m)) = (x'_m, y'_m, f(x'_m, y'_m)) \\ &= (v_i(x_m, y_m), s_i f(x_m, y_m) + o_i) \end{aligned} \tag{2.24}$$

Où :

$f(x_m,y_m)$: est la luminance du pixel de coordonnées (x_m, y_m) à l'intérieur du bloc D_j.

$f(x'_m,y'_m)$: est la luminance du pixel de coordonnées (x_m', y_m') à l'intérieur du bloc R_i

v_i : dénote la transformation géométrique d'un triangle D_j sur un triangle R_i.

Les coefficients s_i et o_i contrôlent respectivement le contraste et la luminosité de la fonction des niveaux de gris.

Le collage d'un triangle sur un autre se fait à l'aide d'une transformation affine :

$$v_i \begin{pmatrix} x_m \\ y_m \end{pmatrix} = \begin{pmatrix} a_i & b_i \\ c_i & d_i \end{pmatrix} \begin{pmatrix} x_m \\ y_m \end{pmatrix} + \begin{pmatrix} e_i \\ f_i \end{pmatrix} = \begin{pmatrix} x'_m \\ y'_m \end{pmatrix} \tag{2.25}$$

Les coefficients a_i, b_i, c_i, d_i, e_i, et f_i sont calculés en considérant les trois sommets du triangle de départ $(x_1, y_1), (x_2, y_2)\ et\ (x_3, y_3)$ et les trois sommets du triangle d'arrivée $(x_1^{'}, y_1^{'}), (x_2^{'}, y_2^{'})\ et\ (x_3^{'}, y_3^{'})$. La déformation d'un triangle source peut donc être calculée à partir de ses trois sommets et des trois sommets du triangle destination. On obtient de cette façon six équations à six inconnues données sous la forme matricielle suivante :

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 \\ x_2 & y_2 & 1 & 0 & 0 & 0 \\ x_3 & y_3 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_1 & y_1 & 1 \\ 0 & 0 & 0 & x_2 & y_2 & 1 \\ 0 & 0 & 0 & x_3 & y_3 & 1 \end{bmatrix} \begin{bmatrix} a_i \\ b_i \\ e_i \\ c_i \\ d_i \\ f_i \end{bmatrix} = \begin{bmatrix} x'_1 \\ x'_2 \\ x'_3 \\ y'_1 \\ y'_2 \\ y'_3 \end{bmatrix} \tag{2.26}$$

Les isométries peuvent être aisément réalisées grâce à cette matrice de transposition.

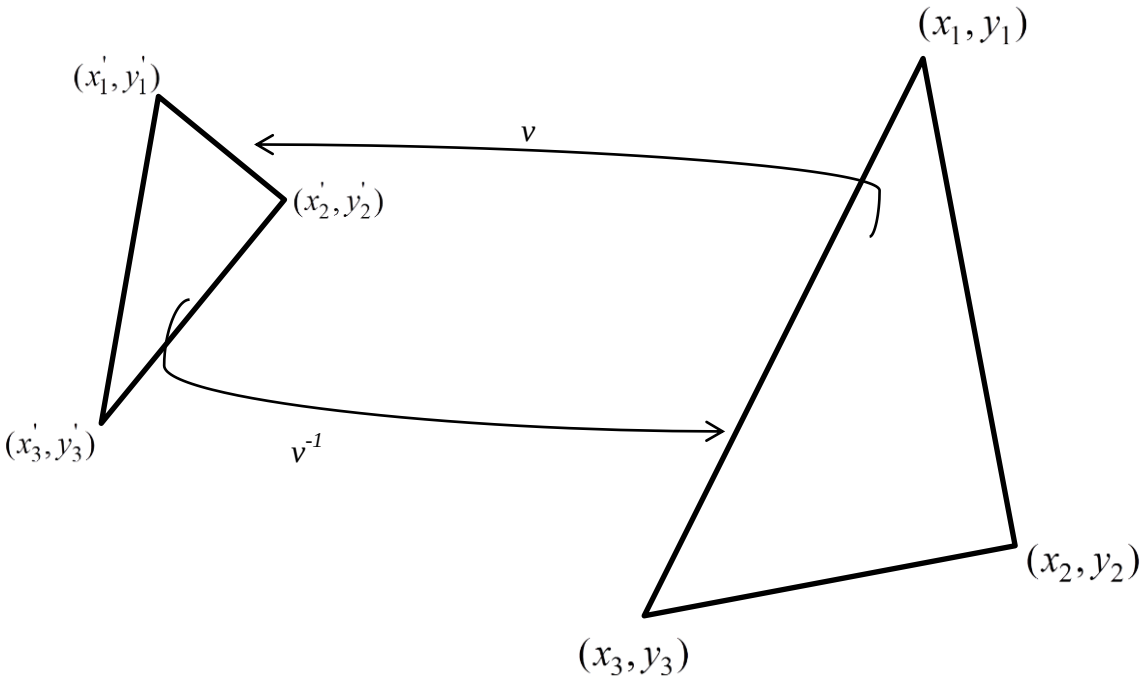


Figure 2.15 : Transformation d'un triangle.

Le calcul de la distance $d(R_i, w_i(D_j))$ considère tous les pixels inclus dans le bloc R_i et les pixels inclus dans le bloc D_j après sous échantillonnage. Si le bloc R_i contient M pixels, la distance est donnée par :

$$d(R_i, w_i(D_j)) = \sum_{m=1}^{M} (f(x'_m, y'_m) - s_i f(v_m^{-1}(x'_m, y'_m) - o_i)^2 \quad (2.27)$$

où (x_m', y_m') sont les coordonnées d'un pixel d'indice m dans le bloc destination R_i.

Les coefficients d'échelle s_i et de décalage o_i optimaux sont calculés de façon à annuler les dérivées partielles de l'erreur d par rapport à s_i et o_i, de façon similaire que celle utilisée par Fisher.

Le codage de la transformation w_i consiste à mémoriser la partition R adaptée au contenu de l'image et l'information nécessaire au codage des N transformations w_i.

2.5.4 Accélération du codage par fractales :

La durée importante de la phase de codage par fractales est due essentiellement au grand nombre de comparaisons entre les blocs destination et les blocs source. Les principales approches de réduction du temps de calcul se basent sur la classification de blocs. Dans le schéma de classification, les blocs destination et source sont groupés en un nombre fini de classes selon leurs caractéristiques communes. Durant la phase de codage, seul les blocs source ayant la même classe que le bloc destination, seront comparés à celui-ci. Deux méthodes de classification ont déjà été traitées précédemment, celle proposée par Jacquin et celle utilisée par Fisher. Un autre paramètre de classification qui pourrait s'avérer efficace est la dimension fractale [58]. La dimension fractale s'estime par une analyse conjointe spatio-fréquentielle. Elle donne l'information sur l'état de rugosité d'une surface (image) observée. Dans les travaux de Hamzaoui [96], les blocs source sont classifiés en regroupant leurs vecteurs caractéristiques en cellules de Voronoï dont les centres sont issus de l'image de test ou d'un ensemble d'images d'apprentissage. Une autre méthode de classification des blocs basée sur la transformée en cosinus discrète (DCT), consiste à classer les blocs en trois classes : bloc homogène, bloc avec contour diagonal/sous diagonal et bloc avec contour horizontal/vertical. Elle est basée sur le calcul de deux coefficients de la DCT à savoir les coefficients horizontaux et verticaux de plus basses fréquences [97].

Une autre manière de limiter le nombre de comparaisons inter-blocs est la quantification des blocs source. Lepsoy [98] organise l'ensemble des blocs source dans une structure arborescente, dans le but d'accélérer le codage sur un partitionnement carré régulier. Davoine et al [99] proposent de quantifier les triangles source à l'aide d'un quantificateur vectoriel de façon à ne conserver dans la partition source qu'un nombre réduits de triangles représentatifs de l'image. Boss et al. [100] ont proposé une méthode qui utilise la même philosophie que le LBG mais en gardant l'aspect des LIFS durant la phase de classification.

D'autres méthodes de réduction de complexité de l'algorithme classique, basées sur la réduction du nombre des blocs source ont été présentées. Monro et Dubridge [101] localise l'ensemble des blocs source relativement au voisinage du bloc destination courant. Fondée sur l'hypothèse que des blocs source qui sont voisins au bloc destination concerné, sont bien

admis pour l'assortiment de ce bloc. Saupe [102] a proposé une procédure de codage fractale de complexité réduite à celle de la méthode classique, en fusionnant l'algorithme de compression basé sur le quadtree de Fisher et l'algorithme de recherche d'une approximation du plus proche voisin de Arya et al [103]. D'autres travaux se basent sur la diminution du cardinal de l'ensemble des blocs sources par l'élimination d'un certain nombre de blocs source ayant une faible variance [104] ou une entropie élevée [105].

Distasi et al [106] ont proposé de situer chaque bloc image par rapport à l'erreur relative à un bloc destination moyen. Cette erreur est utilisée pour placer les blocs dans un kd-tree qui servira à trouver rapidement un domaine susceptible de coder un bloc destination donné. D'autres approches de réduction du temps de calcul ont été proposées avec l'élaboration des schémas hybrides et le parallélisme de l'algorithme [107][108][109][110].

2.6 Conclusion:

Nous avons introduit dans ce chapitre les bases mathématiques de la théorie des systèmes de fonctions itérées utilisés dans les schémas de compression par fractales. Dans un second temps nous avons présenté trois principaux algorithmes de compression des images selon l'approche fractale.

Bien évidemment le type de partitionnement joue un rôle important dans la compression, et le taux change avec le changement du type et cela en minimisant chaque fois le nombre de transformations locales. L'application des transformations sur le domaine fréquentiel à l'aide d'une DWT (Discrete Wavelet Transform), ont permis de déboucher sur des méthodes hybrides plus efficaces. C'est précisément sur les méthodes hybrides basées sur les fractales et les ondelettes que se situe le prochain chapitre.

CHAPITRE 3

Chapitre 3 : Codage hybride par fractales et ondelettes

La transformée en ondelettes est une technique consistant à décomposer un signal en sous bandes. Elle présente l'avantage de faire ressortir à la fois les grandes variations et les détails de l'objet analysé. L'intérêt de l'introduction de la théorie des ondelettes en compression d'images par fractales est justifié par l'existence des autosimilarités dans la décomposition multi-résolution. Dans ce chapitre, nous donnons les aspects théoriques des ondelettes, puis nous présentons quelques méthodes de compression d'images basées les ondelettes et les fractales. A la fin du chapitre nous présentons notre schéma de compression hybride fractale-ondelette sans perte.

3.1 Théorie d'ondelettes:

La première base orthogonale dont les fonctions s'obtiennent à partir des translatées dilatées d'une seule fonction mère a été construit par Haar en 1909 [111]. Dans les années 1940, Gabor [112] a réalisé des analyse temps-fréquence, par transformée de Fourier fenêtrée, avec une fenêtre Gaussienne. Par la suite en 1984, Goupillaud, Grossman et Morlet [113] ont proposé une méthode de reconstruction des signaux sismiques permettant une restauration des hautes fréquences à l'aide de la représentation temps-fréquence des signaux.

Sur la base de ces travaux, plusieurs chercheurs tels Meyer [114], Mallat [115], Lemarié [116], Daubechies [117], Chui [118], Wornell [119] et Holscneider [120] ont étudié et développé cet outil puissant, tant du point de vue pratique que du point de vue théorique, pour aboutir à la théorie de l'analyse par ondelettes largement utilisée actuellement.

Le champ d'application des ondelettes est très varié [121]. Nous nous intéressons à son utilisation pour la compression d'images [122]. Avant d'entamer l'analyse par ondelettes on commence par un rappel de l'analyse de Fourier.

3.1.1 Rappel sur la transformée de Fourier:

La transformée de Fourier (TF) [123] est un outil fondamental en traitement de signal. Elle s'écrit pour une fonction *f*:

$$TF\big(f(v)\big) = \hat{f}(v) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{+\infty} e^{-ivt} f(t) dt \tag{3.1}$$

La transformée inverse existe et s'écrit alors :

$$\overline{TF}\hat{f}(t) = f(t) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{+\infty} e^{ivt}\, \hat{f}(v) dv \tag{3.2}$$

La transformée de Fourier, qui consiste à décomposer f sur des fonctions e^{ivt} (très localisées en fréquence mais non localisées en espace) permet une analyse de l'ensemble des fréquences de la fonction *f* (analyse spectrale). L'analyse spectrale est un outil fondamental en traitement du signal, notamment pour l'étude des signaux stationnaires (c'est à dire ayant des propriétés statistiquement invariantes au cours du temps). Cependant, il existe de nombreux cas où la relation temps-fréquence ou espace-fréquence est indispensable, notamment l'étude des signaux transitoires où des événements imprévisibles apparaissent.

3.1.2 Transformée de Fourier à fenêtre glissante:

Afin d'améliorer la localisation en temps, tout en préservant au mieux la localisation en fréquence, on peut utiliser la transformée de Fourier à fenêtre glissante de Gabor [112], elle s'écrit:

$$TF_{fen} f(v, b) = \int_{-\infty}^{+\infty} h^*(t - b) e^{-ivt} f(t) dt \tag{3.3}$$

où *f* est la fonction à analyser et *h* une fonction "fenêtre" appropriée centrée sur *b*. h^*est le conjugué complexe de *h*.

En fait, cette transformée effectue le calcul de la transformée de Fourier de la fonction *f*(.)*h*(.-*b*) à *b* fixé. De nombreux choix de *h* sont possibles. En général, cette fonction est choisie à support compact, suffisamment régulière et bien localisée en espace et en fréquence. Le choix classique de *h* est celui de la gaussienne.

La transformée de Fourier à fenêtre glissante de Gabor [112] permet une représentation temps-fréquence ou espace-fréquence des fonctions. Cependant la taille de la fenêtre est fixe, cette représentation n'est pas adaptée aux fonctions ou signaux ayant des composantes de tailles très différentes de la taille de la fenêtre.

3.1.3 Les ondelettes:

On peut définir l'ondelette comme étant une fonction mathématique à valeur moyenne nulle et limitée dans le temps (d'où le nom de petite onde), elle doit être bien localisée en temps et en fréquence. Les ondelettes sont générées à partir d'une ondelette mère $\psi(t)$, par translation et dilatation :

$$\psi_{s,\tau}(t) = \frac{1}{\sqrt{s}}\psi\left(\frac{t-\tau}{s}\right) \tag{3.4}$$

avec s est le facteur de dilatation (ou coefficient d'échelle), et τ est le facteur de translation. Si $s > 1$ l'ondelette sera plus étalée et correspondra donc à une fréquence plus faible, si $s < 1$ l'ondelette sera plus étroite et correspondra à une fréquence plus élevée que celle de l'ondelette mère. τ est le paramètre de translation, ce qui correspond à un changement de phase dans le domaine de Fourier.

3.1.4 Transformée en ondelette continue :

La transformée en ondelettes continue d'un signal $f(t) \in L^2(\mathbb{R})$ est définit comme :

$$\gamma(s,\tau) = \int f(t).\psi^*_{s,\tau}(t).dt \tag{3.5}$$

où $\psi^*_{s,\tau}$ est le conjugué complexe de $\psi_{s,\tau}$.

$\gamma(s,\tau)$ sont appelés coefficients d'ondelettes

La transformée en ondelettes mesure la similitude entre le signal et l'ondelette, pour différentes dilatations et translations de cette dernière. Cette transformation est dite continue car c'est une fonction continue des paramètres de dilatation, translation.

Pour retrouver le signal d'origine $f(t)$ à partir de sa transformée en ondelettes. L'ondelette doit vérifier certaines conditions d'admissibilité :

- $\int \psi(t)dt = 0$ (L'ondelette est oscillante)
- $\int |\psi^2(t)|dt < +\infty$ (Conservation de l'énergie)
- $C_\psi = \int \frac{|\widehat{\psi(\nu)}|^2}{|\nu|} d\nu < +\infty$, où $\hat{\psi}(\nu)$ est la transformée de Fourier de l'ondelette.

Alors la formule de reconstruction par transformée en ondelette inverse est :

$$f(t) = \frac{1}{C_\psi} \iint \frac{1}{|s|^2} \gamma(s,\tau) \psi_{s,\tau}(t) ds d\tau \qquad (3.6)$$

3.2 Transformée en ondelettes discrète et analyse multi-résolution (AMR):

Meyer et Mallat [114][115] ont développé un cadre fonctionnel (une famille de sous-espaces de L^2 (ℝ)), appelé une analyse multirésolution (AMR), dans laquelle les ondelettes sont construites.

Définition 1 :

Une analyse multirésolution (AMR) de $L^2(\mathbb{R})$ est une suite croissante de sous espaces emboîtés $\{V_j\}_{j\in\mathbb{Z}}$

Tels que :

1. $\forall j \in Z : V_j \subset V_{j+1}$
2. L'intersection des V_j est nulle, soit $\bigcap_{j=-\infty}^{+\infty} V_j = \emptyset$
3. L'union des V_j : $\bigcup_{j=-\infty}^{+\infty} V_j$, est dense dans $L^2(\mathbb{R})$.
4. $f(x)$ est dans V_j si et seulement si sa version contractée par un facteur 2 est dans V_{j+1}, c'est-à-dire :

$$f(x) \in V_j \iff f(2x) \in V_{j+1} \iff f(2^{-j}x) \in V_0$$

5. si $f(x)$ est dans V_j, alors ces translatées entières sont dans V_j :

$$f(x) \in V_j \Leftrightarrow f(x-k) \in V_j\,, \quad \forall k \in \mathbb{Z}$$

6. Il existe une fonction $\phi(x)$ de V_0 telle que la famille $\{\phi(x-k), k \in \mathbb{Z}$ est une base de V_0. Une telle fonction ϕ est appelé une fonction d'échelle de l'AMR.

Dans le cas dyadique, si l'on peut trouver une fonction ϕ de V_0 telle que $\phi(x-k)$ soit une base de Riesz de V_0 alors : $\left\{\phi_{j,k}(x) = 2^{\frac{j}{2}}\phi\left(2^j x - k\right),\ \ k \in \mathbb{Z}\right\}$ est une base de Riesz de V_j.

Définition 2 :

$\phi(x\text{-}k)$ est une base de Riesz de V_0 si pour tout $f(x)$ de V_0, il existe une séquence unique $\alpha_{k\in\mathbb{Z}} \in L^2(\mathbb{Z})$ tel que :

$$f(x)=\sum_{k\in Z}\alpha_k\phi(x-k) \quad \text{et} \quad A\left(\sum_k|\alpha_k|^2\right)^{1/2}=\left\|\sum_k\alpha_k\phi(x-k)\right\|_2^2\le B\left(\sum_k|\alpha_k|^2\right)^{1/2}$$, avec A et B sont des constantes indépendantes de f telles que $0 < A < B < \infty$.

3.2.1 Espaces des approximations et des détails :

Les espaces V_j constituent **des espaces d'approximation**. La projection d'une fonction $f(x)$ de $L^2(\mathbb{R})$ sur le sous-espace V_j constitue une approximation de $f(x)$ à l'échelle 2^j. Toute fonction $f(x)\in L^2(\mathbb{R})$ peut être approximée par une fonction de V_j selon :

$$P_jf(x)=\sum_{k\in\mathbb{Z}}a_{j,k}\phi_{j,k}(x) \tag{3.7}$$

où P_j est le projecteur de f à l'échelle 2^j sur l'espace V_j et P_jf est une approximation de f à l'échelle 2^j. Les coefficients $a_{j,k}$ sont appelés coefficients d'approximation et sont obtenus par :

$$a_{j,k}=\int_{IR} f(x)\phi_{j,k}(x)dx \tag{3.8}$$

La fonction d'échelle ϕ peut se décomposer dans la base : $\{\sqrt{2}(2x-k), k\in\mathbb{Z}\}$ de V_1 puisque ϕ est dans $V_0\subset V_1$. Soit h_k ses coordonnées dans cette base, alors :

$$\phi(x)=\sum_k h_k\sqrt{2}\phi(2x-k) \tag{3.9}$$

Afin de réduire la redondance d'information dans la représentation précédente, on considère l'espace W_j complément de l'espace d'approximation V_j tel que :

$$V_{j+1}=V_j\oplus W_j \tag{3.10}$$

L'espace W_j contient l'information manquante lors du passage de V_j à V_{j+1}, il est aussi appelé **l'espace des détails** de l'échelle j. l'union des W_j est dense dans $L^2(\mathbb{R})$.

W_j est le supplémentaire de V_j dans V_{j+1}. Dès lors, si l'on peut trouver une fonction $\psi(x)$ dans W_0, telle que la famille $\{\psi(x-k), k \in \mathbb{Z}\}$ soit une base orthonormée de W_0, alors ψ est appelé ondelette.

Dans le cas dyadique : $\left\{\psi_{j,k}(x) = 2^{\frac{j}{2}}\psi(2^j x - k) \;/\; j \in \mathbb{Z},\ k \in \mathbb{Z}\right\}$ constitue une base de $L^2(\mathbb{R})$.

Comme la fonction d'ondelette $\psi(x)$ est dans $W_0 \subset V_1$, on peut écrire :

$$\psi(x) = \sum_k g_k \sqrt{2}\phi(2x-k) \tag{3.11}$$

Tel que : $g_k = (-1)^k h_{1-k}$

3.2.2 Orthogonalité et bi-orthogonalité :

Une AMR $\{\widetilde{V_j}\}_{j\in\mathbb{Z}}$, de fonction d'échelle $\tilde{\phi}$ est duale d'une autre AMR $\{V_j\}_{j\in\mathbb{Z}}$ de fonction d'échelle ϕ si et seulement si :

$$\forall j,k,j',k' \in Z \qquad \left\langle \phi_{j,k} \middle| \widetilde{\phi}_{j',k'} \right\rangle = \delta_{j,j'}\delta_{k,k'} \tag{3.12}$$

où $\left\langle \phi_{j,k} \middle| \widetilde{\phi}_{j',k'} \right\rangle$ désigne le produit scalaire des fonctions $\phi_{j,k}$ et $\widetilde{\phi}_{j',k'}$.

Soit deux AMR duales $\{V_j\}_{j\in\mathbb{Z}}$, $\{\widetilde{V_j}\}_{j\in\mathbb{Z}}$ engendrées par les fonctions d'échelles duales ϕ et $\tilde{\phi}$, dotées des sous espaces complémentaires $\{W_j\}_{j\in\mathbb{Z}}$et $\{\widetilde{W_j}\}_{j\in\mathbb{Z}}$ engendrées par des ondelettes ψ et $\tilde{\psi}$. ψ et $\tilde{\psi}$ sont duales l'une à l'autre, on aura donc :

$$\forall j,k,j',k' \in Z \qquad \left\langle \phi_{j,k} \middle| \widetilde{\phi}_{j',k'} \right\rangle = \left\langle \psi_{j,k} \middle| \widetilde{\psi}_{j',k'} \right\rangle = \delta_{j,j'}\delta_{k,k'} \tag{3.13}$$

Ondelette orthogonale :

La fonction ψ est une ondelette **orthogonale** si et seulement si : $\forall j \in \mathbb{Z},\ V_j = \widetilde{V_j}$ et $W_j = \widetilde{W_j}$. Les translatée dilatées de ϕ forment alors des bases orthonormées des V_j, et celles de ψ des bases orthonormées de W_j, on a alors $V_{j+1} = V_j \oplus W_j$ et $V_j \perp W_j$ et on peut prendre : $\{\tilde{\psi}, \tilde{\phi}\} = \{\psi, \phi\}$.

Dans le cas dyadique, une fonction ψ de $L^2(\mathbb{R})$ est une ondelette orthogonale si et seulement, la famille $\left\{\psi_{j,k}(x) = 2^{\frac{j}{2}}\psi\left(2^j x - k\right), \quad j \in \mathbb{Z},\ k \in \mathbb{Z}\right\}$ forme une base orthonormée de $L^2(\mathbb{R})$.

Ondelette bi-orthogonale :

On dit que deux ondelettes duales ψ et $\tilde{\psi}$ sont bi-orthogonales si et seulement si : $\forall j \in \mathbb{Z}, \widetilde{V_j} \perp W_j$ et $V_j \perp \widetilde{W_j}$, c'est-à-dire

$$\forall j, j', k'k', \left\langle \tilde{\phi}_{j',k'} \middle| \psi_{j,k} \right\rangle = \left\langle \phi_{j,k} \middle| \tilde{\psi}_{j',k'} \right\rangle = 0. \tag{3.14}$$

Pour établir que deux ondelettes sont bi-orthogonales, il suffit de montrer que :

$$\forall l \in Z, \left\langle \tilde{\phi} \middle| \psi(x-l) \right\rangle = \left\langle \tilde{\psi} \middle| \phi(x-l) \right\rangle = 0 \tag{3.15}$$

3.2.3 Algorithme pyramidale de Mallat [115] :

Soit le sous espace W_j le supplément orthogonal de V_j dans V_{j+1}.On peut écrire pour toute fonction $f(x)$ de $L^2(\mathbb{R})$ que l'approximation à l'échelle j+1 résulte de l'addition de détails (contenues dans W_j) à l'approximation à l'échelle j :

$$\mathrm{Pr}oj(f(x)/V_{j+1}) = \mathrm{Pr}oj(f(x)/V_j) + \sum_k d_{j,k}\psi_{j,k}(x) \tag{3.16}$$

En continuant l'itération, on obtient alors :

$$\mathrm{Pr}oj(f(x)/V_{j+1}) = \mathrm{Pr}oj(f(x)/V_{j0}) + \sum_{i=j_0}^{j}\sum_k d_{i,k}\psi_{i,k}(x) \tag{3.17}$$

On obtient donc une approximation plus fine de la fonction en ajoutant à une approximation plus grossière de cette fonction les détails successifs pour arriver à la bonne résolution.

Les coefficients d'approximation $a_{j,k}$ à l'échelle j se calculent récursivement selon la formule suivante [115]:

$$a_{j,k} = \sum_l h_{l-2k} a_{j+1,l} \tag{3.18}$$

Les coefficients d'approximation à l'échelle j s'obtiennent à partir des coefficients d'approximation à l'échelle plus fine j+1 par convolution, puis décimation.

Quant aux coefficients de détails, on a :

$$d_{j,k} = \sum_l g_{l-2k} a_{j+1,l} \tag{3.19}$$

Les coefficients de détails à l'échelle j s'obtiennent à partir des coefficients d'approximation à l'échelle plus fine j+1 par convolution et décimation.

Notons que les filtres apparaissant dans les relations d'analyse (3.18) et (3.19) sont les retournés dans le temps des filtres h_k et g_k

Pour l'étape de synthèse on utilise la formule (3.18) pour écrire :

$$\sum_l a_{j,l}\phi_{j,l}(x) = \sum_l a_{j-1,l}\phi_{j-1,l}(x) + \sum_l d_{j-1,l}\psi_{j-1,l}(x) \tag{3.20}$$

Ce qui donne [115] :

$$a_{j,k} = \sum_l a_{j-1,l} h_{k-2l} + \sum_l d_{j-1,l} g_{k-2l} \tag{3.21}$$

L'algorithme de synthèse procède à un sur-échantillonnage par 2 des coefficients d'approximation et de détail à une échelle j, puis au filtrage et addition de ces séries.

Les filtres H et *G* ont des propriétés particulières qui leur sont conférées par l'analyse multirésolution (AMR) : Le filtre *H* est un filtre passe bas. On a $H(0) = 1$ et $H\left(\frac{1}{2}\right) = 0$. De plus $G(0) = 0$ et $\left|G\left(\frac{1}{2}\right)\right| = 1$. *G* est passe-haut [115].

Le caractère passe bas de *H* correspond bien à une approximation, et le caractère passe haut de *G* est associé aux détails. Ces filtres sont appelés filtres miroirs en quadrature. La figure 3.1 montre qu'ils permettent une cascade analyse-synthèse avec exacte reconstruction.

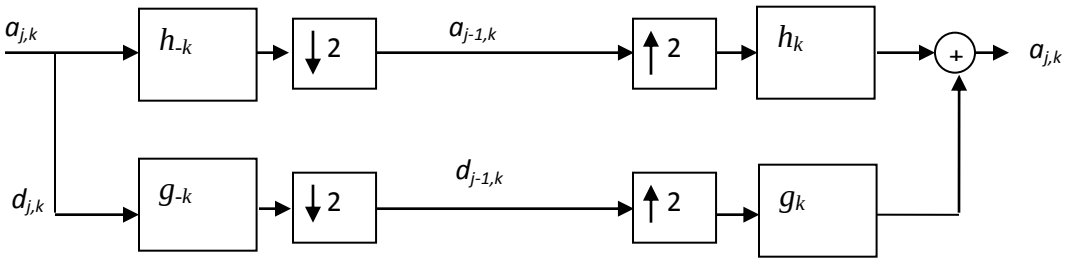


Figure 3.1: Analyse-synthèse dans la transformée orthogonale en ondelettes.

En résumé, l'opération de base de l'AMR est la décomposition du signal en deux parties : une approximation et les détails du signal. L'approximation est obtenue en projetant le signal sur les translatées d'une fonction basse fréquence appelée fonction échelle. Cette projection isole les variations lentes par un filtrage passe-bas. Les détails du signal sont obtenus par projection sur un filtre passe-haut appelé ondelette. La réponse de ce filtre donne les variations rapides du signal ou les détails.

3.2.4 Analyse multirésolution bidimensionnelle:

L'analyse multirésolution de $L^2(\mathbb{R})$ peut se généraliser au cas multidimensionnel $L^2(\mathbb{R}^n)$, en considérant des fonctions à plusieurs variables obtenues par translation d'un vecteur à coordonnées entières. Les ondelettes dans ce cas ne se généralisent aussi facilement, cependant dans le cas bidimensionnel, à l'usage du produit tensoriel, une analyse multirésolution peut se généraliser aux fonctions à deux variables (adapté aux images 2D) de la manière suivante :

Soit $\{V_j\}_{j\in\mathbb{Z}}$une AMR de $L^2(\mathbb{R})$ et ϕ une fonction d'échelle associée à l'ondelette ψ. La fonction d'échelle et les ondelettes pour $L^2(\mathbb{R}^2)$ sont définies comme suit :

$$\begin{aligned}
\phi(x,y) &= \phi(x)\phi(y) \\
\psi^H(x,y) &= \phi(x)\psi(y) \\
\psi^V(x,y) &= \psi(x)\phi(y) \\
\psi^D(x,y) &= \psi(x)\psi(y)
\end{aligned} \tag{3.22}$$

Par passage de la transformée de Fourier des quatre fonctions définies précédemment, on en déduit que ϕ est associé à un filtre passe bas dans les deux directions, ψ^H donne le niveau de détail en x, ψ^V en y et ψ^D en diagonal.

Une telle représentation multirésolution de l'image fournit à chaque échelle les quatre sous images décorrélées, une image de faible résolution, une image de détails horizontaux, une image de détails verticaux, et une image de détails diagonaux. On obtient cette décomposition par filtrage passe-bas (L) et passe-haut (H) dans le sens horizontal (X) et vertical (Y). Après chaque filtrage un sous-échantillonnage est effectué. On obtient ainsi une décomposition en quatre sous-bandes classiquement dénotée LL, LH, HL, HH. La sous bande LL est une version filtrée passe-bas et sous résolue de l'image originale. La sous-bande HL apporte le contenu haute-fréquence dans la direction horizontale et basse fréquence dans la direction verticale. En d'autres termes, les lignes et contours horizontaux Y sont principalement mis en valeur. A l'inverse, la sous-bande

LH répond plus fortement aux contours et lignes verticales alors que la sous bande HH contient principalement les structures diagonales. La figure 3.2 représente un étage d'un schéma de décomposition multirésolution bidimensionnel.

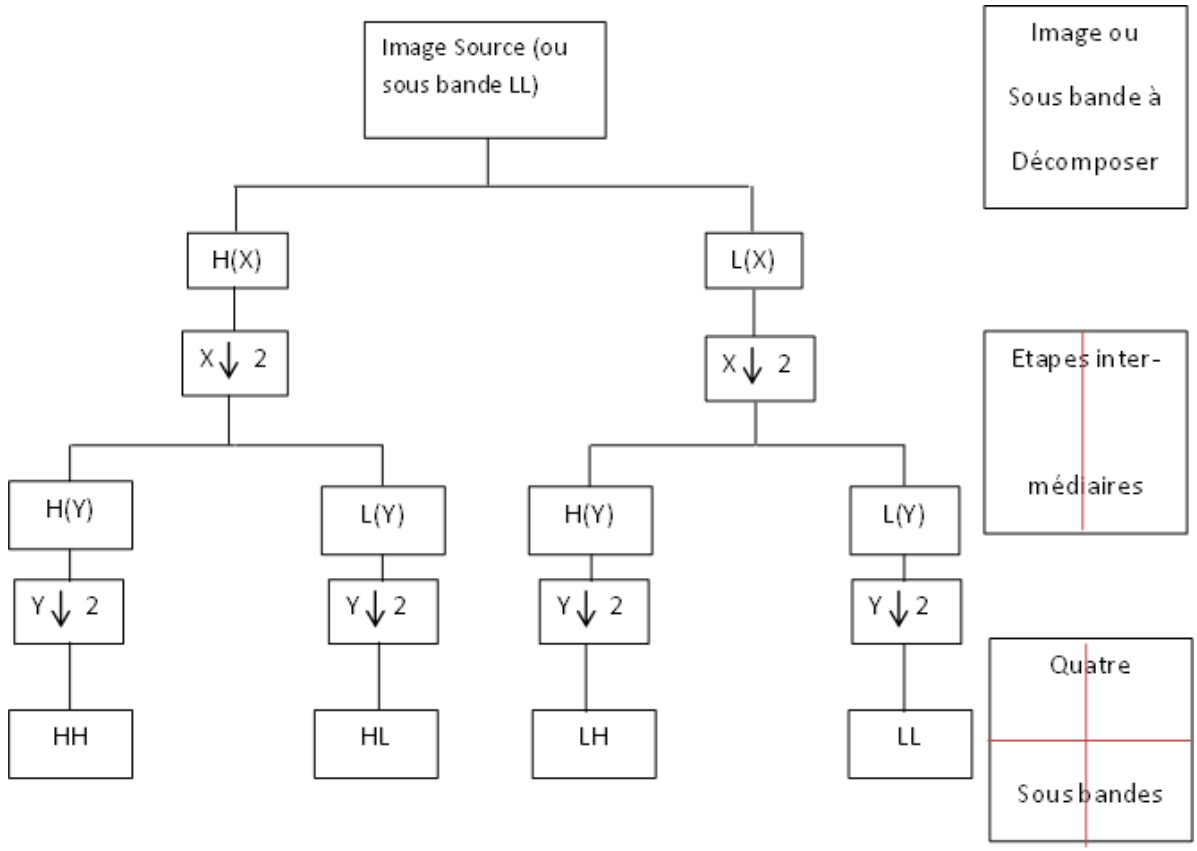


Figure 3.2 : Principe de décomposition 2D

Pour chaque niveau de décomposition, on obtient quatre images de résolutions inférieures. C'est la sous-bande LL qui est utilisée pour établir une nouvelle décomposition. La limite est atteinte quand la sous-bande LL mesure un coefficient, soit huit sous-bandes pour une image 256×256. Le résultat de la décomposition s'organise de la manière suivante :

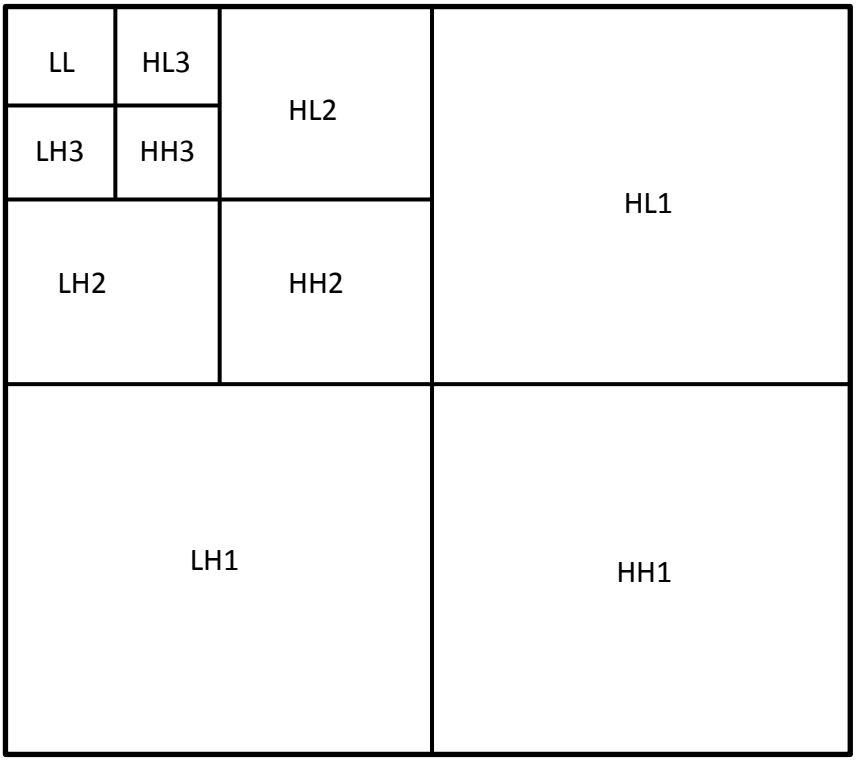


Figure 3.3 : Décomposition multi-résolution d'une image.

3.3 Propriétés des ondelettes :

Chaque ondelette possède des propriétés spécifiques, on donne ici quelques propriétés souhaitables des ondelettes.

3.3.1 Localisation :

Elle est obtenue en faisant varier deux paramètres, l'un d'échelle et l'autre de translation. Ces deux paramètres permettent d'ajuster l'analyse de l'ondelette en agissant comme un zoom d'agrandissement ou de réduction pour une position donnée. Cette propriété est très intéressante pour l'extraction d'indices visuels. Ainsi, elle permet de caractériser précisément des phénomènes locaux ou des événements de plus grande échelle.

3.3.2 Oscillation (Moments) :

Ce paramètre se traduit graphiquement par plusieurs passages par zéro de l'ondelette. On peut aussi la définir comme le nombre *M* de moments nuls de la fonction ou comme le nombre de fois où elle est dérivable. Le nombre de moments utilisé pour une ondelette dépend du nombre de coefficients.

On dit qu'une fonction f a N moments si :

$$\int_{-\infty}^{+\infty} x^k f(x)dx = 0 \qquad \text{pour } 0 \le k < N$$

Ainsi, si une ondelette a N moments, alors les coefficients de la transformation en ondelettes de tout polynôme de degré inférieur à N seront nuls. Donc si une certaine image s'approxime bien à partir de polynômes de degré assez petit (et donc est assez régulière et comporte peu de variations), sa transformation en ondelettes devrait aussi avoir des coefficients assez petits. On recherche donc des ondelettes avec suffisamment de moments pour réduire la taille des coefficients transformés.

3.3.3 Support compact :

On dit qu'une fonction f à un support compact : si support(f) = $\{\overline{x \in \text{IR}/\ f(x) \neq 0}\}$ est un ensemble compact. On remarquera d'abord qu'un support compact permet d'emmagasiner complètement l'information sur une ondelette à partir de ses coefficients, car le nombre de ceux qui sont non-nuls sera fini. Aussi, un petit support permet une localisation plus forte de l'information sur l'image contenue dans les coefficients transformés. Ainsi, cela augmente les chances de bien séparer les zones de l'image plus détaillées de celles qui le sont moins, permettant ainsi de quantifier différemment les coefficients transformés selon les besoins dans chaque zone.

3.3.4 Régularité :

La régularité agit sur la qualité de la reconstruction d'un signal. En image, la régularité est plus importante que la sélectivité en fréquence. L'ordre de la régularité d'une ondelette est égal au nombre de ses dérivées continues. Quand une ondelette est très lisse, elle est dite régulière.

3.3.5 Symétrie :

On dit qu'une fonction $f : \mathbb{R} \to \mathbb{C}$ est symétrique autour de $x_0 \in \mathbb{R}$: si $f(x_0 + x) = f(x_0 - x)$, $\forall x \in \mathbb{R}$. De même, on dit qu'une fonction f est antisymétrique autour de $x_0 \in \mathbb{R}$: si $f(x_0 + x) = -f(x_0 - x)$, $\forall x \in \mathbb{R}$.

La symétrie des ondelettes pourra avoir deux utilités. D'abord, il a été observé que les images du monde réel possèdent une certaine symétrie. On voudra donc garder cette symétrie en reconstruisant les images par des ondelettes symétriques. Aussi, la symétrie est utile pour trouver une base d'ondelettes à une fonction définie sur un intervalle.

3.4 Les ondelettes dans la compression d'images :

Les transformées en ondelettes conservent l'énergie du signal et possèdent notamment des algorithmes rapides, elles sont donc bien adaptées à la compression d'image [122]. Les algorithmes classiques de compression s'appuyant sur la transformée en ondelettes suivent le schéma de codage par transformée. Dans un premier temps, est appliquée la transformée en ondelettes pour décorréler les données de l'image. La transformée en ondelettes ne fait que changer la représentation de l'image en concentrant l'information dans un nombre restreint de coefficients. La distribution des coefficients possède la forme d'un pic et donc une entropie plus faible. Nous pouvons donc éliminer les coefficients de faible amplitude par un seuillage sans créer de distorsion importante dans l'image reconstruite grâce à la propriété d'invariance de l'énergie contenue dans l'image originale et sa transformée. La compression proprement dite se fait alors au terme d'une opération de quantification. La structure approximative de l'image a une représentation très compacte qui autorise alors plusieurs stratégies de quantification des coefficients d'ondelettes. La dernière étape dans le schéma de compression consiste à coder les coefficients quantifiés par un codage entropique. La décompression est faite en appliquant les opérations duales de la phase de compression dans l'ordre inverse.

3.4.1 Quantification vectorielle des coefficients d'ondelettes :

Barlaud et al [124] ont proposé de quantifier vectoriellement les coefficients d'ondelettes orthogonales puis bi-orthogonales. Leur algorithme consiste à construire un dictionnaire multirésolution. Chaque sous image de la décomposition est codée séparément. Le dictionnaire est ainsi constitué de plusieurs sous dictionnaires de petite taille, optimisés pour le codage de chacune des résolutions. Il en résulte un gain de temps au niveau de la recherche des vecteurs de codage, ainsi qu'une meilleure restitution des hautes fréquences, qui étaient lissées par l'algorithme LBG [46] appliqué directement sur l'image originale. La quantification est faite de manière plus ou moins grossière en fonction des résolutions, en tenant compte de la sensibilité de l'œil humain.

3.4.2 Quantification par arbres de zeros:

3.4.2.1 EZW (Embedded Zerotree Wavelet coder):

L'algorithme de codage EZW (Embedded image coding using Zerotrees of Wavelet coefficients) a été introduit par Shapiro [125]. Le principe est d'utiliser la structure hiérarchique de la transformée en ondelettes et consiste à exploiter les redondances inter-échelles des coefficients.

Il est ainsi possible de définir les relations parents-enfants entre les coefficients de même orientation et issus des sous bandes adjacentes. Les sous bandes doublant de résolution à mesure

que l'échelle diminue, un coefficient de la sous-bande LH_l disposera de 4 coefficients descendant à la même localisation spatiale dans la sous-bande LH_{l-1}. Ces relations sont également définies pour les sous-bandes d'orientation HL, HH et pour les niveaux de décomposition $l = 2 \ldots N$.

L'algorithme EZW tire fortement profit de la structure pyramidale de la décomposition en ondelettes afin de réaliser une compression efficace de l'image. La compression provient principalement de deux processus :

Le premier en l'encodage itératif des coefficients dont les valeurs sont fixées par rapport au seuil lors de l'itération considérée, En divisant par deux le seuil à chaque itération, le coefficient est dit significatif (significant) si la valeur absolue du coefficient est supérieure ou égale à ce seuil, et non-significatif (unsignificant) dans le cas contraire.

Le second processus de compression réside dans l'utilisation d'arbre de zéros. Lorsqu'une racine d'arbre est identifiée, tous les coefficients descendants ne sont plus encodés.

Ces comparaisons donnent alors lieu à un codage progressif des coefficients par plan de bits. Les indices représentant les valeurs de comparaisons sont ensuite encodés à l'aide d'un algorithme arithmétique.

3.4.2.2 SPIHT (Set Partitionning In Hierarchchal Trees):

L'algorithme SPIHT [126] (Set Partitionning In Hierarchchal Trees) proposé par Said et Pealman s'inspire et améliore la stratégie d'EZW. La différence essentielle entre EZW et SPIHT est la façon dont les coefficients des arbres sont construits, triés et découpés. Ainsi la structure même des arbres de zéro est différente. Dans EZW, un arbre de zéros défini par un coefficient racine et ses descendants ont tous la valeur zéro à l'intérieur d'un plan de bits. SPIHT utilise lui deux types d'arbres de zéros. Le premier (type A) consiste en une simple racine ayant tous ses descendants à zéro pour un plan de bit donné. Le second type d'arbre (type B) est similaire mais exclu les quatre enfants de la racine. De plus, dans SPIHT, les arbres sont définis de telle façon que chaque nœud ne possède aucun descendant (les feuilles) ou bien 4 descendants qui forment un groupe adjacent 2×2. Les coefficients de la sous-bande LL correspondant aux racines de l'arbre sont également groupés en coefficients 2×2 adjacents.

3.4.3 La norme JPEG2000 :

L'initiative JPEG-2000 [127] est démarrée pour la réalisation d'un nouveau système de codage en utilisant les meilleures techniques de compression basées sur la théorie des ondelettes. Cette norme est capable de travailler avec ou sans perte. Son architecture est choisie pour pouvoir utiliser ce système dans un grand nombre d'applications [128]. La figure 3.4 présente le schéma bloc d'un codeur JPEG2000.

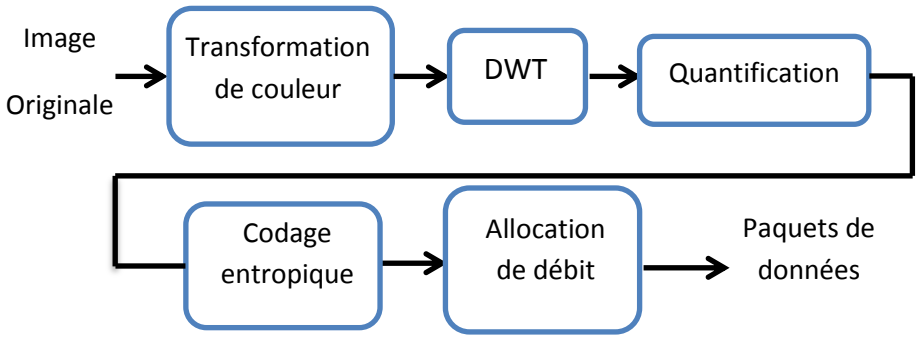


Figure 3.4 : Schéma typique d'un codeur JPEG 2000

La chaîne de codage JPEG 2000, commence par un prétraitement de l'image originale. L'image de départ est divisée en petites images appelées tiles (pour des raisons d'usage de mémoire) qui sont ensuite codés de manière totalement indépendants mais réunis finalement dans un seul et même code stream (au moment de l'allocation de débit). Un tile inclut tous les plans de couleur (RGB). Une transformation de couleur est appliquée à chaque tile. C'est une transformation optionnelle mais peut s'avérer intéressante. Elle permet d'obtenir une représentation de l'image dans un espace luminance/chrominance plus adaptée à la compression des données.

Chaque composante de chaque tile est décorrélée par une transformée séparable pour effectuer une décomposition dyadique en sous-bandes de fréquences. Les filtres autorisés sont soit la paire (9,7) irréversible de Daubechies [129] ou la paire (5,3) réversible de Le Gall [130]. En règle générale, la première permet des taux de compression plus élevés pour une qualité donnée, mais seule la seconde est utilisable pour compresser sans perte.

Lors du codage avec pertes, la précision sur les coefficients d'ondelettes est réduite par quantification scalaire uniforme. Cette étape permet, d'une part, de mettre à zéro les coefficients de faibles amplitudes (influant peu la qualité de l'image) mais fixe aussi le taux de compression minimal de l'image (i.e lorsque l'information quantifiée est intégralement incluse dans le codestream). Dans le cas d'un codage sans perte, le pas de quantification est égal à 1, ce qui signifie qu'aucune quantification n'est appliquée.

Après quantification, l'information est encore plus concentrée sur un nombre réduit de coefficients. JPEG2000 code les index de quantification obtenus précédemment grâce à un codeur arithmétique adaptatif avec contexte. Les coefficients quantifiés sont préalablement groupés, dans chaque sous-bande, en blocs rectangulaires (code-blocks) typiquement de taille 64×64 ou 32×32. Puis chaque code-block est codé plan de bits par plan de bits, en commençant par les bits de poids le plus fort. En réalité, dans chaque plan de bits, les bits sont d'abord séparés

en trois groupes, en fonction de leur voisinage, puis codés en trois passes (coding passes) successives.

Il est important de constater ici que le bit stream obtenu après un tel type de codage peut être tronqué à la fin de chacune des coding passes (il y a donc trois points de troncature possible par plan de bits) qui correspondent alors à des qualités progressives de chaque code-block. Cette propriété est la base de l'algorithme EBCOT (Embedded Block coding with optimized truncation) [131] et est largement exploitée par tous les types d'allocations de débits. Cette fonction peut varier fortement d'un algorithme de codage à un autre suivant les performances et fonctionnalités escomptées pour le codeur. Néanmoins tous les algorithmes d'allocation de débit ont pour but commun la création de paquets de données tels qu'ils sont définis dans la norme.

Chaque paquet correspond à un certain layer (généralement associé au concept de qualité) d'un niveau de résolution d'une composante de l'image. Il est constitué d'un en-tête identifiant son contenu et permettant un accès aléatoire rapide dans le *codestream*, ainsi que de données compressées obtenues par concaténation d'un certain nombre de *coding passes* de code-blocks d'un même niveau de résolution. Il est à noter que, pour obtenir des taux de compression élevés, les dernières *coding passes* d'un code-block sont souvent sautées. Ce dernier cas revient plus ou moins à changer le pas de quantification, et donc à diminuer la précision des coefficients dont les bits de poids les plus faibles ont été évincés. Enfin chaque paquet est ajouté au *codestream* à la suite d'un en-tête (regroupant tous les paramètres de codage) et suivant un ordre d'inclusion dépendant du type de progression désiré.

3.5 Compression hybride fractale-ondelette :

3.5.1 Etat de l'art :

Le codage d'une image par fractales est fondé sur des transformations locales qui opèrent sur des blocs de l'image. Les effets de blocs sont dans ce cas difficiles à masquer lorsque le taux de compression est élevé. Les autosimilarités sont largement présentes dans une décomposition multiresolution, d'où l'idée d'utiliser le codage par fractales pour définir les relations entre les différents niveaux d'une décomposition multirésolution de l'image. Plusieurs schémas basés sur l'hybridation des fractales et la transformation en ondelette discrète ont été proposés dans la littérature dans le but de réduire les effets de blocs dans les images reconstruites en utilisant l'ondelette de Haar ou d'autres ondelettes plus lisses. Nous décrivons ci-après les principaux travaux :

Dans [132] Pentland et Horowitz ont évoqué la possibilité de lier le formalisme mathématique des fractales avec la théorie des ondelettes. Dans cette approche, une décomposition en ondelette est appliquée sur l'image d'origine et les coefficients résultants sont codés en utilisant la

quantification vectorielle et le codage entropique. La partie fractale dans cette approche figure dans l'exploitation statistique des autosimilarités inter-échelles. L'une des premières applications de l'analyse multirésolution dans le codage fractale a été introduite par Baharav et al [70] dans le but de réduire la complexité du décodage. Cette approche est basée sur une analyse hiérarchique du décodage fractale. Elle est similaire à celle proposée par Monro et Dubridge dans le domaine spatial [133].

D'autres travaux cherchent à lier plus directement la transformation fractale à la transformation en ondelettes de Haar et de mettre en évidence les propriétés multirésolution des transformations contractantes dans le cas où les blocs source et destination sont des carrés [134][135]. Van de Wall [136] décrit la relation fractale-ondelette en exploitant le partitionnement quadtree. Une autre méthodologie proposée par Simon [137] consiste à calculer les coefficients de la transformation fractale à partir des coefficients de la représentation multirésolution de l'image.

En parallèle à ces travaux, Rinaldo et Calvagnon [138] ont proposé le premier codeur basé sur la théorie des fractales et la transformée en ondelettes, appelé "*Predictive Pyramid Coder*" (PPC). L'image originale est décomposée en *N* niveau. Chacune des trois sous bandes dans le niveau *N-1* sont codée par une approche fractale à partir des trois sous bandes dans le niveau *N*. Le processus continue ainsi jusqu'à atteindre la résolution 1 de l'image originale. Un autre codeur proposé par Davis [139], appelé "*Self Quantization of Subtree*" (SQS), combine la quantification scalaire optimisée et le codage par fractales avec les ondelettes. Les trois opérations fractales couramment utilisées (sous-échantillonnage, isométries et coefficient d'échelle) ont leur correspondance dans le domaine transformé.

Li et Kuo [140] ont également utilisé les fractales pour prédire les coefficients des sous bandes de niveau inférieur à partir de celles de niveau supérieur. Mais, contrairement à PPC et SQS, l'algorithme code le résidu de la prédiction fractale par un codeur plans de bits. Une extension proposé par Zhang et al [141] permet d'exploiter des blocs source et destination de tailles variables en employant le partitionnement quadtree.

Différentes ondelettes et leurs variantes ont été utilisées par la suite dans des schémas hybrides fractale-ondelette pour améliorer les performances de compression. Le schéma proposé par Kim [142] est basé sur l'algorithme EZW [125] et le codage par fractales. Iano et al [143] ont utilisé la méthode de Fisher [4] pour coder la sous bande basse résolution et l'algorithme SPIHT [126] pour coder les sous bandes de détails. Duraisamy et al [144] ont combiné la vitesse de la transformée en ondelettes avec la méthode fractale itéré [145] pour réduire le temps d'exécution.

3.5.2 Schéma proposé : codeur hybride fractale-ondelette sans perte :

3.5.2.1 Structure générale :

Nous avons proposé [146] un schéma de compression sans perte d'information en combinant la théorie des ondelettes, le formalisme mathématiques des fractales et le codage de Huffman [15]. Dans ce schéma nous avons mis à profit les avantages de chaque méthode: la technique fractale pour son taux de compression, la transformée en ondelettes pour sa rapidité, et le codage de Huffman pour la qualité de l'image reconstruite.

Dans le processus de compression, l'image est d'abord décomposée par la transformation en ondelette discrète (DWT). Les sous bandes de détails représentant les hautes fréquences sont codées par un système de fonction itérés locales (LIFS), selon l'approche du Jacquin. La recherche des autosimilarités s'effectue entre chaque paire de sous bandes adjacentes de même direction (horizontale, verticale, et diagonale). La reconstruction de l'image est réalisée en utilisant l'algorithme de décodage fractal itératif et la transformée en ondelettes inverse (IDWT). Le résidu de la prédiction fractale sera ensuite codé par l'algorithme de Huffman. Le processus de compression est décrit sur la figure 3.5. La décompression se fait par transformations inverses.

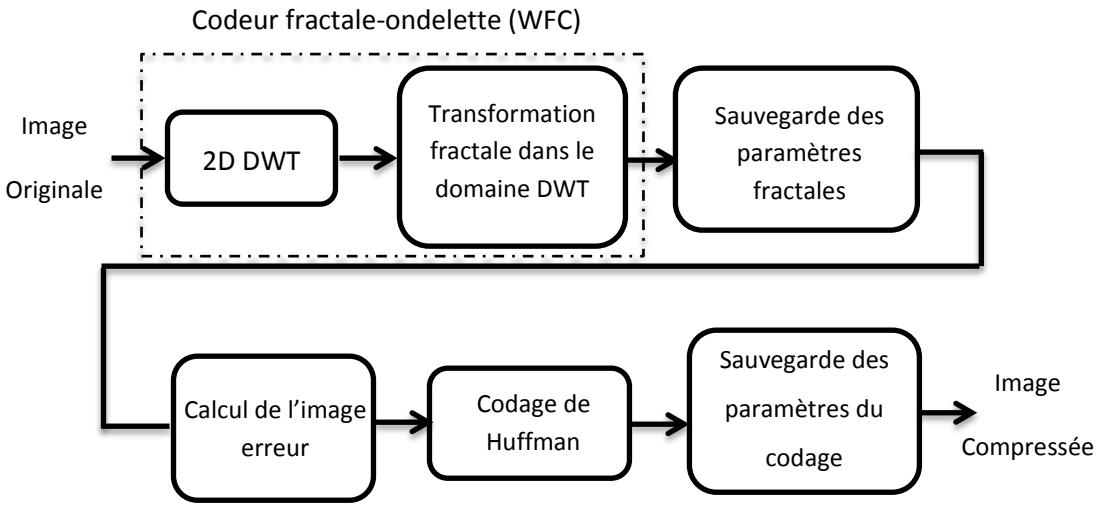


Figure 3.5 : Schéma bloc du codeur hybride fractal-ondelette sans perte.

3.5.2.2 Processus de compression :

La compression d'images par fractales dans le domaine des ondelettes est considérée comme la prédiction de l'ensemble des coefficients d'ondelette des sous bandes de niveau inférieur à partir des sous bandes de niveau plus élevé [140]. L'erreur de prédiction après un processus complet de compression/décompréssion est codée par l'algorithme de Huffman.

3.5.2.2.1 Structure d'arbre:

L'image à coder est décomposée en une sous bande de fréquence lissée et différentes sous bandes de détails, en utilisant une transformation en ondelette discrète. Chaque sous bande contient des structures traduisant des variations dans une direction privilégiée du plan spatial.

Dans la représentation de l'image ondelettes, chaque coefficient peut être considéré en tant qu'ayant quatre descendants à la même localisation spatiale dans la sous-bande de niveau inférieur. Par exemple, Dans une décomposition en trois niveaux d'une image, on trouve 1 sous bande de coefficients d'approximations LL3 et 3 sous bandes de HL : HL1, HL2, HL3, 3 sous bandes de LH : LH1, LH2, LH3 et 3 sous bandes de HH : HH1, HH2, HH3. Dans ces sous bandes de même nature, les coefficients descendent les uns des autres. Prenons les coefficients HH, un coefficient HH dans le plus haut niveau de décomposition HH3 possède 4 descendants dans HH2. Chaque coefficient de HH2 possède 4 descendants dans HH1. On aura donc trois sous arbre associés à chaque direction (horizontale, verticale et diagonale). La recherche des similarités se fait donc au sein de ces différents arbres à l'aide de la transformation fractale. La figure 3.6 montre la procédure d'appariement dans le domaine des ondelettes.

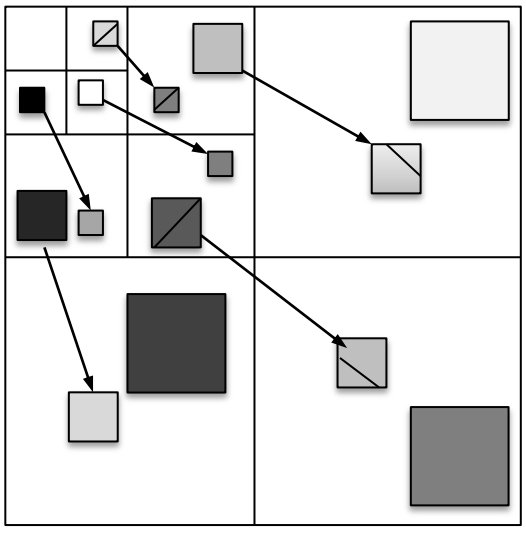


Figure 3.6 : La procédure d'appariement au sein des sous bandes de détails.

3.5.2.2.2 Transformation fractale dans le domaine de la transformée en ondelette :

La transformation contractante spatiale a une analogie dans le domaine des ondelettes. La procédure d'appariement s'effectue entre un sous arbre source d'un niveau *n* de décomposition et un sous arbre destination de niveau *n*-1 [147] :

La transformation géométrique : englobe l'opération du sous échantillonnage et permet de ramener la taille du sous arbre source D_n à celle du sous arbre destination R_{n-1}. Par la suite, l'opération d'isométrie s'effectue au sein de chacune des sous bandes de détails.

La transformation massique : elle consiste à multiplier le sous arbre sous échantillonné par un facteur d'échelle. Dans ce cas, La constante additive n'est pas nécessaire car elle est déjà contenue dans les coefficients de la fonction d'échelle.
Chaque sous arbre destination R_{n-1} est associer à un sous arbre source D_n selon l'équation [142][147]:

$$W(D_n) = \alpha * O(S(D_n)) = R_{n-1} \tag{3.23}$$

Où *S* désigne la transformation spatiale, *O* est l'opération d'isométrie et α est le facteur d'échelle. La recherche des similarités au sein du dictionnaire des sous arbre source, se fait par minimisation erreur quadratique moyenne. Soit $X = (x_1,\ldots,x_k)$ l'ensemble ordonné des coefficients de l'arbre destination et $Y = (y_1,\ldots,y_k)$ l'ensemble ordonné des coefficients sous échantillonné de l'arbre source. L'erreur quadratique moyenne est donnée par l'équation:

$$MSE = \left\| R_{n-1} - W(D_n) \right\|_2^2 = \sum_{i=1}^{k} (x_i - \alpha * y_i)^2 \tag{3.24}$$

La valeur optimale de α est obtenue en minimisant l'erreur quadratique moyenne:

$$\frac{d(MSE)}{d\alpha} = 0$$

Ce qui donne :

$$\alpha = \frac{\sum_{t=1}^{k} x_t * y_t}{\sum_{t=1}^{k} y_t^{\,2}} \tag{3.25}$$

Les paramètres à sauvegarder sont la position du sous arbre source, l'index de l'isométrie et le facteur d'échelle. La figure 3.7 représente la procédure d'approximation fractale-ondelette.

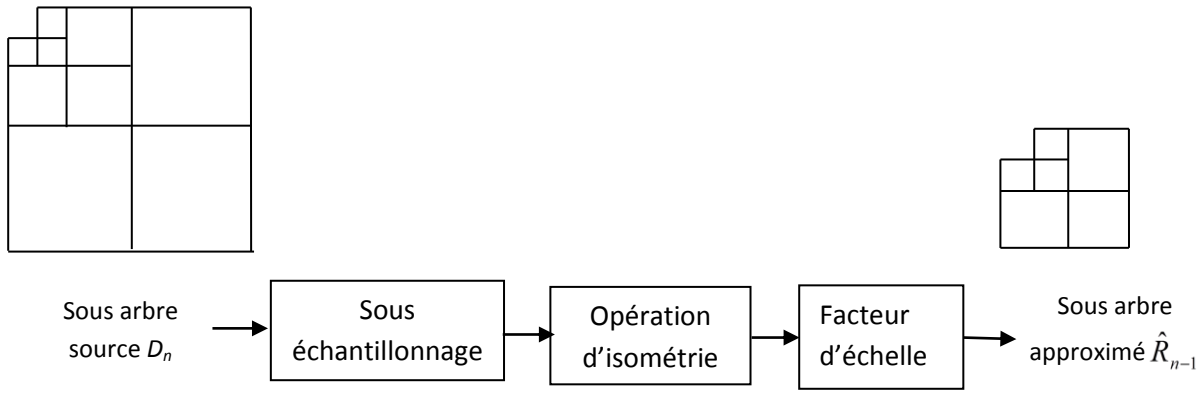


Figure 3.7 : La transformation fractale-ondelette.

L'erreur de prédiction fractale après un processus complet de compression/décompression est codée par l'algorithme de Huffman [52]. Les paramètres du codage de Huffman sont par la suite sauvegardés.

3.6 Conclusion :

La transformée en ondelettes est très utilisée dans les codeurs récents car elle possède des propriétés intéressantes. L'introduction des ondelettes dans le codage par fractales permet de réduire les effets de blocs dans les images reconstruites et d'accélérer la phase du codage.

Le schéma de compression sans perte que nous avons proposé est fondé sur la théorie des ondelettes, le formalisme mathématiques des fractales et le codage de Huffman. Ce codeur permet de profiter des avantages de chaque méthode employée : le taux de compression issu de la compression fractale, la qualité du décodage donné par l'algorithme de Huffman et la rapidité des ondelettes.

CHAPITRE 4

Chapitre 4 : Résultats et Discussions

Le but de ce chapitre est de présenter les résultats obtenus avec les algorithmes de compression que nous avons développés dans les chapitres précédents.

Ce chapitre est décomposé en quatre parties : La première section abordée présente les résultats de la compression fractale. Dans un deuxième temps nous exposerons les résultats obtenus par l'algorithme de compression hybride fractale-ondelette. Nous évaluerons ensuite la compression sans perte en utilisant le schéma proposé. Enfin, nous examinerons l'application des méthodes fractales et ondelettes, sans ou avec pertes, sur des images radiographiques des défauts de soudure.

Les tests ont tous été réalisés dans les mêmes conditions. Un ordinateur équipé d'un processeur i7 cadencé à 3,4 GHz avec 8 Go de mémoire RAM.

4.1 Mesures de performances d'une méthode de compression [12][148]:

4.1.1 Mesure de compression :

Ce critère est d'importance puisqu'il est directement dépendant de la technique de compression utilisée. L'évaluation de la compression peut être donnée par plusieurs paramètres, parmi eux, on cite :

$$Rapport\ de\ Compression = R_c = \frac{Taille\ des\ données\ Avant\ la\ compression}{Taille\ des\ données\ Aprés\ La\ compression} \tag{4.1}$$

$$Taux\ de\ Compression\ = CR = \left(1 - \frac{1}{R_c}\right) \times 100\%\ ,exprimé\ en\ pourcentage \tag{4.2}$$

$$Débit = \frac{1}{R_c} \times profondeur\ de\ l'image, exprimé\ en\ bit\ par\ pixel\ (bpp) \tag{4.3}$$

4.1.2 Mesure de la qualité d'image reconstruite :

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. C'est l'un des critères les plus importants pour comparer des algorithmes de compression d'images.

En fait, il faudrait plutôt parler de la fidélité d'une image compressée, c'est-à-dire sa ressemblance comparée à celle de l'image originale, plutôt que de seulement considérer la qualité perçue d'une image donnée sans aucune référence à l'image de départ. Généralement quand on utilisera le terme qualité d'une image, c'est à la fidélité d'une image compressée qu'on fera référence. La qualité de l'image peut se mesurer de deux manières :

Une première méthode pour mesurer la qualité d'une image reconstruite est évidemment de faire appel au système de vision humain (SVH). Il est possible de construire des expériences psycho visuelles où un certain nombre de cobayes humains donnent leurs appréciations d'images compressées selon des critères déterminés à l'avance. Il est par exemple recommandé de prendre une échelle qualitative classant les images en 5 catégories: excellente, bonne, correcte, pauvre ou de qualité médiocre. On peut ensuite donner une note à chaque image selon la moyenne des opinions exprimés. Même si c'est la méthode la plus fiable pour mesurer la qualité des images puisque que c'est la seule qui fait directement appel au système de vision humain, elle a de sévères lacunes : en premier lieu, la difficulté de contrôler les conditions de l'expérience et ensuite le temps astronomique requis pour effectuer ce genre d'expérience.

Ce qu'on aimerait donc posséder, c'est un modèle mathématique fournissant une métrique qui mesure la distance entre deux images, c'est-à-dire la différence plus ou moins grande perçue entre deux images. Pour les mesures de distorsion, on utilise l'erreur quadratique moyenne *EQM* entre l'image originale et l'image compressée. Cette métrique est la plus répandue dans la littérature pour la mesure de la fidélité entre deux images. On mesure la métrique *EQM* tout simplement comme une erreur L^2 sur les pixels de l'image. Donc, si I_1, I_2 sont deux images couleurs de dimension $N \times M$ avec les éléments de chaque composante de couleur RVB prenant des valeurs entières entre 0 et 255 (image 24 bits), alors l'erreur *EQM* est donnée par :

$$EQM = \frac{1}{3NM} \sum_{i=1}^{N} \sum_{j=1}^{M} \left(I_1(i, j) - I_2(i, j) \right)^2 \tag{4.4}$$

L'erreur *EQM* est nulle si et seulement si les deux images à comparer sont identiques. Notons que la valeur maximale de l'erreur, 255, est donnée quand on compare une image complètement noire à une image complètement blanche.

Une autre métrique très utilisée pour évaluer la qualité objective de l'image est le PSNR (Peak Signal to Noise Ratio) définit par :

$$PSNR(dB) = 10\log_{10}\frac{Vmax^2}{EQM} \tag{4.5}$$

Où Vmax est l'intensité maximale de l'image (255 pour une représentation 8bpp)

Ces mesures sont très utilisées et très simples à calculer. Cependant elles ne tiennent pas compte des caractéristiques du système de vision humain, mais mesurent seulement la différence pixel par pixel entre deux images. Si par exemple tous les pixels d'une image étaient translatés, La distorsion serait très élevée, alors que la qualité visuelle serait parfaitement bonne. De plus, elles sont des mesures globales sur toute l'image qui ignorent les variations locales.

4.1.3 Mesure de la complexité de l'algorithme :

La complexité calculatoire d'un algorithme ou d'une méthode de compression 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.

4.2 Evaluation de la compression fractale :

La compression fractale d'une image consiste à trouver les similitudes entre les différentes parties de celle-ci moyennant certaines transformations. Ces dernières portent sur la géométrie et les modifications des niveaux de gris des parties de l'image considérées. Nous nous intéressons dans cette partie à l'axe du partitionnement de l'image et l'accélération du codage fractale.

4.2.1 Rôle du partitionnement :

La partition intervient sur la qualité de l'image dans le sens où si elle est bien adaptée à l'image, un nombre élevé de couples destination/source seront susceptibles de correspondre avec une très bonne qualité de reconstruction. Le partitionnement intervient également sur la taille des données compressées.

Pour montrer l'utilité du partitionnement de l'image dans le schéma de compression fractale nous avons choisi trois méthodes opérant sur différents modèles de partitionnement : la méthode de Jacquin fondée sur le partitionnement carré, elle est la plus répandue, la méthode de Fisher basée sur un partitionnement quadtree qui est une extension de la première et la méthode de compression employant la triangulation de Delaunay. Nous avons met en application la compression fractale par une série d'images numériques fixes en 256 niveaux de gris de taille 256×256 au format BMP (Lena, boats, cameraman). La figure 4.1 présente les images de test. Pour évaluer la compression nous avons utilisé deux critères: le taux de compression et la qualité visuelle de l'image décodée en faisant appel au système de vision humain (SVH).

Figure 4.1 : Images de test de taille 256×256.

4.2.1.1 Evaluation de la méthode de Jacquin :

La première méthode que nous avons testée est la méthode de Jacquin, la mise en œuvre de la méthode consiste à définir les ensembles des blocs destination et source, les transformations contractantes qui lient les blocs ainsi que la mesure d'erreur pour chaque couple destination/source.

L'image originale est tout d'abord partitionnée en blocs destination R_i de taille B×B (non chevauchés). On définira ensuite un ensemble de blocs source D_{init}, qui contiendra tous les blocs d'image possibles et qui peuvent être extraites de l'image originale. Cet ensemble peut être obtenu en glissant une fenêtre de taille D×D (D = 2B) à travers l'image originale. La fenêtre est localisée la première fois au point (0,0) de l'image, puis elle est déplacée d'un pas p_x de pixels horizontalement et d'un pas p_y pixels verticalement. Une partition sans chevauchement a été choisie afin d'accélérer le temps de codage.

La construction de l'ensemble des transformations qui représente le code fractal peut être effectuée en deux étapes différentes : La première étape est la contraction spatiale, elle est équivalente à la sélection d'un bloc source D_j de l'image de taille D×D, qui sera transformé en bloc de taille B×B. La deuxième étape consiste à sélectionner un traitement propre pour le bloc contracté, en lui trouvant une transformation massique appropriée. Par la suite une isométrie parmi huit qui minimise la distorsion entre le bloc source transformé et le bloc destination sera sélectionnée. Les paramètres à sauvegarder sont la position du bloc source, l'index d'isométrie et les coefficients de la transformation massique (s_i et o_i).

La décompression de l'image consiste à appliquer un certain nombre de fois les transformations extraites dans la partie compression. Nous remplaçons le contenu de chaque bloc destination par le bloc image situé aux coordonnées du bloc source correspondant, après application de la transformation liant le bloc destination et le bloc source, calculée lors de la compression. Au fur et à mesure que les transformations seront appliquées à l'image, la suite des images constituées converge vers l'image compressée. L'image initiale peut être une image quelconque. La Figure 4.2 illustre la convergence de l'image de Lena. On remarque à

travers nos tests que l'image est stable après 6 itérations, elle évolue peu après 7 à 8 itérations. Le nombre d'itérations peut donc être fixé.

Les algorithmes de compression et décompression fractale selon l'approche de Jacquin sont reproduits ci-dessous :

Algorithme 4.1 : Algorithme de codage (Méthode de Jacquin)

```
Début
    Créer l'ensemble de blocs source
    Créer l'ensemble de blocs destination
    Pour chaque bloc destination
        Pour chaque bloc source
            Appliquer la réduction du bloc source vers le bloc destination
            Appliquer les isométries
            Ajuster la luminance et le contraste des pixels du bloc source déformé
            Calculer la distorsion entre le bloc source transformé et le bloc destination
             Si la distorsion est minimale
                 Alors Sauvegarder les modifications effectuées
             Fin Si
        Fin Pour
    Fin Pour
Fin
```

Algorithme 4.2 : Algorithme de décodage (Méthode de Jacquin)

```
Début
    Prendre une image initiale quelconque.
    Itérer
        Pour chaque bloc destination de l'image
            Lire le numéro du bloc source à manipuler
            Appliquer la transformation fractale à ce bloc
            Insérer le résultat dans l'image
        Fin pour
        Prendre l'image résultante comme étant image initiale
    Fin Itérer
Fin
```

Figure 4.2 : Décodage itératif de l'image de Lena.

Le taux de compression et la qualité finale de l'image décompressée par rapport à son apparence originale avant compression dépend de la taille B des blocs destination. Les figures 4.3, 4.4 et 4.5 représentent les images de test restituées avec différentes tailles des blocs destination (B). Les variations du taux de compression en fonction de la taille des blocs sont présentées sur les figures 4.16.

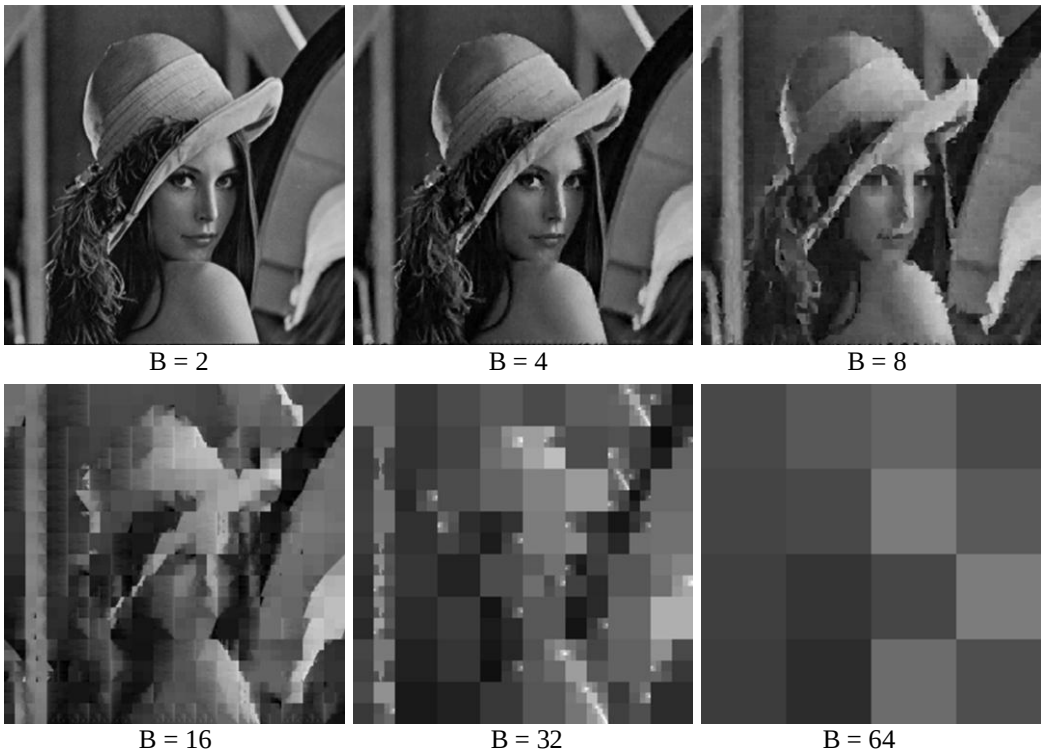


Figure 4.3 : Reconstruction de l'image Lena en utilisant un partitionnement carré.

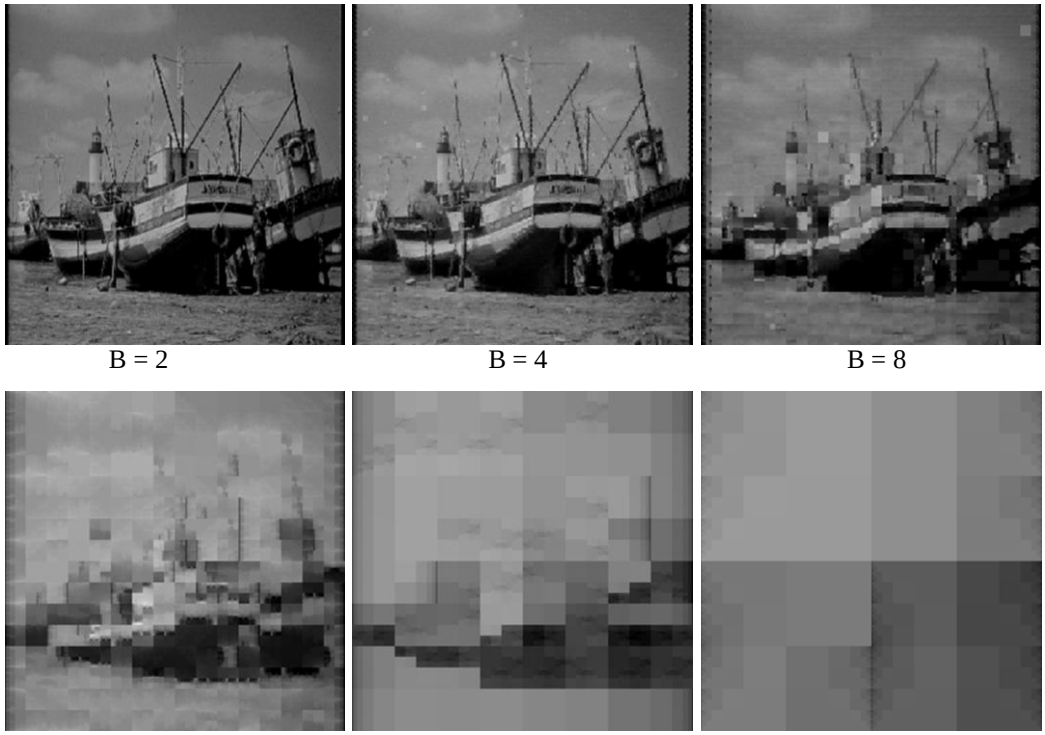


Figure 4.4 : Reconstruction de l'image boats en utilisant un partitionnement carré.

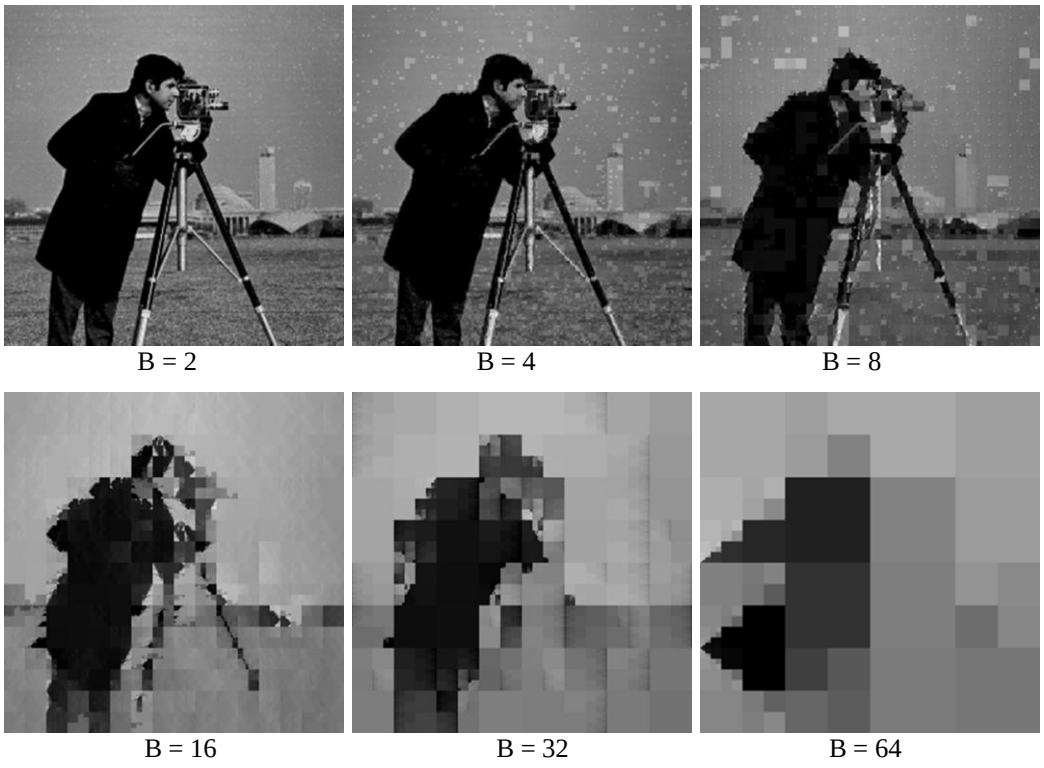


Figure 4.5 : Reconstruction de l'image cameraman en utilisant un partitionnement carré.

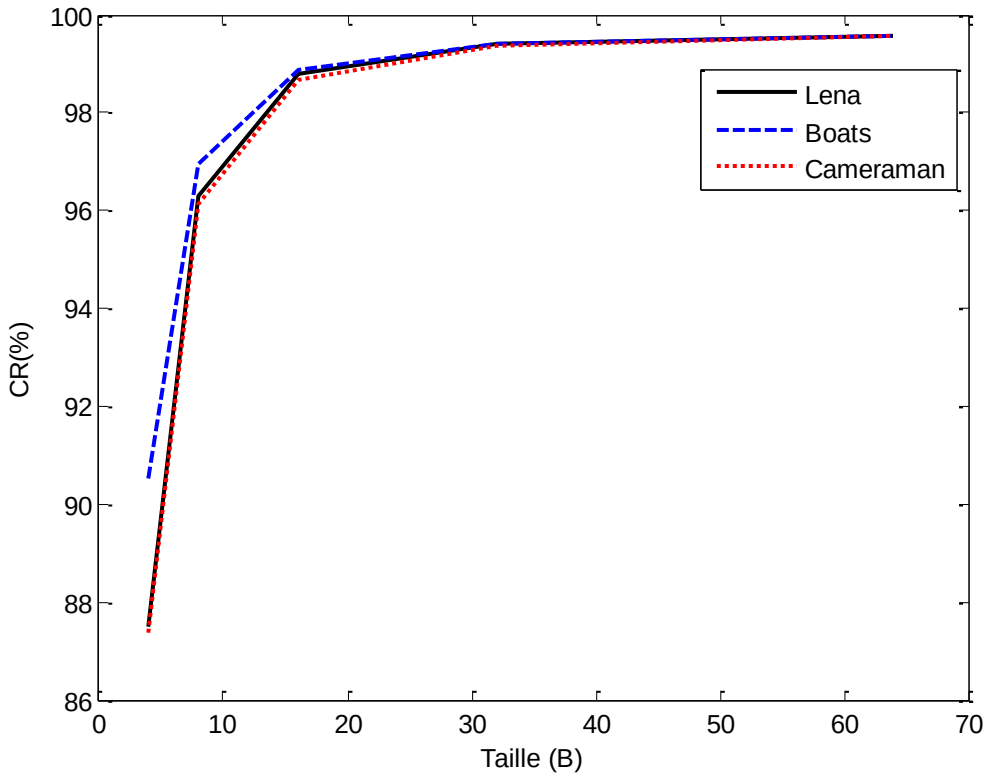


Figure 4.6 : Variations du taux de compression en fonction de la taille des blocs destination.

Nous constatons que la taille des blocs destination considérée lors du codage influe sur la qualité du décodage. Plus les blocs (destination/source) qu'on doit faire correspondre sont de petite taille, plus il est aisé d'avoir une bonne reconstruction, et ce avec des transformations simples. Nous remarquons que les images obtenues en utilisant une taille des blocs destination de 2×2 présente le meilleur aspect visuel. Dans le cas 64×64, l'image est totalement dégradée.

Dans le graphe de la figure 4.6, nous remarquons que le taux de compression est proportionnel à la taille des blocs choisie. En effet le taux de compression augmente lorsque le nombre de blocs source diminue, c'est-à-dire lorsque la taille des blocs augmente. Le meilleur taux de compression est obtenu en fixant la taille des blocs à 64×64. La valeur la plus faible est obtenue en fixant la taille à 2×2.

4.2.1.2 Evaluation de la méthode de Fisher :

La deuxième méthode testée est la méthode de Fisher. Le partitionnement de l'image en blocs destination est effectué en découpant l'image en quatre blocs de même taille appelé quadrants. Chaque quadrant est découpé à son tour en quatre sous-quadrants jusqu'à que la taille du quadrant est égale à la taille maximale B_{max} fixée par l'utilisateur. A partir de cette taille les recherches des autosimilarités s'effectuent en parallèle avec le partitionnement. La profondeur maximale P du quadtree est fixée à l'avance ce qui impose la taille minimale B_{min}.

La construction des blocs source s'effectue en plaçant une fenêtre de taille D, qui peut prendre les valeurs suivantes : $2B_{max}$, B_{max}, $B_{max}/2$,…, $2B_{min}$. Elle est déplacée d'une position par un pas de chevauchement de P_x horizontalement et de P_y pixels verticalement.

Chaque bloc destination est comparé aux blocs source. Le bloc source est décimé (réduit à la taille du bloc destination en remplaçant les groupes de 2×2 pixels par leur moyenne), ajusté et orienté en huit isométries pour approximer le bloc destination courant. La distorsion entre ce bloc source et le bloc destination courant sera calculée. Si la distorsion minimale est inférieure à une tolérance fixée par l'utilisateur, on considère que le bloc source est similaire au bloc destination et les paramètres de la transformation sont stockés. Si l'erreur minimale est supérieure au seuil alors le partitionnement précédent se poursuit jusqu'à la taille minimale B_{min} des blocs destination (la profondeur maximale P est atteinte). Dans ce cas la meilleure distorsion sera prise même si elle est supérieure au seuil préfixé.

Algorithme 4.3 : Algorithme de codage **(**Méthode Fisher)

Début

Construire le dictionnaire de blocs source D.

Fixer un seuil d'erreur (Tolérance) T.

Créer un partitionnement régulier large de blocs destination R

***Pour** chaque bloc destination R_i*

***Pour** chaque bloc source D_j*

Trouver la transformation fractale wi

***Si** la distance $d(wi(D_j), R_i)$ est inférieure au seuil T.*

Alors

Encoder la référence du bloc source (les coordonnées et la taille du bloc) et les coefficients de la transformation fractale.

Sinon

***Si** la taille du bloc destination est supérieure à la limite*

Subdiviser le bloc destination en quatre sous-blocs et les ajouter dans l'ensemble des blocs destination à traiter.

Fin si

Fin si

Stocker les paramètres du codage fractale.

Fin Pour

Fin Pour

Fin

Dans les simulations, il existe plusieurs paramètres qui peuvent être ajustés pour produire des résultats différents aux points de vue taux de compression et qualité visuelle de l'image décodée. Certains paramètres sont bénéfiques pour un de ces points mais en même temps nocif pour l'autre. Dans ces situations il faut faire des compromis. Nous fixons tout d'abord le pas de chevauchement $p_x = p_y = 1$ pour avoir une meilleure qualité d'image reconstruite. Nous faisons varier ensuite la profondeur maximale P du quadtree et la tolérance T pour montrer leurs influences sur la qualité d'images et sur le taux de compression. Les Figure 4.7, 4.8 et 4.9 présentent les images reconstruites pour différentes valeurs de la profondeur et la tolérance. Les variations du taux de compression en fonction de la profondeur et la tolérance sont présentées sur les figures 4.10 et 4.11 respectivement.

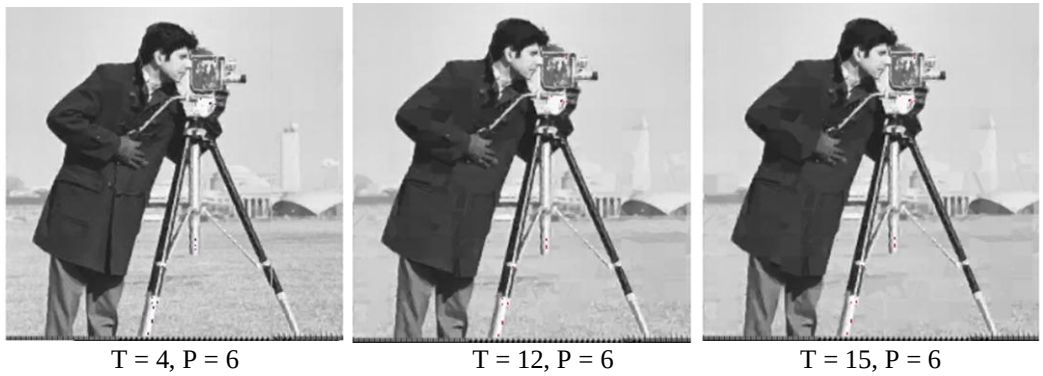
Figure 4.7 : Reconstruction de l'image Lena en utilisant le partitionnement quadtree.

P = 4, T = 8 P = 5, T = 8 P = 8, T =8

T = 4, P = 6 T =12, P = 6 T = 15, P = 6

Figure 4.8 : Reconstruction de l'image boats en utilisant le partitionnement quadtree.

P = 4, T = 8 P = 5, T = 8 P = 8, T = 8



T = 4, P = 6 T = 12, P = 6 T = 15, P = 6

Figure 4.9 : Reconstruction de l'image cameraman en utilisant le partitionnement quadtree.

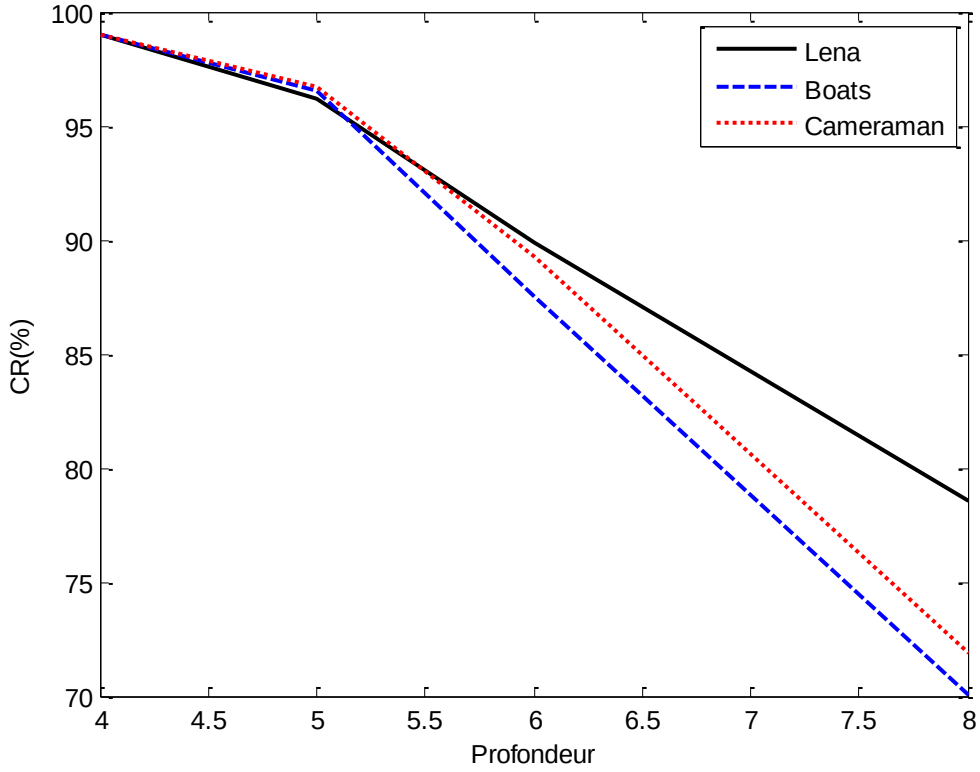


Figure 4.10 : Variations du taux de compression en fonction de la profondeur du quadtree.

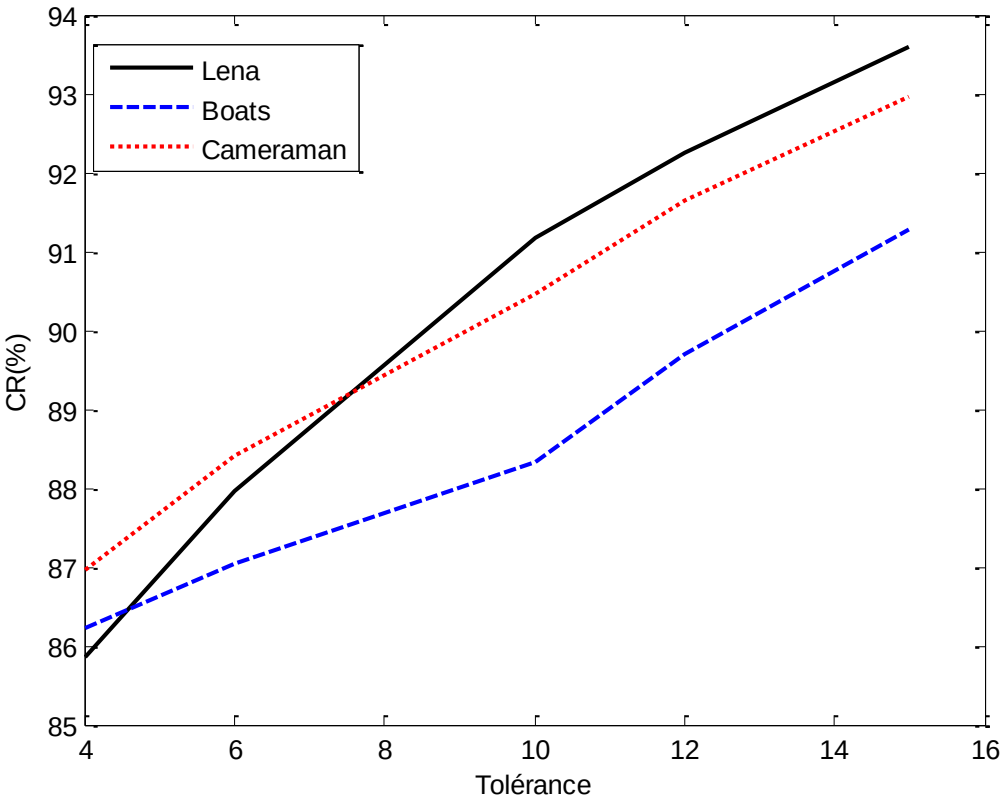


Figure 4.11 : Variations du taux de compression en fonction de la tolérance.

D’après les résultats obtenus nous pouvons constater que:

- La qualité visuelle de l’image reconstruite s’améliore lorsque la profondeur du quadtree augmente, car cela permet de couvrir les détails de l’image avec des petits blocs.
- Lorsque la tolérance, qui représente un seuil relatif à la mesure d’erreur, diminue la qualité visuelle de l’image est meilleure.
- Lorsque la profondeur augmente le nombre de blocs destination augmente, et le taux de compression va diminuer par conséquent.
- Lorsque la tolérance augmente le nombre de partition quadtree diminue ce qui permet d’augmenter le taux de compression.

La compression fractale en utilisant le partitionnement quadtree est efficace du point de vue qualité des images décompressées, mais il faut bien choisir les paramètres du codage fractale.

4.2.1.3 Evaluation de la compression fractale basée sur la triangulation de Delaunay :

Nous présentons dans cette partie les résultats du codage fractale basé la triangulation de Delaunay. Le partitionnement en blocs source est réalisé avec une triangulation régulière de l’image avec un pas déterminé. La partition destination en triangles de Delaunay est calculée avec un nombre des germes de Voronoi initial, un processus de division-fusion a été adopté. Les paramètres du codage fractale sont présentés dans le tableau 4.1.

Tableau 4.1 : Paramètre du codage fractale en utilisant la triangulation de Delaunay

PARAMETRES	**VALEURS**
Pas des triangles	16
Nombre de division	2
Nombre d’isométrie	5
Nombre de points	50

Les étapes de la création de la partition en triangles de Delaunay sont résumées dans l'algorithme suivant :

Algorithme 4.4 : Génération de la partition destination

```
Début
    Mettre un nombre de points au hasard de l'image
    Calculer le diagramme de Voronoï de ces points
    Calculer la triangulation de Delaunay
    % Division
    Pour toutes les divisions désirées
        Pour tous les triangles de Delaunay
            Si le triangle n'est pas homogène
                Alors ajouter un germe de voronoï (Centre de gravité du triangle)
            Fin si
        Fin pour
        Calculer le diagramme de Voronoï de ces points
        Calculer la triangulation de Delaunay
    Fin pour
    % Fusion
    Pour tous les germes de Voronoï
        Pour tous les triangles de Delaunay associé au germe
            Calculer la moyenne des pixels du triangle
        Fin pour
        Si les moyennes sont égales
            Alors supprimer le germe de Voronoï courant
        Fin si
    Fin pour
    Retourner triangles de Delaunay
Fin
```

La figure 4.12 montre le partitionnement de Delaunay des trois images de test, les résultats du décodage et les images de différence obtenus à un taux de compression égale à 32%.

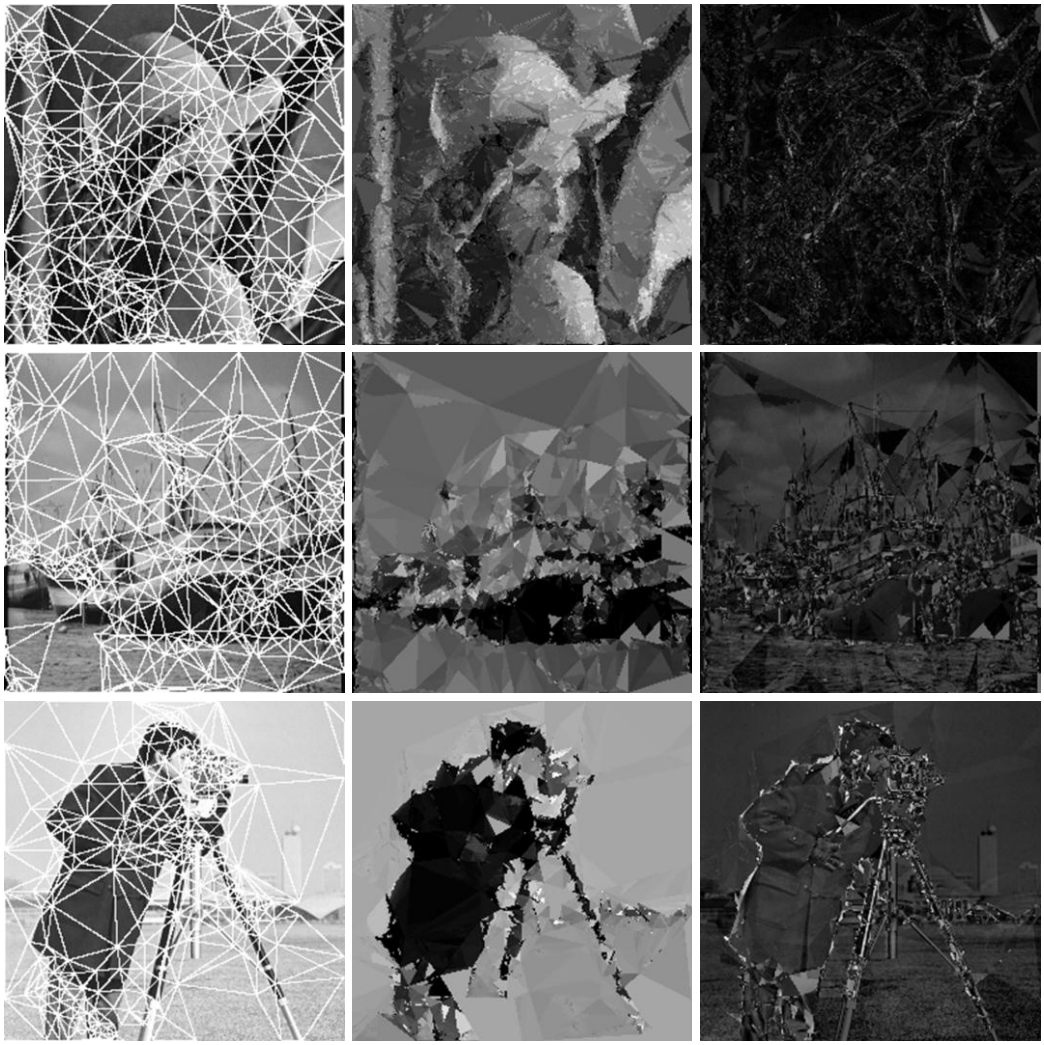


Figure 4.12 : Partition en triangles de Delaunay des images originales, images reconstruites, images de différence.

D'après les résultats de compression, nous remarquons que la qualité des images reconstruites est dégradée car la partition des triangles de Delaunay ne recouvre pas toute l'image. Pour améliorer la qualité des images nous devons augmenter le nombre de germes ou le nombre de division, ce qui nécessite un temps de compression énorme, en plus le taux de compression va diminuer au-dessous de 32%, ce qui n'est pas avantageux pour une méthode de compression avec pertes. Cette méthode de partitionnement est couteuse au niveau du temps de calcul et n'offre pas une amélioration significative au niveau débit/distorsion.

Nous constatons qu'il n'est pas toujours évident de caractériser et trouver ce qu'est un partitionnement bien adapté à l'image a priori. Pour cette raison, une solution peut être de choisir un partitionnement plus fin, ce qui correspond dans le cas d'un partitionnement en blocs carrés, à passer par exemple de blocs image de taille 4×4 pixels à des blocs de taille 2×2 pixels.

4.2.2 Evaluation de la complexité de l'algorithme de compression fractale :

Le temps accordé à la recherche de couples destination/source, ainsi que la stratégie de parcours de ces couples influent sur l'image compressée. En observant les deux algorithmes compression / décompression fractale nous constatons le décalage important en coût de calcul qu'il peut y avoir entre les deux. En effet, la phase de compression peut rapidement tendre vers le parcours exhaustif de couples destination/source. Au contraire, la décompression est immédiate, nous ne faisons qu'appliquer les transformations trouvées lors de la compression.

Dans le cas où nous parcourons tous les couples destination/source (recherche exhaustive), le nombre de blocs destination pour une image de taille $M \times M$ pixels, partitionnée en $B \times B$ blocs, est égale à:

$$N_r = \left(\frac{M}{B}\right)^2$$

Le nombre de blocs source avec un pas de chevauchement p_x et 8 isométrie est égale à :

$$N_d = 8\left(\frac{M}{p_x}\right)^2$$

Puisque chaque correspondance destination/source nécessite B^2 multiplications et B^2 additions. Le nombre total des multiplications ou d'addition est donné par :

$$N_r N_d B^2 = 8\frac{M^4}{p_x^2}$$

$$\text{i.e. } 8\left(\frac{M}{p_x}\right)^2 \text{ par pixel} \qquad (4.6)$$

Le tableau 4.2 donne les PSNRs en dB, les rapports de compression, les nombres d'opérations par pixel (NBR OPT) et les temps de compression (CT) en seconde pour l'image de Lena (256×256).

Tableau 4.2 : Résultats numériques de la compression fractale de l'image de Lena

TAILLE (B)	PSNR (dB)	R_C	NBR OPT	CT (s)
4	23.72	7.98	16384	764.95
8	20.83	26.90	4096	49.85
16	18.70	84.00	1024	3.41
32	17.28	170	256	0.29
64	15.27	247.64	64	0.068

Nous remarquons que la taille des blocs destination influe sur le temps de calcul, la complexité diminue en choisissons des blocs de grande taille mais on perd en précision.

Cette énorme complexité de calcul montre clairement le besoin d'accélérer la vitesse d'exécution. Deux solutions peuvent être envisagées, pour accélérer la phase du codage fractale, une première solution consiste à limiter la recherche des blocs source dans une plage de blocs au voisinage du bloc destination courant. Dans le cas où la taille des blocs destination est égale à 2, une région des pixels adjacents de 6×6 pixels est choisie. L'application de cette méthode sur l'image de Lena (256×256) a permis de réduire le temps de compression considérablement mais avec une qualité d'image reconstruite dégradée. Une autre solution consiste à définir un seuil relatif à la mesure d'erreur pour chaque couple destination/source. Le seuil permet de dire si la correspondance entre deux blocs destination/source est bonne ou non. Pour chaque bloc destination, nous devons trouver au moins un bloc source qui lui corresponde. L'algorithme s'arrête lorsque l'erreur de reconstruction atteint le seuil choisi. Nous avons testé cette approche sur l'image de Lena avec une taille des blocs destination égale à 2 et une valeur du seuil égale à 150. Nous constatons que le temps d'exécution diminue mais avec une qualité de reconstruction moins bonne. Le tableau 4.3 résume les résultats de compression fractale en utilisant ces deux méthodes d'accélération.

Tableau 4.3 : Accélération de la phase du codage fractale pour l'image de Lena avec des blocs destination de 2×2 pixels

Compression fractale avec recherche exhaustive			*Compression fractale avec recherche au voisinage (6×6 pixels)*			*Compression fractale avec seuil (T=150)*		
PSNR (dB)	R_c	CT (s)	PSNR (dB)	R_c	CT (s)	PSNR (dB)	R_c	CT (s)
27.70	**1.89**	**11587**	**26.98**	**1.89**	**35.52**	**25.88**	**1.89**	**19.34**

4.2.3 Compression fractale des images couleurs :

Pour les images couleurs, chaque pixel est caractérisé par 3 intensités lumineuses, celles des canaux rouge, vert et bleu (codage RVB), définissant des images 3×8=24 bits. Le codage RVB parfois noté RGB (Red, Green, Blue) est aussi le codage utilisé par la télévision ou les écrans d'ordinateur pour reproduire les couleurs. En effet, d'après les principes de la colorimétrie Il faut 3 composantes indépendantes pour avoir un espace de couleur complet. Ces trois composantes RVB présentent une redondance entre elles. C'est pourquoi la vidéo analogique couleur est souvent transportée dans un autre mode de représentation appelé YCrCb. Y représente la luminance et CrCb les composantes Rouge et Bleu sans la luminance et on les appelle alors les chrominances. Le système visuel humain est moins sensible aux nuances de couleurs qu'aux différences d'intensités lumineuses, d'où l'intérêt du codage YCrCb au moment de la compression.

Pour passer d'un espace RVB a l'espace YCrCb et inversement nous disposons d'expressions mathématiques assez simples :

$$\begin{aligned} Y &= 0.257R + 0.504V + 0.098B + 16 \\ Cr &= 0.439R - 0.368V - 0.071B + 128 \\ Cb &= -0.148R - 0.291V + 0.439B + 128 \end{aligned} \tag{4.7}$$

et inversement nous avons :

$$\begin{aligned} B &= 1.164(Y - 16) + 2.018(Cb - 128) \\ V &= 1.164(Y - 16) - 0.813(Cr - 128) - 0.391(Cb - 128) \\ R &= 1.164(Y - 16) + 1.596(Cr - 128) \end{aligned} \tag{4.8}$$

Les traitements de la compression fractale sont appliqués sur chacune des composantes RVB ou YCrCb de l'image. Nous avons testé l'algorithme de compression fractale sur trois images couleurs de taille 128×128: image Lena, image Pepper et image baboon. Les images originales ainsi que les images reconstruites en utilisant les deux modèles du codage (RVB et YCrCb) sont présentées dans la figure 4.13. Les blocs destination utilisés sont de taille 4×4 pixels. Les résultats numériques de compression sont donnés dans le tableau 4.4.

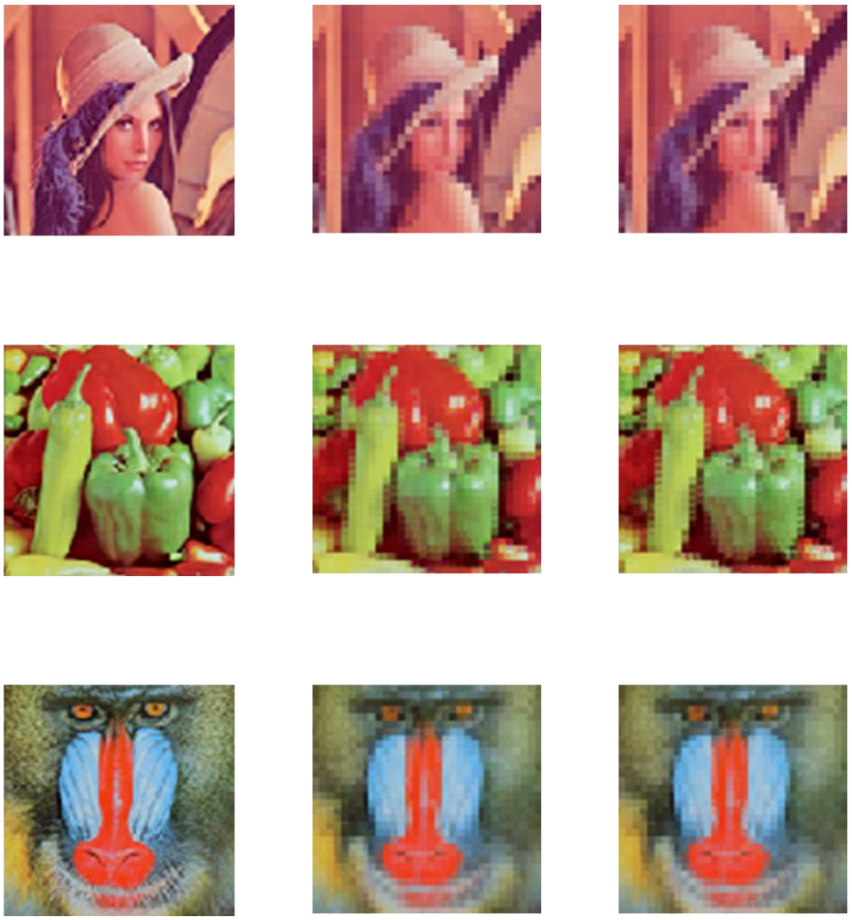


Figure 4.13 : Images couleurs originales, images reconstruites dans l'espace RVB, images reconstruites dans l'espace YCrCb, avec taille des blocs destination = 4×4.

Tableau 4.4 : Résultats de la compression fractale des images couleurs (Taille de bloc destination = 4×4)

Images	***Codage RVB***			***Codage YCrCb***		
	PSNR (dB)	**R_c**	**CT (s)**	**PSNR (dB)**	**R_c**	**CT(s)**
Lena	22.77	6.65	144	22.73	7.63	146.13
Pepper	21.50	7.73	140.79	21.73	8.71	141.96
Baboon	22.57	7.59	144.18	22.64	8.18	148.96

Il est évident que la qualité de reconstruction dépend fortement du contenu des blocs source, nous remarquons des erreurs de codage sont situées essentiellement aux contours des images. On remarque aussi que les meilleurs rapports de compression sont obtenus avec l'espace couleur YCrCb, ce qui justifie son pouvoir décorrélatif.

4.3 Evaluation de la compression hybride fractale-ondelette :

4.3.1 Résultats de l'algorithme de compression hybride fractale-ondelette :

Nous montrons dans cette section l'avantage d'introduire les ondelettes dans le schéma de compression fractale en termes de qualité d'image et du temps de compression. Pour expérimenter les performances du codage hybride fractale-ondelettes, nous avons réalisé une série de tests.

Dans un premier temps, nous avons implanté l'algorithme de compression hybride fractale-ondelette (WFC) que nous l'avons comparé avec l'algorithme standard de compression fractale (FIC). Pour cela, nous avons réalisé un codage fractale après transformations en ondelettes discrètes. La qualité de l'image compressée dépend essentiellement du choix de l'ondelette. La base bi-orthogonale de Daubechies CDF9/7 donne généralement les meilleurs résultats.

Le déroulement de l'algorithme est le suivant : L'image originale est décomposée en 5 niveaux notés n (n = 1,…, 5). La recherche des similarités au sein des sous bandes de détails (de même direction) se fait entre blocs source et destination de même taille 8×8, 4×4, 2×2 et 1×1. L'appariement est réalisé entre les blocs source des sous bandes de niveaux 2, 3, 4 et 5 et les blocs destination des niveaux 1, 2, 3 et 4. Le facteur d'échelle est ainsi calculé pour chaque appariement dans les sous bandes horizontal, vertical et diagonal. Notons que les isométries ne sont pas considérées dans cette implémentation.

Les algorithmes de compression et décompression fractale-ondelette sont résumés ci-après :

Algorithme 4.5 : Codage hybride fractale-ondelette

Début

Lire l'image à traiter

Transformée en ondelettes de l'image originale en N niveau de décomposition

Pour *chacune des sous bandes de détails (HL, LH, HH) du* $n^{ème}$ *niveau de décomposition*

Créer une partition des blocs source

Pour *chacune des sous bandes de détails (HL, LH, HH) du niveau (n-1)*

Créer une partition des blocs destination

Fin Pour

Fin Pour

Pour *chaque bloc destination de chaque sous bande (niveau n-1)*

Pour *chaque bloc source de la sous bande de même direction (niveau n)*

Trouver la transformation fractale appropriée

Fin pour

Sauvegarder les paramètres de la transformation fractale

Fin pour

Fin

Algorithme 4.6 : Décodage hybride fractale-ondelette

Début

Prendre une image initiale quelconque

Calculer la transformation en ondelettes discrète (N niveau)

Itérer

Pour *chaque bloc destination de chaque sous bande de détails HL LH et HH*

Lire le numéro du bloc source à manipuler

Appliquer la transformation fractale à ce bloc

Insérer le résultat dans l'image

Fin pour

Prendre l'image résultante comme étant image initiale;

Fin Itérer

Calculer les transformées inverses (IDWT)

Fin

Pour les tests nous avons utilisé une série d'images standards de 8 bpp contenant 512×512 pixels au format BMP (Lena, couple, Pepper, baboon, man, boats, Barbara, lake et goldhill) La figure 4.14 présente les images originales de test, les images reconstruites par l'algorithme de compression fractale (FIC) en utilisant des blocs destination de taille 4×4 et les images décompressées par l'algorithme hybride fractale-ondelette. Les résultats de comparaison en termes de rapport de compression R_c, rapport signal sur bruit (PSNR) et de temps compression en secondes sont fournis au tableau 4.5. Le graphe de la figure 4.15 donne l'évolution de temps de compression pour les deux algorithmes.

Figure 4.14 : Images originales de test, Image reconstruites par l'algorithme de compression fractale et Images reconstruites par l'algorithme par l'algorithme hybride.

Tableau 4.5 : Résultats numériques de la compression fractale-ondelette.

Méthodes / Images	FIC			WFC		
	PSNR (dB)	Rc	CT (s)	PSNR (dB)	Rc	CT (s)
Lena	29.16	6.66	12512	32.34	10.08	57.84
Couple	24.72	9.24	12636	27.10	10.26	59.82
Pepper	26.62	10.04	15526	30.23	10.12	58.79
Baboon	23.44	10.54	12349	25.19	10.68	58.83
Man	26.62	10.30	12316	27.95	10.34	58.63
Boat	24.60	6.83	12706	26.99	10.13	58.68
Barbara	23.06	9.05	12658	24.45	10.04	58.73
Lake	24.66	10.64	9104	26.69	10.92	59.04
Goldhill	26.60	8.15	12734	28.87	10.34	58.76
MOYENNE	25.5	9.35	12500	27.75	10.32	58.79

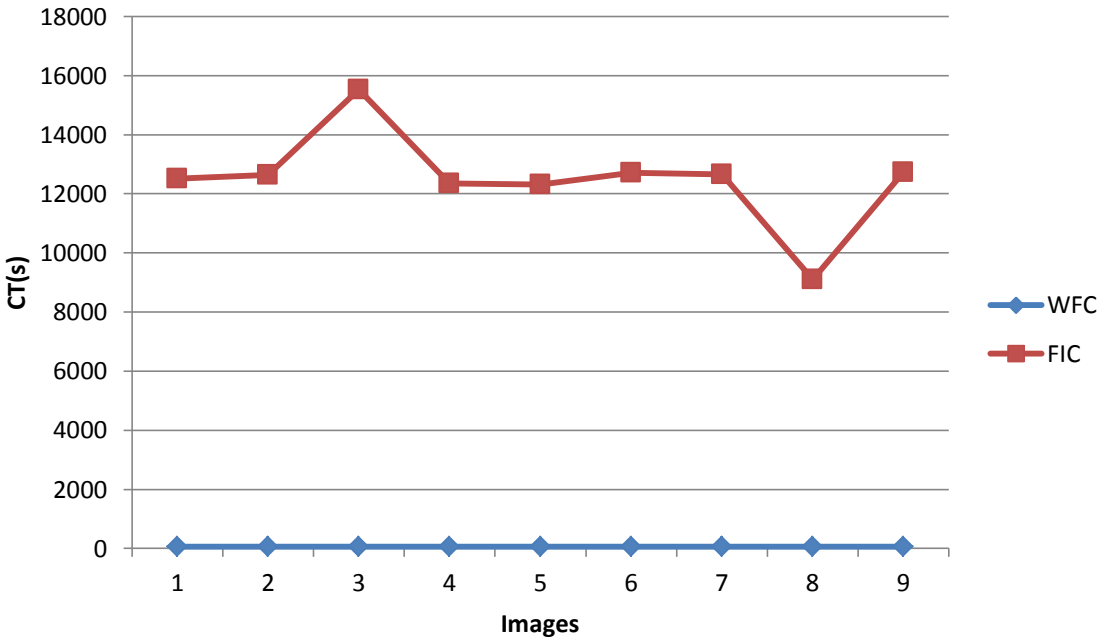


Figure 4.15 : Les temps de compression pour l'ensemble des images de test.

En examinant les résultats de la compression, on remarque que l'algorithme de compression fractale-ondelette (WFC) fournit de meilleures valeurs du PSNR, un gain moyen constaté est de l'ordre de 2.25 dB pour un rapport de compression moyen de 10.32. Nous pouvons constater que le codeur hybride fractale-ondelette (WFC) est meilleur du point de vue rapport de compression/qualité d'image dans la plus part des cas par rapport à l'algorithme de compression fractale standard (FIC).

De plus, la complexité de l'algorithme hybride fractale-ondelette est réduite vu que l'opération du codage fractale va manipuler des images de petites dimensions dans le domaine des ondelettes. Nous avons réalisé 5 niveaux de décomposition, donc nous avons manipulé des images de différentes résolution, 256×256, 128×128, 64×64, 32×32 et 16×16. L'appariement est fait entre les blocs destination de taille 8×8, 4×4, 2×2 et 1×1 des sous bandes des niveaux 1, 2, 3 et 4 et les blocs source de même taille issu de la décomposition à la résolution 2, 3, 4 et 5. En appliquant la formule (4.6), le nombre d'opérations effectuées pour chaque paire de sous bande adjacente de même direction est égale à 256 opérations par pixel. Le nombre d'opérations totale sera donc égale à 3*4*256 ie. 3072 opérations par pixel. La compression fractale sur les images de test (512×512 pixels) avec des blocs destination de taille 4×4, nécessite un nombre d'opérations de 65536 par pixel. Par conséquent, le temps de compression hybride fractale-ondelette est considérablement réduit par rapport à la compression fractale.

4.3.2 Influence du niveau de décomposition :

Nous allons maintenant nous intéressé à l'influence du niveau de décomposition (N) sur la compression hybride fractale-ondelette. Nous appliquons l'algorithme hybride (WFC) sur trois images standards de taille 512×512 (Lena, couple, Pepper) avec différents niveaux de décomposition. Les graphes 4.16 et 4.17 montrent l'effet du niveau de décomposition sur les valeurs du PSNR et les temps de compression (CT) pour un taux de compression CR d'environ 91%.

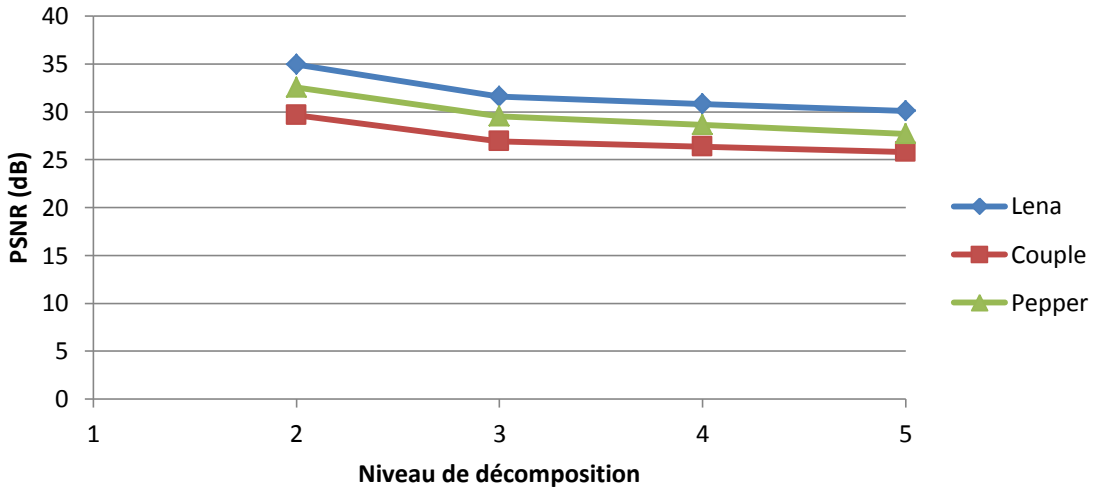


Figure 4.16 : Les PSNRs pour différents niveaux de décomposition.

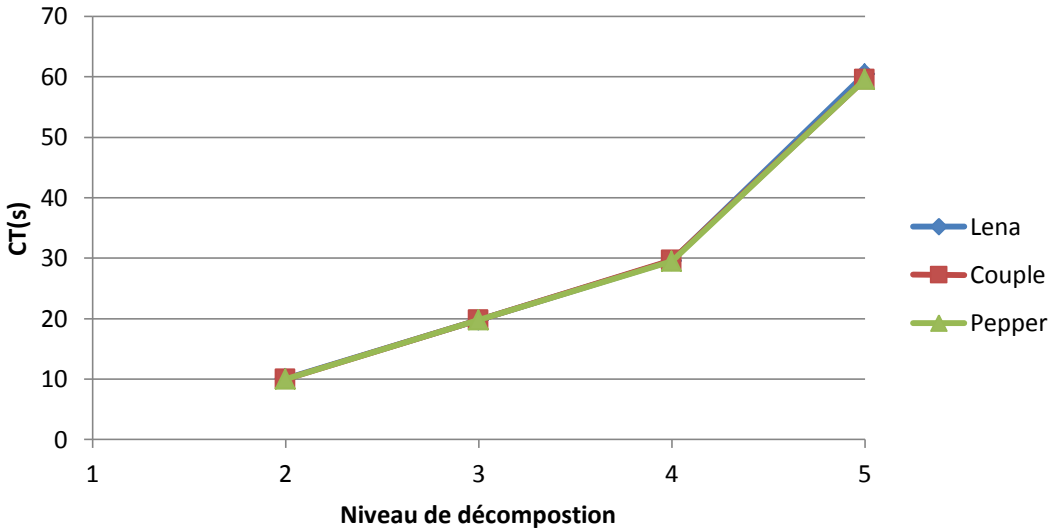


Figure 4.17 : Temps de compression pour différents niveaux de décomposition.

Nous observons que les meilleures valeurs du PSNRs sont obtenues lorsque le niveau de décomposition est petit. Ceci s'explique par le fait que quand le niveau de décomposition augmente plus amples détails sont ajoutés, mais plus les erreurs de reconstruction augmente. Nous remarquons aussi que la complexité de calcul augmente avec le niveau de décomposition car le nombre d'opérations se double à chaque niveau supplémentaire.

4.3.3 Etude comparative:

Pour réaliser la comparaison, nous avons compressé l'image de Lena par le codeur hybride fractale-ondelette (WFC) en utilisant l'ondelette CDF 9/7 avec deux niveaux de décomposition, puis nous avons comparé les résultats avec la norme JPEG et l'algorithme SPIHT, pour des débits binaires allant de 0.52 bpp à 1.36 bpp. Le tableau 4.6 fournit les valeurs du PSNR pour les trois algorithmes de compression et la figure 4.18 donne les courbes débit/distorsion.

Tableau 4.6 : Les résultats de comparaison pour l'image de Lena.

Bpp	**PSNR (dB)**		
	WFC	SPIHT	JPEG
0,52	**37,39**	**36,73**	**33,51**
0,56	**37,41**	**37,06**	**33,76**
0,59	**37,52**	**37,29**	**33,97**
0,71	**38,24**	**38,11**	**34,79**
1,36	**41,23**	**41,48**	**37,77**

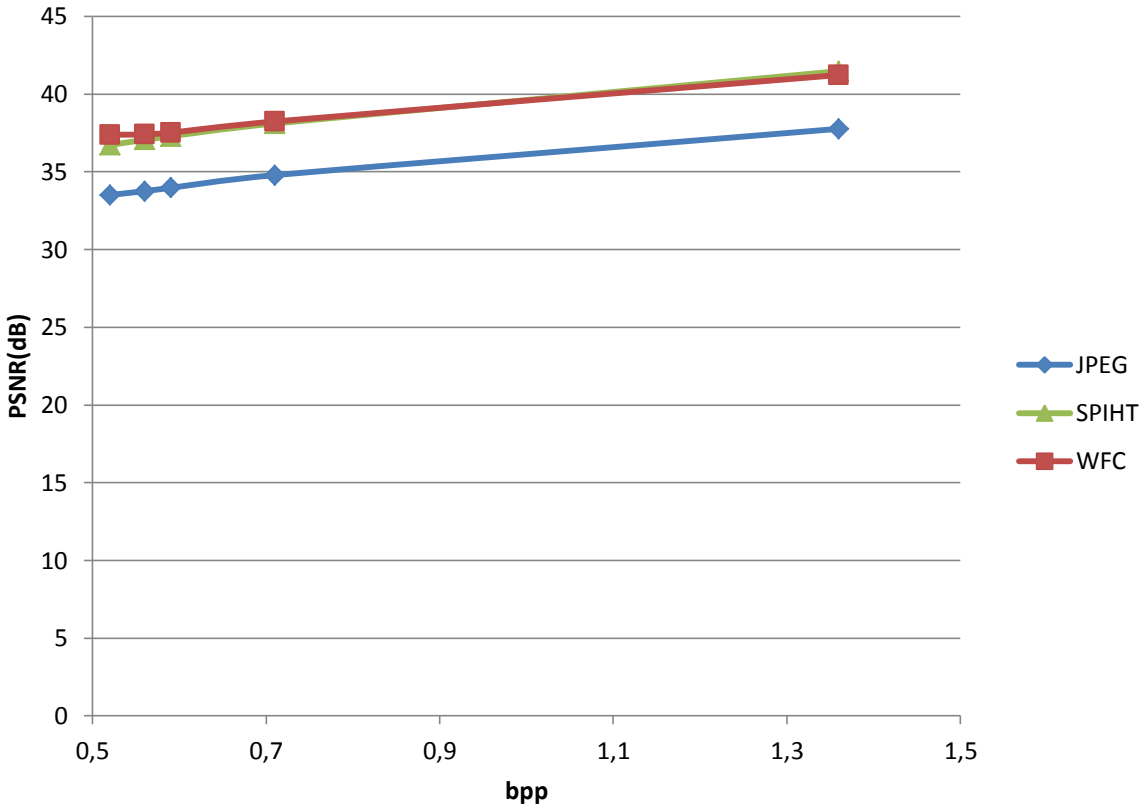


Figure 4.18 : Comparaison entre l'algorithme de compression fractale-ondelette, SPIHT et JPEG sur l'image de Lena.

Nous pouvons constater que l'algorithme hybride fractale-ondelette garantie un gain de 3.59 dB par rapport à la norme JPEG et offre des valeurs comparables avec celles obtenues par l'algorithme SPIHT.

4.4 Evaluation du codage hybride fractale-ondelette sans perte :

Le schéma proposé (WFC_LS) qui combine le codeur hybride fractal-ondelette (WFC) avec le codage de Huffman est testé sur les images précédentes et est comparé à l'algorithme de Huffman. L'organigramme de la méthode proposée (WFC_LS) est représenté sur la figure 4.19. Les résultats en termes de débits en bpp et de temps d'exécution en secondes sont fournis au tableau 4.7 avec un ensemble d'images testé (Lena, couple, Pepper, baboon, man, boats, Barbara, lake et goldhill).

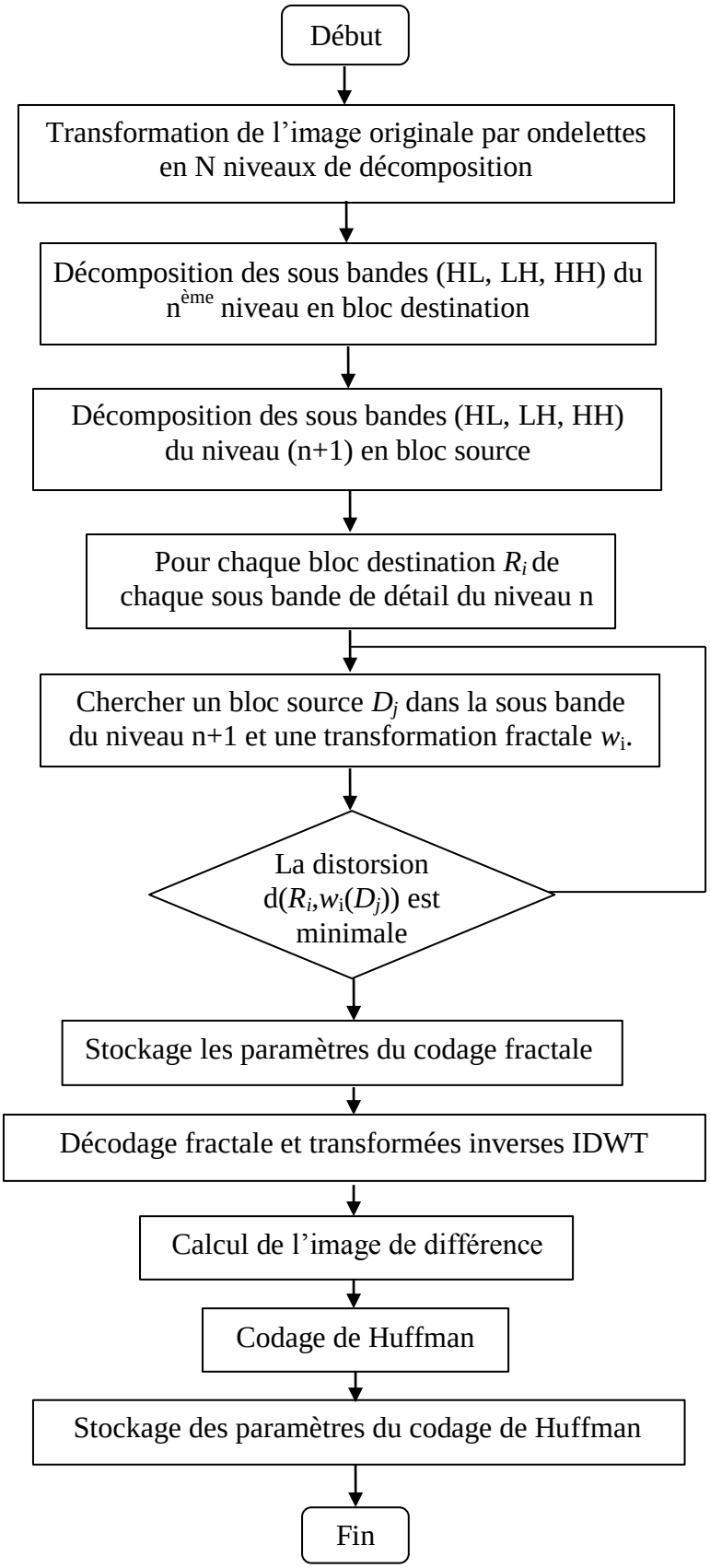


Figure 4.19 : Organigramme du codeur hybride fractale-ondelette sans perte

Tableau 4.7 : Résultats de la compression sans perte.

Méthodes / Images	WFC_LS (Proposé)		Huffman	
	bpp	**CT (s)**	**bpp**	**CT (s)**
Lena	5.79	77	7.14	17.04
Couple	6.88	76.21	7.12	25.11
Pepper	6.34	75.91	7.61	23.51
Baboon	7.44	74.75	7.31	25.98
Man	6.72	75.83	7.27	23.66
Boats	6.83	90.77	7.27	26.59
Barbara	7.27	77.15	7.69	24.82
Lake	7.00	74.43	7.51	25.13
Goldhill	6.55	77.64	7.54	23.56
Moyenne	**6,75**	**77,74**	**7,38**	**25,37**

Nous pouvons constater que le schéma proposé (WFC_LS) fournit des débits sensiblement inférieurs par rapport à l'algorithme de Huffman Notre codeur donne en moyenne 6.75 bpp i.e. 0.62 bpp moins que le codeur de Huffman qui fournit seulement 7.38 bpp. Même si ce gain paraît faible, il reste acceptable et critique pour une méthode de compression sans perte.

Dans le cas de l'image Baboon, on obtient seulement 7.44 bpp avec notre schéma de compression. L'image Lena, nous permet d'atteindre le meilleur résultat de compression: 5.79 bpp. Nous pouvons constater que la performance de la compression dépend du contenu de l'image à coder. L'autre observation que nous faisons est que notre schéma est moins rapide. Ce constat s'explique par la complexité que représentent les trois méthodes combinées.

4.5 Application aux images radiographiques des défauts de soudure :

4.5.1. La radiographie industrielle :

Le contrôle et la caractérisation des matériaux et des composants manufacturés sont des tâches essentielles dans de nombreux domaines industriels pour connaître la qualité des produits en cours de fabrication. Le contrôle non destructif est l'ensemble de méthodes qui permettent de caractériser l'état d'intégrité de structure sans les dégrader, permettant ainsi leur utilisation ultérieur. Les défauts de soudure sont en général inhérents aux procédés de soudage. Ils dépendent de plusieurs facteurs : mauvaise exécution, qualité du métal de base et/ ou du produit d'apport, propreté des régions à souder et du procédé lui-même.

Dans l'industrie, la radiographie qui est l'une des méthodes de contrôle non destructif, présente l'avantage de fournir des images directement exploitables. Elle sert à détecter les criques internes, les vides, les porosités et les inclusions dans les matériaux [149]. La détection des défauts est parfois très difficile à cause de la mauvaise qualité des films, le mauvais contraste, le bruit et les faibles dimensions des défauts. En plus, l'interprétation humaine de la qualité de la soudure est très subjective et dépend de l'aptitude et l'expérience de l'interpréteur. La nécessité d'aider l'interpréteur dans les cas de diagnostic délicat a conduit les spécialistes à utiliser les techniques de traitement d'images. Le traitement numérique des images recouvre l'ensemble des procédés d'extraction d'information qualitative des images numériques, allant du prétraitement jusqu'à l'extraction de la zone de l'objet à identifier [150][151][152].

4.5.2 Compression avec pertes des images radiographiques des défauts de soudure :

L'utilisation des images radiographique sous leurs formes numériques ne serait possible sans la compression préalable de celles-ci [153]. Dans ce contexte nous avons met en application les algorithmes de compression fractale (FIC) et hybride (WFC) par une série d'images radiographiques de taille 128×512 pixels au format BMP. Elles représentent plusieurs défauts de soudures : Manque de pénétration, Porosité, Manque de fusion, Fissure, Caniveau externe et inclusion de tungstène (Figure 4.20).

On donne ici une brève description et les causes des défauts de soudage présentés sur les images de test [154] :

- Manque de pénétration : Remplissage incomplet du fond de la rainure de soudage avec le métal de la soudure, dû à l'utilisation d'électrodes de diamètres trop grand, à l'emploi d'un courant d'intensité trop faible ou d'une mauvaise préparation des chanfreins.
- Porosité : cavité gazeuse emprisonnée durant le processus de solidification, due essentiellement à la présence de l'humidité dans les pièces et les électrodes.
- Manque de fusion : manque de liaison dans le cordon de soudure ou bien entre celui-ci et le métal de base dû à une intensité du courant de soudage trop faible, une avance

trop rapide ou des cordons trop bombés formant entre eux un angle aigu dans le cas de soudage multipasse.

- Fissure : rupture dans le métal qui résulte de la présence des contraintes localisées qui dépassent la résistance limitée du matériau.
- Caniveau : c'est un manque de métal en forme de sillon en bordure du cordon. Ils sont dus à un courant de soudage trop intense et à une technique opératoire défectueuse.
- Inclusion : corps solide étrange emprisonné dans la masse du métal fondu dû au nettoyage insuffisant des cordons.

La figure 4.21 présente les images reconstruites par la compression fractale et le codage hybride fractal-ondelette [155][156]. Une comparaison des performances est présentée dans le tableau 4.8.

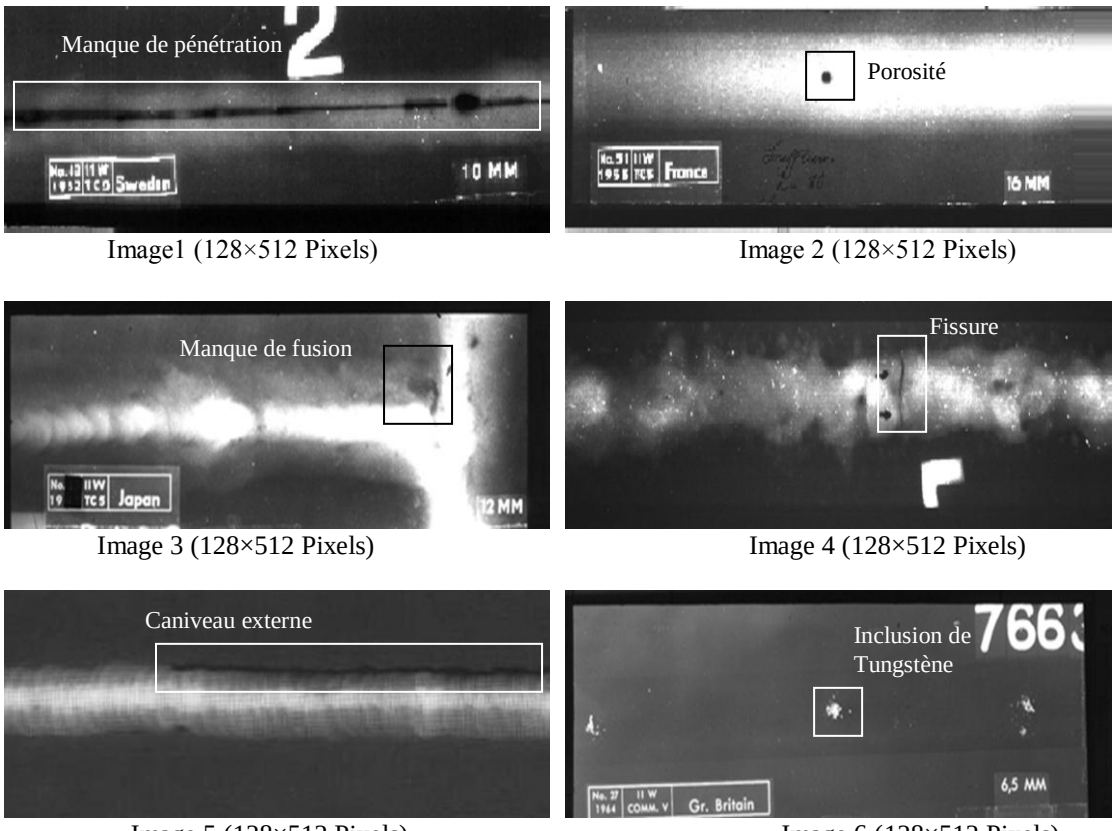
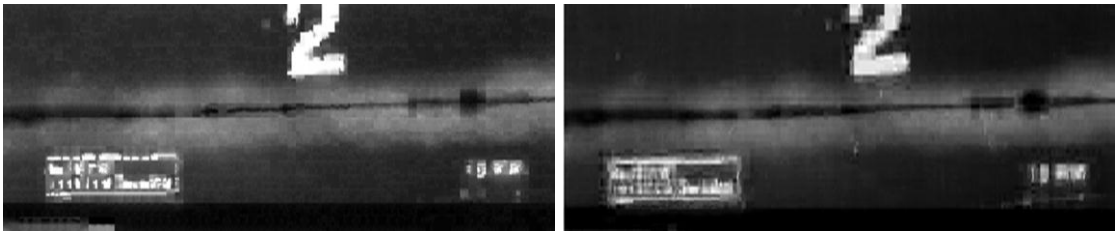


Image1 (128×512 Pixels) Image 2 (128×512 Pixels)

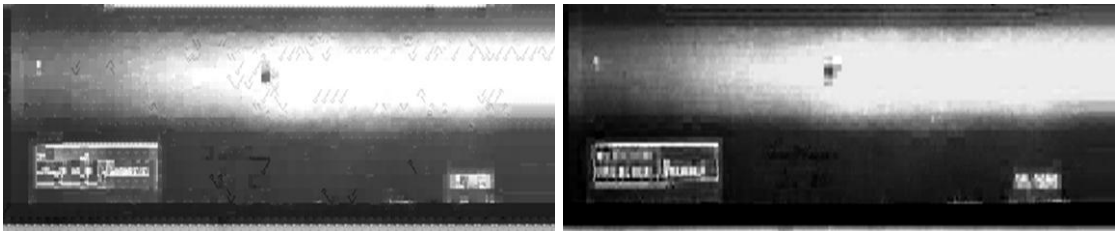
Image 3 (128×512 Pixels) Image 4 (128×512 Pixels)

Image 5 (128×512 Pixels) Image 6 (128×512 Pixels)

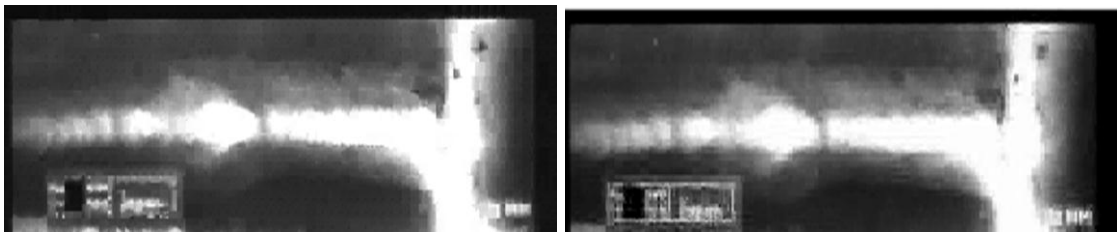
Figure 4.20 : Images radiographiques de test.



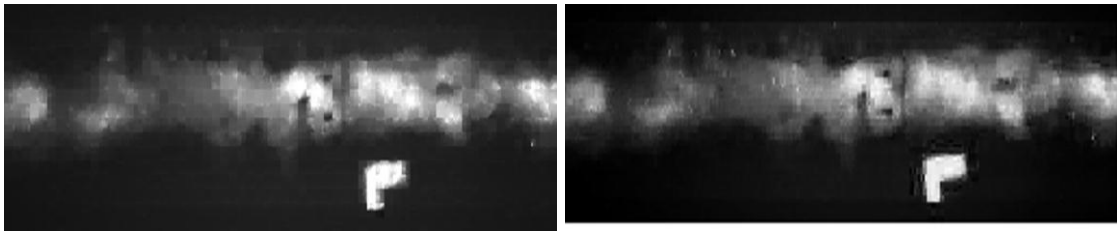
PSNR = 21.55 dB PSNR = 24.29 dB



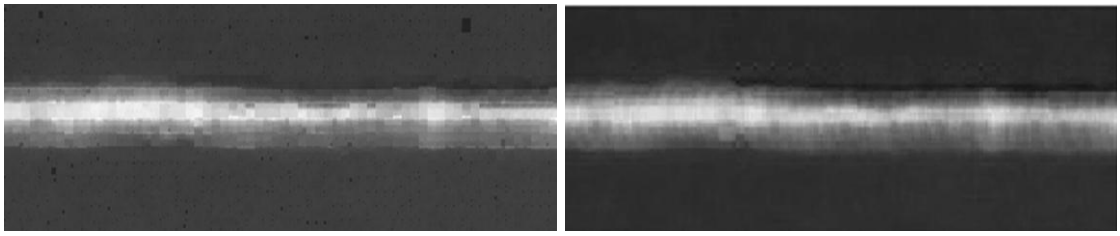
PSNR = 19.51 dB PSNR = 22.95 dB



PSNR = 23.06 dB PSNR= 26.80 dB



PSNR = 22.74 dB PSNR = 28.83 dB



PSNR = 28.49 dB PSNR = 35.83 dB

PSNR = 20.14 dB PSNR = 26.71dB

Figure 4.21 : Résultats du codage à 1.12 bpp, gauche par l'algorithme de compression fractale et droite par le codage hybride fractale-ondelette.

Tableau 4.8 : Comparaison de performances pour les images radiographiques de test.

bpp	PSNR (dB)											
	Image 1		Image 2		Image 3		Image 4		Image 5		Image 6	
	FIC	WFC	FIC	WFC	FIC	WFC	FIC	WFC	FIC	WFC	FIC	WFC
0.96	20.13	21.70	9.45	17.35	19.61	19.88	21.30	21.45	27.89	34.19	19.52	21.70
1.12	21.55	24.29	19.51	22.95	23.06	26.80	22.74	28.83	28.49	35.82	20.14	26.71

En examinant les résultats de compression on constate que la qualité des images reconstruites par le codage hybride est meilleure, les rapports signal sur bruit le prouvent. Malgré l'existence de quelques régions bruitées, on peut identifier les défauts de soudure sur les images décompressées. Tandis que la détection des défauts sur les images fournies par la compression fractale est difficile.

4.5.3. Segmentation des images radiographiques des défauts de soudure :

Nous abordons principalement le problème de la segmentation non supervisée des images radiographiques. La segmentation constitue un des problèmes les plus importants en traitement d'images, car le résultat obtenu à l'issue de cette étape conditionne fortement la qualité finale de l'interprétation. L'objectif de la segmentation est de décrire l'importante quantité d'informations contenues dans l'image en recherchant des indices visuels pertinents et discriminants permettant de la représenter sous forme plus condensée et facilement exploitable [151].

Les méthodes de segmentation par classification de pixels affectent chaque pixel à une classe en fonction d'un ou plusieurs attributs de ce pixel. La classification est dite supervisée lorsque des informations à priori sont utilisées pour la construction de classe sous la forme d'ensemble d'apprentissage. Dans le cas où aucune connaissance à priori n'est disponible, on parle de classification non supervisée.

Nous proposons une segmentation des images radiographiques par l'algorithme de classification Expectation Maximisation (EM) [157]. Notre objectif principal est d'appliquer l'algorithme EM dans la segmentation des images des films radiographiques, dans un souci d'extraire le défaut de soudure du joint soudé d'une part, et ce dernier du métal de base d'autre part. On considère la distribution de la luminance de chaque pixel comme une variable aléatoire approximée par un modèle de mélanges gaussiens. Les paramètres de ce modèle sont estimés par l'algorithme EM [151].

Pour vérifier si la compression avec pertes altère les résultats de la segmentation nous avons compressé un ensemble de trois images radiographiques de taille 128×512 par l'algorithme de compression hybride fractale-ondelette (WFC). La figure 4.22 présente les images originales et les images reconstruites par l'algorithme hybride. Par la suite nous avons

segmentées les images compressées par l'algorithme EM et nous les avons comparées aux images originales. Les résultats de la segmentation sont donnés dans la figure 4.23.

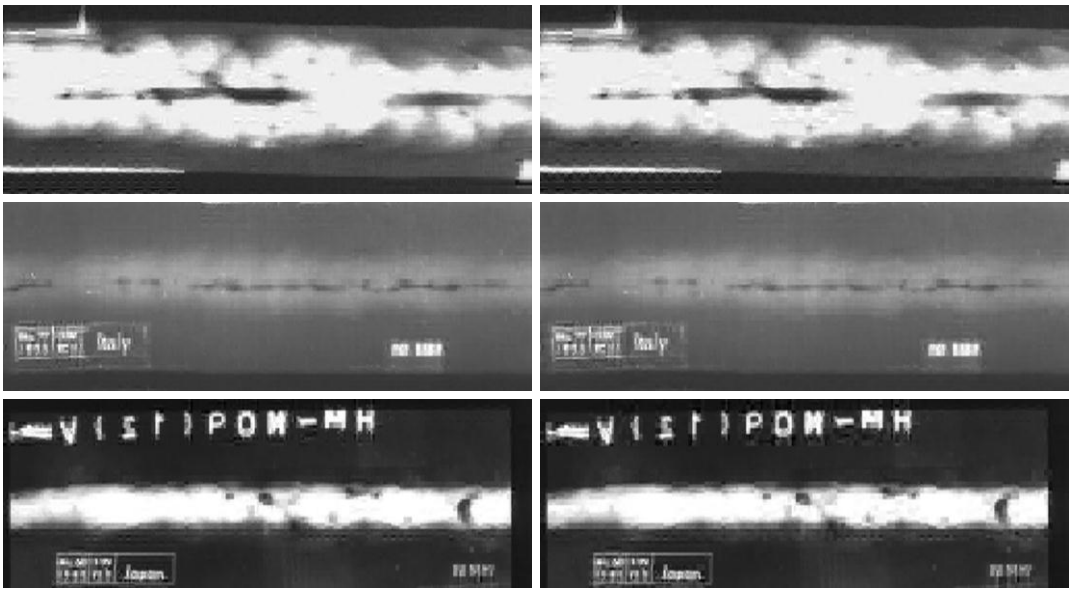


Figure 4.22 : Images radiographiques de test et images compressées par l'algorithme hybride

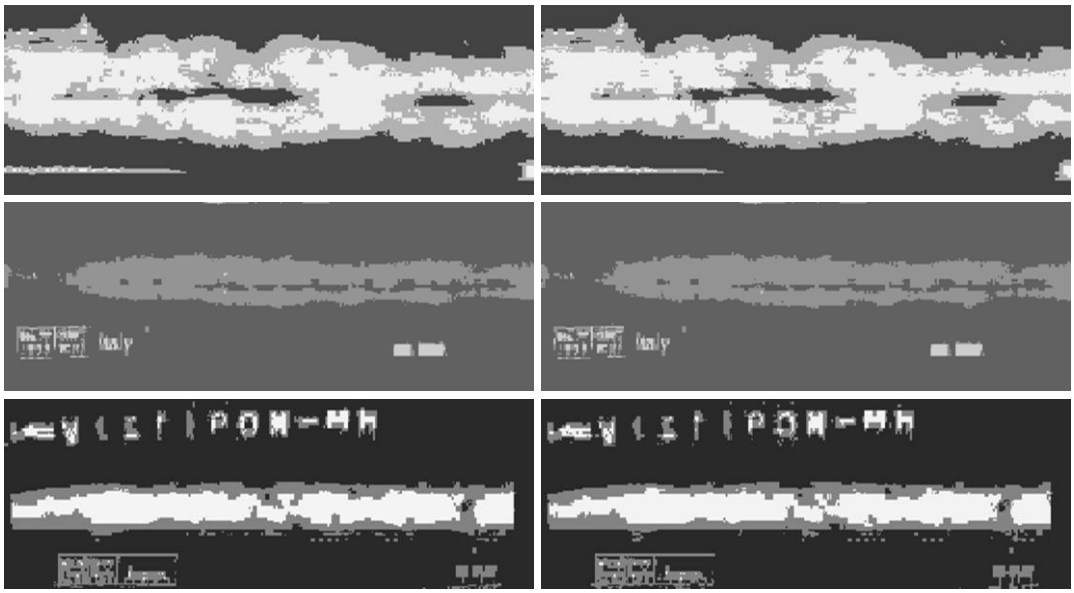


Figure 4.23 : résultats de la segmentation par l'algorithme EM des images originales et compressées.

En examinant les résultats de la segmentation obtenus, on remarque que les images segmentées après compression sont similaires aux images originales segmentées, ce qui montre que la méthode de compression hybride fractale-ondelette fournit des images de bonne qualité et n'affecte pas le résultat de l'interprétation. Pour l'ensemble des images, les défauts, le joint soudé et le métal de base sont misent en évidence. Cela montre que la segmentation EM est une méthode robuste et capable de fournir des performances satisfaisantes.

4.5.4. Compression sans perte des images radiographiques des défauts de soudure :

Dans le contrôle non destructif, on favorise en général la qualité d'image par rapport au taux de compression. Pour cette raison nous avons testé notre schéma sans perte sur un ensemble contenant 25 images radiographiques des défauts de soudure. Les résultats obtenus ont été comparés à l'algorithme de Huffman. Le graphe de la figure 4.24 montre les valeurs des débits pour les 25 images de test.

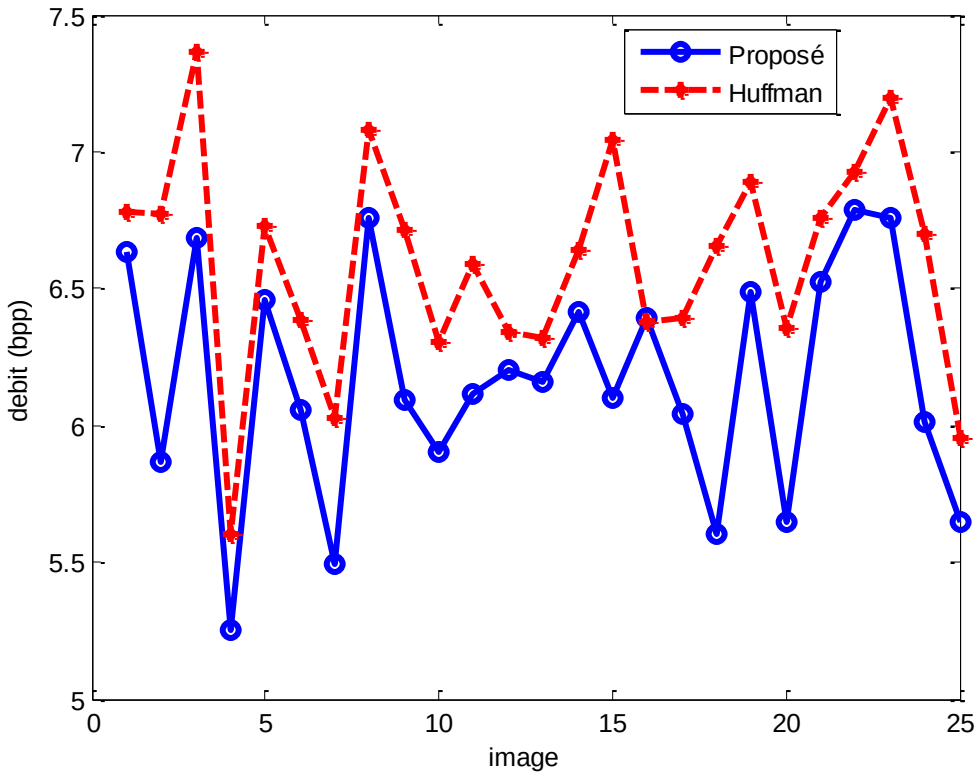


Figure 4.24 : Résultats de la compression sans perte pour les images radiographiques

Nous constatons que le schéma de compression proposé génère des fichiers de taille plus petite que celle fournie par l'algorithme de Huffman. Notre schéma donne en moyenne 6.16 bpp cependant avec le codage de Huffman on obtient 6.59 bpp i.e. un gain de 0.43 bpp.

4.6 Conclusion :

Dans ce chapitre nous avons montré l'importance du choix d'un modèle de partitionnement dans le schéma de compression fractale. La partition carrée reste un bon choix pour cette méthode. La fixation de la taille des blocs se fait conformément à nos besoins et suivant la qualité de l'image reconstruite que nous voulons obtenir.

Nous avons également évalué le codeur hybride fractal-ondelette. Le schéma hybride développé a été comparé à l'algorithme standard de compression fractale. Les images décompressées par l'algorithme hybride sont de meilleures qualités visuelles que celles obtenues avec la compression fractale. En plus, l'utilisation des ondelettes permet d'accélérer la phase du codage fractale.

Le schéma de compression hybride fractale-ondelette sans perte, que nous avons proposé, est plus performant que l'algorithme de Huffman. Le codeur hybride fractale-ondelette sans perte s'adapte bien aux images radiographiques des défauts de soudures qui nécessitent une qualité parfaite de l'image reconstruite.

Conclusion générale:

Dans le cadre de cette thèse nous avons présenté diverses méthodes de compression des images couleurs ou en niveaux de gris basées sur les fractales. Dans un premier temps nous avons montré l'intérêt d'utiliser un modèle de partitionnement géométrique dans le schéma de compression fractale. On a constaté qu'avec le partitionnement carré on obtient de bons résultats en choisissant la taille des blocs. Dans le cas où les détails de l'image ne sont pas importants, on peut privilégier le taux de compression et négliger un peu la qualité de l'image décodée ou reconstruite en choisissant une taille large des blocs destination. D'autre part, si les détails de l'image sont importants il faut donc privilégier la qualité de l'image décodée et négliger le taux de compression en utilisant des petites tailles.

Le codage fractale nécessite, pour chaque bloc de l'image, des comparaisons de manière systématique et séquentielle, afin de trouver le bloc le plus similaire pour une distorsion donnée. Le temps de codage obtenu est donc prohibitif, pour réduire le temps de cette recherche et accélérer la phase du codage, nous avons présenté deux approches, une est basée sur la recherche au voisinage et l'autre consiste à définir un seuil relatif à la mesure de distorsion. L'hybridation des fractales et la théorie des ondelettes permet aussi d'accélérer la vitesse d'exécution.

A travers le Protocol expérimental que nous avons suivi, nous avons montré que le codeur hybride fractale-ondelette est meilleur du point de vue débit/distortion que l'algorithme de compression fractale standard et la norme de compression JPEG et est comparable avec l'algorithme SPIHT. Par la suite nous avons proposé un nouveau schéma de compression sans perte, en faisant associer le codage fractale, la transformée en ondelette avec une autre méthode de compression sans perte. Ce schéma tire profit des avantages de chaque méthode employée : le taux de compression issu de la compression fractale, la qualité du décodage donnée par l'algorithme de Huffman et la rapidité des ondelettes.

L'application des méthodes développées sur des images standards et des images radiographiques des défauts de soudure, nous a permis de constater que les performances de la compression par fractales et ondelettes dépendent de plusieurs paramètres (taille des blocs, niveau de décomposition, choix de l'ondelette) qui peuvent être ajustés pour produire des résultats différents du point de vue taux de compression et qualité visuelle de l'image décodée. Le schéma du codage que nous avons proposé permet de restituer les images originales sans aucune distorsion et dépasse le codeur de Huffman.

Les perspectives et les directions de recherches à suivre pour améliorer nos résultats portent sur l'utilisation d'autres modèles de partitionnement géométrique dans le schéma hybride. L'association des transformations dérivées des ondelettes telles que les curvelettes et les contourlettes avec le codage par fractales. L'incorporation d'autres méthodes de compression sans perte telles que LZW et RLE, dans le schéma proposé. L'extension des méthodes de codage présentées dans cette thèse au codage des séquences d'images.

Annexe 1: Génération des fractales IFS

Les IFS permettent la génération d'images totalement auto-similaires, en utilisant un nombre restreint de transformations. Des méthodes simples pour générer des fractales définies par des codes de système de fonctions itérées (IFS) ont été développées par différents chercheurs [60]. Nous allons présenter deux approches algorithmiques de calcul de fractales.

A1.1 Algorithme déterministe :

Cet algorithme applique les itérations sur des ensembles de points plutôt que sur des points singuliers. Il commence avec un ensemble de points quelconque et applique chacune des transformations w_i ; i = 1, 2, …….N à cet ensemble. Tous les nouveaux points générés deviennent les points de départ pour la prochaine itération.

Le pseudo-code de l'algorithme est donné par :

Entrée : w_i *;* i *= 1, 2, …….N transformation contractantes, K entier,* X_0 *un ensemble de points initial*

Sortie : Attracteur de l'IFS

Pour k de 0 jusque K

Pour i de 1 jusque N

Calculer $w_i(X_k)$

Calculer $X_{k+1} = \bigcup_{i=1}^{N} w_i(X_k)$

Fin de i

Fin de k

Exemple 1: Triangle de Sierpinsky

Considérons les trois transformations suivantes :

$$w_1\begin{pmatrix}x\\y\end{pmatrix}=\begin{bmatrix}0.5 & 0\\0 & 0.5\end{bmatrix}\begin{pmatrix}x\\y\end{pmatrix},\quad w_2\begin{pmatrix}x\\y\end{pmatrix}=\begin{bmatrix}0.5 & 0\\0 & 0.5\end{bmatrix}\begin{pmatrix}x\\y\end{pmatrix}+\begin{pmatrix}0\\0.5\end{pmatrix},\quad w_3\begin{pmatrix}x\\y\end{pmatrix}=\begin{bmatrix}0.5 & 0\\0 & 0.5\end{bmatrix}\begin{pmatrix}x\\y\end{pmatrix}+\begin{pmatrix}0.5\\0.5\end{pmatrix}$$

Chacune de ces trois transformations est une contraction pure par un facteur de contraction 0.5 suivie d'une translation. L'attracteur de l'IFS est montré sur la figure A1.1, il est appelé triangle de Sierpinski. Le caractère fractal de cet ensemble est clairement vu, puisqu'une petite partie ressemble au tout.

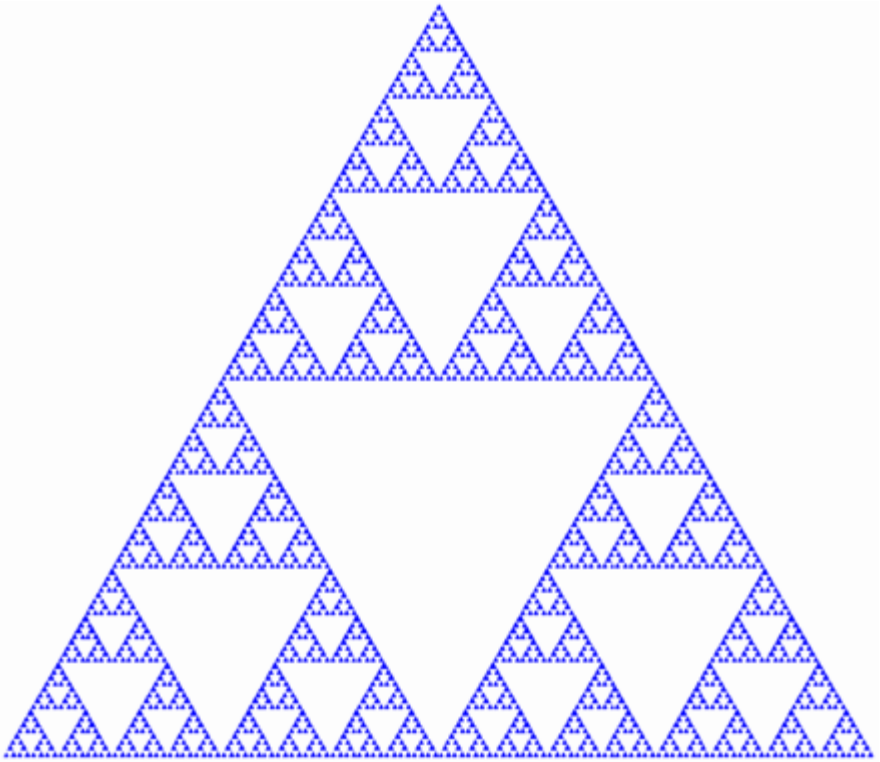


Figure A1.1. Triangle de Sierpinski [60]

A1.2 Algorithme probabiliste :

Cet algorithme offre une méthode simple pour construire une approximation grossière de l'attracteur. Associons à L'IFS définit par les transformations w_i ; i = 1, 2, …….N, un jeu de probabilités $P_i > 0$ i = 1, 2, ……..N. :

$$P_i = \frac{|\det w_i|}{\sum_{i=1}^{N} |\det w_i|}$$

On choisit arbitrairement un point initial x_0 ensuite on calcule les points x_i successifs, en appliquant les fonctions choisies aléatoirement à partir de l'IFS.

L'algorithme peut être décrit par le pseudo-code :

Entrée : $w_1, w_2, \ldots w_N$ transformation contractantes, K entier, x_0 un point initial

Sortie : Attracteur de l'IFS

Calculer $P_1, P_2, \ldots P_N$

Pour k de 0 jusque K

Choisir i entre 1 et N avec probabilité P_i

Calculer $x_{i+1} = w_k(x_i)$

Si k > min – itérations alors tracer x_i

Fin

Exemple 2 : Fougère de Barnsley

Un exemple classique de la théorie des IFS concerne la feuille de Fougère représentée sur la figure A1.2 et calculée à l'aide de l'algorithme probabiliste. La feuille de fougère est l'attracteur de L'IFS défini par les quatre transformations suivantes :

$$w_1\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0.16 \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$w_2\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 0.85 & 0.04 \\ -0.04 & 0.85 \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 0 \\ 1.6 \end{bmatrix}$$

$$w_3\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 0.2 & -0.26 \\ 0.23 & 0.23 \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 0 \\ 1.6 \end{bmatrix}$$

$$w_4\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} -0.15 & 0.28 \\ 0.26 & 0.4 \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

et son jeu de probabilité associé est [0.01, 0.85, 0.07, 0.07]

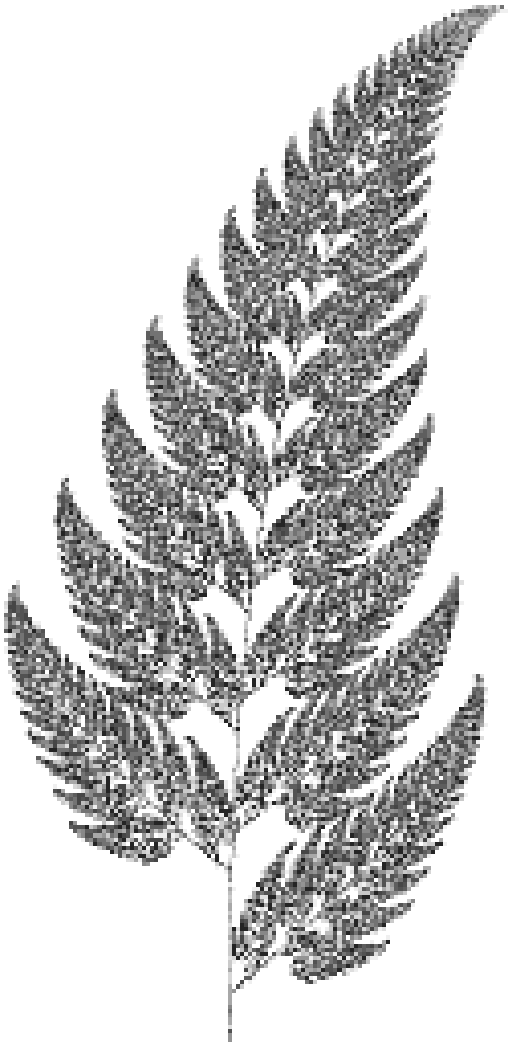


Figure A1.2. Fougère de Barnsley [2].

Annexe 2 : Exemples d'ondelettes

A2.1 Ondelette de Haar :

Historiquement, la première base orthogonale d'ondelettes est la base de Haar [111]. Son interprétation dans la théorie des ondelettes est récente. L'ondelette de Haar est définie par :

$$\psi(t) = \begin{cases} 1 & \text{si } t \in [0,1/2] \\ -1 & \text{si } t \in [1/2,1] \\ 0 & \text{sinon} \end{cases} \qquad \text{(A2.1)}$$

La base de Haar est constituée par la famille $\psi_{j,k} = \left\{2^{\frac{j}{2}}\psi(2^j t - k)\right\}_{j,k\in\mathbb{Z}}$. La fonction échelle associée est l'indicatrice de [0,1], $\phi(t) = \mathbf{1}_{[0,1]}(t)$.

Toute fonction *f* de $L^2(\mathbb{R})$ peut se décomposer sur la base de Haar, les coefficients étant donnés par :

$$d_{j,k} = \prec f(t)\psi_{j,k}(t) \succ = \int f(t)\psi_{j,k}(t)dt$$

De plus, on reconstruit *f* selon :

$$f(t) = \sum_{j,k\in Z} d_{j,k}\psi_{j,k}(t)$$

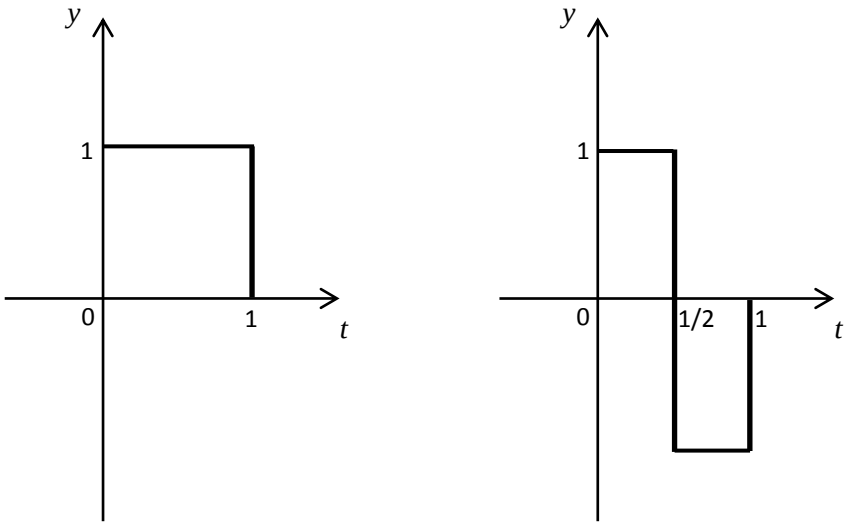
La base de Haar consiste à approcher une fonction par une somme de fonctions constantes par morceaux sur des intervalles de longueur 2^{-j}.
Dans l'analyse multirésolution, on a :

$$\mathrm{Pr}oj(f(t)/V_{j+1}) = \mathrm{Pr}oj(f(t)/V_j) + \sum_k d_{j,k}\psi_{j,k}(x) \qquad \text{(A2.2)}$$

Supposons que la fonction *f* soit dans V_1. Alors l'équation précédente se réécrit :

$$f(t) = \sum_k a_{0,k}\phi(t-k) + \sum_k d_{0,k}\psi(t-k) \qquad \text{(A2.3)}$$

Les ondelettes de Haar ont la propriété remarquable d'être symétrique et à support compact, puisqu'elles sont portées par l'intervalle [0,1].



La fonction échelle ϕ La fonction d'ondelette ψ

Figure A2.1: Ondelette de Haar

A2.2 Ondelettes de Daubechies :

Les ondelettes de Daubechies [117] orthogonales à support compact notées db_N sont indexées par un paramètre N et s'écrivent $\phi^N(t)$ (fonction échelle) et $\psi^N(t)$ (ondelette). N représente le nombre de coefficients servant pour l'étude, ce nombre est lié à la régularité de l'ondelette. L'ondelette de Haar est donc la première ondelette de Daubchies.

En effet on peut montrer que $\psi^N(t) \in C^{N-1}(R)$ où $C^N(R)$est l'ensemble des fonctions continues N-1 fois dérivables. Ainsi $\psi^N(t)$ pour N >1 est continue est peut être dérivée N-2 fois. De plus pour les ondelettes, cette notion est liée aux nombre de moments nuls de l'ondelette. Donc $\psi^N(t)$ vérifie :

$$\int t^n \psi^N(t)dt = 0, \ \forall n = 0,1,\ldots,N-1$$

Mentionnons que les filtres associés à $\phi^N(t)$ et $\psi^N(t)$ ont 2N coefficients non nuls, et que le support de la fonction échelle et de l'ondelette est de longueur 2N-1.

A2.3 Ondelettes bi-orthogonales symétriques :

Il est possible de construire des ondelettes à support compact symétrique. En effet, rien nous oblige à prendre les mêmes bases d'ondelettes pour la décomposition et la reconstruction, tant qu'on soit capable de récupérer la fonction (ou l'image) de départ. Aussi, on pourrait donner différentes propriétés aux ondelettes de décomposition et de reconstruction pour répondre aux besoins différents de ces deux étapes. Ainsi, à la décomposition, on vise à concentrer le signal dans seulement quelques coefficients, ce qui se

fait grâce à des ondelettes avec un nombre de moments élevé. Tandis que pour la reconstruction, on vise plutôt une ondelette symétrique et assez régulière pour permettre l'obtention d'une belle image.

La figure A2.2 montre la paire d'ondelettes bi-orthogonales CDF9/7 [129] utilisée comme base dans l'algorithme JPEG 2000 [127].

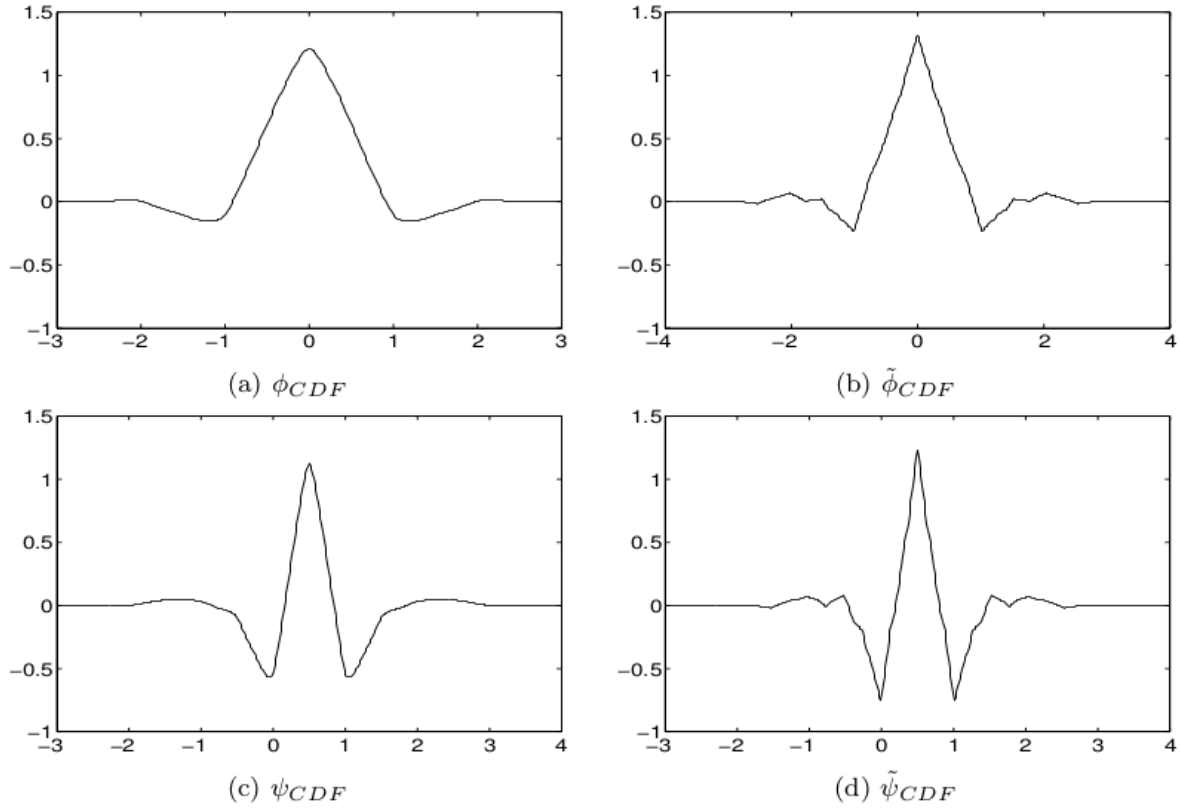


Figure A2.2 : Ondelettes bi-orthogonales CDF 9/7 [129].

Bibliographies:

[1] SALOMON David, Data Compression, The Complete Reference, Springer 2004. 833p.. ISBN978-0-387-21832

[2] BARNSLEY, M.F. and SLOAN, A.D., A Better way to compress images, *BYTE magazine* 1988, vol. 13 , n° 1, p.p. 215-223.

[3] JACQUIN, A.E., Image coding based on a fractal theory of iterated contractive image transformations, *IEEE Transactions on Image Processing*, January, 1992, vol. 1, n°1, p.18-30.

[4] FISHER, Yuval, Fractal image compression, Springer Verlag, 1995, 315 p, ISBN 978-1-4612-2472-3

[5] BARTHEL, K.U., SCHUTTEMEYER, J., VOYE, T. and NOLL, P, *A new image coding technique unifying fractal and transform coding*, IEEE International conference on image processing, November, 1994, p. 112-116.

[6] ZHAO, Y. and YUAN, B., A Hybrid image compression scheme combining block based fractal coding and DCT, *Signal processing: image communication*, March 1996, vol. 8, n° 2, p. 73-78.

[7] WELSTEAD, S., Fractal and wavelet Image Compression Techniques, SPIE Publications, 1999, 254p. ISBN 9780819435033.

[8] SHANNON, C. E., A Mathematical Theory of Communication, *Bell System Technical Journal*, July, October, 1948, vol. 27, pp. 379–423, 623–656.

[9] GRAY, R. M., Source coding theory", Kluwer, Academic Press, 1990.

[10] BATTAIL, G., Théorie de l'Information, Elsevier Masson, 1997, 396 p.

[11] RODIRIGUES José Marconi, Transfert sécurisé d'images par combinaison de techniques de compression, cryptage et marquage, Octobre 2006, Université de MONTPELLIER.

[12] ALEXANDRU Isar, CUBITCHI Andrei, and NAFORNITA Miranda, Algorithmes et techniques de compression, Editura ORIZONTURI POLITEHNICE, 2002.

[13] FANO, R. M., The transmission of information, *Technical Report* No. 65, 1949. *Research Laboratory of Electronics, M.I.T.*, Cambridge, USA.

[14] NELSON Mark, La compression des données, Edition Dunod, 1993, 420 p.

[15] HUFFMAN, D. A., A method for the construction of minimum redundancy codes, *Proc. of the IRE*, September 1952, vol. 40, p. 1098-1101.

[16] BENABDELLAH Mohammed, Outils de compression et de crypto-compression : Applications aux images fixes et vidéo, Juin 2007, Université MOHAMMED V, AGDAL.

[17] RISSANEN, J.J., Generalized Kraft Inequality and Arithmetic Coding, *IBM, J. Res.*, December 1976, vol 20, p.198-203.

[18] HOWARD, P. G. and VITTER J. S., Arithmetic coding for data compression, *Proc. of the IEEE*, 1994, vol. 82, n° 6, p. 857-865.

[19] PUIMATTO, G., Notions sur la compression des données dans les applications multimédias, *Note de l'ingénierie éducative*, CNDP, Septembre 1994 p.1-44.

[20] ZIV, J. LEMPEL A., A universal algorithm for sequential data compression, *IEEE Trans on information theory*, May 1977, vol. 23, n° 3, p. 337-343.

[21] ZIV J. and LEMPEL, A, Compression of individual sequences via variable rate coding, *IEEE Transactions on Information Theory*, September 1978, vol. 24, n° 5, p.530-536.

[22] WELCH, T., A technique for high performance data compression, *IEEE Computer*, 1984, vol.17, n° 6, p. 8-19.

[23] BONSTOCK Andrew, Practical Algorithms for Programmers, Addison-Wesley Professional, 1995, 577 p.

[24] JAIN K. Anil, Fundamentals of Digital Image Processing, Prentice Hall Information and System Sciences series, 1989.

[25] MUSMANN, H.G., Predictive Image Coding, 1979, In Image Transmission Techniques, W. K. Pratt (Ed.), Academic Press, New York.

[26] NETRAVAL, A.N., On quantizers for DPCM coding of picture signals, *Information theory*, 1977, vol. 23, p. 360-370.

[27] KYUNG, J.S., GSCHWIND, D.R. and BOSE, T., ADPCM encoding of images using a conjugate gradient based adaptive algorithm, *IEEE International Conference on Acoustics, Speech, and Signal Processing* (ICASSP-96), 1996, vol. 4.

[28] VANDER, R.A. and GRAY, R.M., Lossy compression of clustered dot halftones using sub-cell prediction, *In DCC Proc, Data Compression Conference*, 1995, p. 112-121.

[29] CLARKE, R.J., Transform Coding of Images, London, Academic Press, 1985, p.432.

[30] SIBADE Cédric, Compression de données pour les systèmes de traitement de document grand format, Thèse de Doctorat, 2003, Université de Marne la vallée.

[31] WATANABE, S., Karhunen-Loève expansion and factor analysis, *Trans. Fourth Prague Conf. on Information theory*, 1965.

[32] LIGHTHILL, M.J., Introduction to Fourier Analysis, Cambridge University Press, January 1958.

[33] AHMED, N., NATARAJAN, T. and RAO K.R., "Discrete Cosine transform", *IEEE Trans. On Computers*, 1974, vol C-23, p. 90-93.

[34] AKAMSU, A.N. and SMITH, M.J.T., Subband and Wavelet transforms: Design and Applications, Kluwer Academic Publishers, 1995, 412 p.

[35] RABBANI, P.W. Jones, Digital image compression techniques, SPIE Publications, 1991.

[36] MADIHALLY, S.N., NARASIMHA, J. and PETERSON, A.M., On the computation of Discrete Cosine Transform, *IEEE Trans. Com.*, 1978, vol. 26, n° 6, p. 934-936.

[37] CLARKE, R.J., Relation between the Karhunen Loève and cosine transforms, *IEE Proc.*, 1981, vol.128, Part F, n°.6, p.359-360.

[38] DUHAMMEL, P. and VETTERLI, M., Fast Fourier Transforms: A tutorial review and state of the art, *Signal Processing*, April 1999, vol. 19, pp. 259-299.

[39] ZIV, J., On universal quantization, *IEEE Trans. Inf. Theory*, 1985, vol. IT-31, n° 3, p.344-347.

[40] LIOYD, S.P., Least squares quantization in PCM, *IEEE transactions on Information Theory*, IT- March 1982, vol. 28, n° 2, p. 129-137.

[41] MAX, J., Quantizing for minimum distortion, *IRE Transactions on Information Theory*, IT- March 1960, vol. 6, p. 7-12.

[42] DU, Q., EMELIANENKO, M., and JU L., Convergence of the Lloyd algorithm for computing centroidal Voronoi tessellations, *SIAM Journal on Numerical Analysis*, 2006, vol. 44, n° 1, p. 102–119.

[43] GERSHO, A., On the Structure of Vector Quantizers, *IEEE Trans. Inform. Theory*, 1982, vol. 28, n° 2, p. 157-166.

[44] GRAY R.M., Vectors Quantization, *IEEE ASSP Mag*, 1984, vol. 1, n° 2, p.4-29.

[45] GERSO, A. and GRAY R.M., Vector Quantization and Signal Compression, Kluwer Academic Publishers, 1993, 689 p.

[46] LINDE Y., BUZO A. and GRAY R.M., An Algorithm for Vector Quantizer Design, *IEEE Transactions on Communications*, 1980, vol. 28, p. 84-94.

[47] VALADE Cédric, Compression d'images complexes avec pertes : application à l'imagerie radar, Thèse de doctorat, 2006, Ecole Nationale supérieure des télécommunications.

[48] BEAUREPAIRE, Patricia, Compression d'images appliquée aux Angiographies cardiaques, Aspects algorithmiques, evaluation de la qualité diagnostique, Thèse de Doctorat, 1997, Institut National de Sciences Appliquées de LYON.

[49] JPEG. "Joint Photographic Experts Group, JPEG Homepage", http://jpeg.org/jpeg/index.html

[50] PENNEBAKER, W.B. and MITCHELL J.L., JPEG Still Image Data Compression Standard, New York: Van Nostrand Reinhold, 1993.

[51] HP Labs LOCO-I/JPEG-LS Home Page". Hewlett-Packard Development Company. May 13, 2005. http://www.labs.hp.com/research/info_theory/loco/,

[52] WEINBERGER, M J., SEROUSSI, G., and SHAPIRO, G., LOCO-I: A low complexity, context-based, lossless image compression algorithm, *in Proc. Data Compression Conference*, March 1996, p. 140–149.

[53] WEINBERGER, M.J., SEROUSSI, G., and SHAPIRO, G., LOCO-I: new developments, November 1994 ISO Working Document ISO/IEC JTC1/SC29/WG1 N245.

[54] SHI, Y.Q. and HUIFANG S., Image and Video Compression for Multimedia Engineering: Fundamentals, Algorithms, and Standards, 2008, CRC press, 576 p., ISBN 9780849373640.

[55] RICHARDSON, E.G. Lain, Video Codec Design: Developing Image and Video Compression Systems, 2002, John Wiley & Sons Inc., 314 p., ISBN: 978-0-471-48553-7.

[56] MANDELBROT Benoît, Fractals form, chance, and dimension, W.H. Freeman and Company, 1977, 365 p.

[57] JULIA, G., Mémoire sur l'itération des fonctions rationnelles, *Journal de Mathématiques Pures et Appliquées*, 1918, vol. 8, p. 47-245.

[58] VON KOCH, Sur une courbe continue sans tangente, obtenue par une construction géométrique élémentaire, *Arkiv för matematik, astronomi och fysik*, 1904, vol. 1, p. 681-704.

[59] HUTCHINSON, J., Fractal and self-similarity, *Journal de Mathématique de l'université Indiana*, 1981, vol. 30, p. 713-747.

[60] BAENESLEY, Michael. F, Fractal Everywhere, Morgan Kaufmann Publishers, 1st Edition, 1988.

[61] CHRISTOPHE Portefaix, Modélisation des signaux et des images par les attracteurs fractals de systèmes de fonctions itérées (IFS), Thèse de Doctorat, 2004, Université D'ANGERS.

[62] REUSENS, E., Partitioning complexity issue for iterated functions systems based image coding, *In Proc. Of VII European signal processing conference*, Edinburg, U.K., September, 1994, vol.1, p.171-174.

[63] SAUPE, D., RUHL, M., HAMZAOUI, R., GRANDI, L. and MARINI, D., Optimal hierarchical partitions for fractal image compression, *IEEE International conference on image processing* Chicago, October 1998.

[64] HARTENSTEIN, H., RUHL, M., and SAUPE, D., Region based fractal image compression, *IEEE Trans. on Image processing*, 2000, vol.9, p.1171-1184.

[65] FISHER Yuval, Fractal image compression: theory and application to digital images, 1995, Springer Verlag, New York.

[66] HÏRTGEN, B. and STILLER, C., Fast hierarchical codebook search for fractal coding of still images, *SPIE proceeding, Video commun. PACS Med. Applic*, April 1993, vol.1977, p. 397-408.

[67] POLVERE, M. and NAPPI, M., Speed-up in fractal image coding: Comparison of methods, *IEEE Transactions on image processing*, June 2000,vol.9, n°.6, p. 1002-1009.

[68] GHARAVI-ALKHANSARI M., and HUANG, T.S., A fractal based image block coding algorithm, *In Proceedings of ICASSP*, 1993, p. 345-348.

[69] LIN, H. and VENETSANOPOULOS, A.N., Incorporating nonlinear contractive functions into the fractal coding, *In Proc. Int. Workshop on Intelligent signal processing and communication systems*, October 1994, p. 169-172.

[70] BAHARAV, Z., MALAH, D. and KARNIN, E., Hierarchical interpretation of fractal image coding and its application to fast decoding, *in Proc IEEE International conference on digital signal processing*, Nicosi, Cyprus, July 1993, p.190-195.

[71] DUBRIDGE, F., Least squares block coding by fractal functions, in Fractal image compression: Theory and applications to digital images, Y. Fisher (ed), Springer Verlag, 1995.

[72] HAMZAOUI, R., A new decoding algorithm for fractal image compression: ftp.informatik.uni-freiburg.de, 1996.

[73] GHARAVI- ALKHANSARI, M., Fractal based image and video coding using matching pursuit, Phd Thesis, 1997, University Illinois, Urbana-Champaign.

[74] MUKHERJEE, J. Mukherjee, KUMAR, P. and GHOSH, S.K., A Graph theoretic approach for studying the convergence of fractal encoding algorithm, *IEEE Trans on image processing*, March 2000, vol.9, n°.3, p.366-376.

[75] MAZAL, D.S. and HAYES, M.H., Using iterated function systems to model discrete sequences, *IEEE Trans on signal processing*, 1992, vol.40, n°.7, p. 1724-1734.

[76] HURD, L. P.,GUSTAVU, M. A. and BARNESLEY, M.F., Fractal video compression, *compcon Springer*, conf.37, 1992, p.41-42.

[77] LAZAR, M.S. and BRUTON, L.T.,Fractal Block Coding of Digital video, *IEEE Transactions on circuits and systems for video technology*, June 1994, vol.4, n° 3, p. 297-838.

[78] SAUPE, D., HAMZAOUI, R. and HARTENSTEIN, H., Fractal image compression: an introductory overview, Technical Report, 1997.

[79] WOHLBERG, B. and DE JAGER, G., A Review of the fractal image coding literature, *IEEE Trans on image processing*, December 1999, vol.8, n° 12, p.1716-1729.

[80] KOLI, N.A. and ALI, M.S., A survey on fractal image compression key issues, *Information technology Journal*, 2008, vol.7, n° 8, p.1085-1095.

[81] BARLAUD, M. and LABIT, C., Compression et codage des images et des vidéos, Janvier 2002, Paris, Hermès.

[82] SAMET, H., Region representation: quadtrees from binary arrays, *CVGIP*, 1980, vol. 13, p. 88-93.

[83] FISHER, Y. and MENLOVE, S., Fractal encoding with HV partitions, In Fisher [60], 1995, p. 119-136.

[84] FORTUNE, S., Voronoi diagrams and Delaunay triangulation, Handbook of Discrete and computational Geometry, 1995.

[85] CHASSERY, J.M., DAVOINE, F. et BERTIN, E. Bertin, Compression fractale par partitionnement de Delaunay, *14 ème coloque GRETSI*, Septembre 1993, p. 819-822.

[86] AGEN, F, MICHOT, J., Projet C: La compression Fractale, Méthodes de Jacquin, Subdivision de trianges et Delaunay, 2005, webfractales.free.fr.

[87] DAVOINE, F. et CHASSERY, J.M., Un partitionnement adaptatif en triangles et quadrilatères pour la compression des images par fractales, *15ème colloque GRETSI*, Septembre 1995, p.721-724.

[88] ROBERT, G., DAVOINE, F. et CHASSERY, J.M., Compression d'images par fractales à base de maillages polygonaux de Voronoï, *16ème Colloque GRETSI*, Septembre 1997, p.179-182.

[89] THOMAS, L. and DERAVI, F., Region-based fractal image compression using heuristic search, *IEEE Transactions on Image Processing*, 1995, vol. 4, n° 6, p.832–838.

[90] CHANG, Y.C., SHYU, B.K., and WANG, J.S., Region-based fractal compression for still image, *In Proc. WSCG'2000, The 8-th International Conference in Central Europe on Computer Graphics, Visualization and Interactive Digital Media*, 2000, Plzen, Czech Republic.

[91] OCHOTTA T. and SAUPE, D., Edge-based partition coding for fractal image compression, *The Arabian Journal for Science and Engineering, Special Issue on Fractal and Wavelet Methods*, 2004, vol. 29, n° 2C.

[92] DAVOINE, Frank, Compression d'images par fractales basée sur la triangulation de Delaunay, Thèse de Doctorat, 1995, Institut National Polytechnique de Grenoble.

[93] RAMAMURTHI, B. and GERSHO, A., Classified vector quantization of images, *IEEE Trans. on communications*, 1986, vol.34, n°11, p.1105-1115.

[94] JACQUIN, A., Fractal image coding: A review, *Proceeding of the IEEE*, October 1993, vol.81, n° 10, p.1451-1465.

[95] JACOBS, E. W., BOSS, R. D. and FISHER, Y., Fractal based image compression, Technical Report 1362, *Naval Ocean systems center*, CA 92152-5000, June 1990.

[96] HAMZAOUI, R. and SAUPE, D., Combining fractal image compression and vector quantization, *IEEE Trans on Image processing*, 2000, vol. 9, n° 2, p.197-208.

[97] DUH, D. J., JENG, J.H. and CHEN, S.Y., DCT based simple classification scheme for fractal image compression, *Image and vision computing*, 2005, vol.23, p.1115-1121.

[98] LEPSOY, S., Attractor image compression-fast algorithms and comparisons to related techniques, Phd Thesis, June 1993, The Norwegian Institute of Technology.

[99] DAVOINE, F., ANTONINI, M., CHASSERY, J.M. and BARLAUD, M., Fractal image compression based on Delaunay triangulation and vector quantization, IEEE *Trans. on image processing*, Feb 1996, vol. 5, n° 2, p.338-346.

[100] BOSS, R. D. and JACOBS, E. W., Archetype Classification in an Iterated Transformation Image Compression Algorithm, in Fractal Image Compression: Theory and Applications to Digital Images, Yuval Fisher, Ed. Springer-Verlag, New York, 1995.

[101] MONRO, D. M. and DUBRIDGE, F., Approximation of image blocks, *International conference on acoustic, speed and signal processing*, 1992, vol.3, p.4585-4588.

[102] SUAPE, D., Accelerating fractal image compression by multi-dimensional nearest neighbor search, *Data compression conference* (DCC'95), March 1995.

[103] ARYA, S., MOUNT, D.M., NETANYAHU, N.S., SILVERMAN, R., and WU, A. An optimal algorithm for approximate nearest neighbor searching, *in Proc. 5th annual ACM-SIAM Symposium on discrete algorithms*, 1994, p.573-582.

[104] SAUPE, D., Lean domain pools for fractal image compression, *Proceedings IS&T/SPIE1996 Symposium on Electronic Imaging: Science & Technology Still Image Compression* II, June 1996, vol.2669.

[105] HASSABALLAH, M., MAKKY, M.M. and MAHDY, Y., A fast fractal image compression method based entropy, *Electronic letters on computer vision and image analysis*, 2005, vol.5, n° 1, p.30-40.

[106] DISTASI, R., NAPPI, M. and RICCIO, D., A Range/Domain Approximation Error-Based Approach for Fractal Image Compression, *IEEE Trans. on Image Processing*, 2006, vol.15, n° 1.

[107] MELNIKOV, G. and KATSAGGELOS, A.K., A jointly optimal fractal/DCT compression scheme, *IEEE Trans. on Multimedia*, 2002, vol.4, p.413-422.

[108] AVANAKI Mohammad R. N., ABER, A. and EBRAHIMPOUR, Compression of cDNA Microarray Images based on pure-fractal and wavelet-fractal techniques, *ICGST-GVIP Journal*, 2001, vol. 11, n°1, p.43-52.

[109] COSTA, S. and FIORI, S., Image compression using principal component neural networks, *Image vision computing*, 2001, vol. 19, p.649-668.

[110] SUN, Y. D. and ZHAO, Y., A parallel implementation of improved fractal image coding based on tree topology, *Chinese J. Elec*, 2003, vol.12, p. 152-156.

[111] HAAR, A.Z., Theorie der orthogonalen Funktionensysteme, *Mathematische Annalen*, 1910, vol.69, p.331–371.

[112] GABOR, D., Theory of communication, *J. Inst. Elect. Eng*., 1946, vol.93, p.429-457.

[113] GOUPILLAUD, P, GROSSMANN, A. and MORLET, J., Cycle-octave and related transforms in seismic signal analysis, *Geoexploration*, 1985, vol.23, p. 85–102.

[114] MEYER, Y., Wavelets and Applications, Number 20 in Research notes is Applied Mathematics, Springer Verlag, 1991.

[115] MALLAT, S. G.t, A theory for multiresolution signal decomposition: The wavelet representation, *IEEE Trans. Patt. Anal. Mach. Intell*., 1989, vol. 11, n° 7, p. 674-693.

[116] LEMARIE, P. G., Les Ondelettes en 1989, Number 1438 in Lecture Notes in Math. Springer-Verlag, 1990.

[117] DAUBECHUES, I., The wavelet transform, time-frequency localization and signal analysis, *IEEE Trans. Inform. Theory*, 1990, vol. 36, n° 5, p. 961-1005.

[118] CHUI, C. K., An Introduction to Wavelets, Academic Press Professional, Inc. San Diego, CA, USA, 1992

[119] WORNELL, G. W., A Karhunen-Loieve-like Expansion for 1/f. Processes via Wavelets, *IEEE Trans on Information Theory*, 1990, vol.36, n°4, p. 859-861.

[120] HOLSCNEIDER, M., Wavelets: An Analysis Tool, Oxford University Press, USA, 1999.

[121] BOUCHEREAU Ammanuelle Bournay, Analyse d'images par transformées en ondelettes, Apllication aux images sismiques, Thèse de Doctorat, 1997, Université Joseph FOURIER, GRENOBLE I.

[122] WICKERHAUSER, M. V., Comparison of Picture Compression Methods, Wavelets: theory, Algorithms and Applications, Academic Press, San Diego, 1994.

[123] GASQUET, C. and WITOMSKI, P., Analyse de Fourier et applications, Dunod, 1996.

[124] BARLAUD, M., SOLE, P., GAIDON, T., ANTONINI, M., and MATHIEU, P., "Pyramidal lattice vector quantization for multiscale image coding", *IEEE Transactions on Image Processing*, 1994, vol.3, n°4, p.367-381.

[125]SHAPIRO, J. M., Embedded image coding using zerotrees of wavelet coefficients, *IEEE Trans. On Signal Processing*, December 1993, vol.41, n° 12, p.3445-3462.

[126] SAID A. and PEARLMAN, W. A., A new, fast and efficient image codec based on set partitioning in hierarchical trees, *IEEE Trans. On Circuits Systems. Video technology*. June 1996, vol. 6,n°.3, p. 243-250.

[127] ISO/IEC 15444-1, Information Technology - JPEG2000 Image Coding System-Part 1: Core Coding System, 2000.

[128] CHRISTOPOULOS, C., SKODRAS, A. and ABRAHIMI, T., "The JPEG2000 Still Image Coding System : An Overview", *IEEE Transactions on Consumer Electronics*, November 2000, vol. 46, n° 4, p.1103-1127.

[129] COHEN, A., DAUBECHIES, I. and FEAUVEAU, J.C., Biorthogonal bases of compactly supported wavelets, *Commu Pure Appli-Math*, 1992, vol.45, n°5, p. 485-560.

[130] LE GALL D. and TABATABI, A., Sub-band coding of digital images using symmetric short kernel filters and arithemetic coding techniques, *Proc. Int. Conf. Acoust. Speech and Signal Processing*, April 1988, p. 761-764.

[131] TAUBMAN, D., High Performance scalable image compression with EBCOT, *IEEE Trans. On Image Processing*, July 2000, vol. 9, n°7, p. 1158-1170.

[132] PENTLAND, A. and HOROWITZ, B., A practical approach to fractal based image compression, *Proceeding of Data Compression Conference (DCC)*, 1991.

[133] MONRO, D.M. and DUBRIDGE, F., Fast image coding by hierarchical fractal construction, In preprint of the UCSD, San diego 1995.

[134] DAVIS, G. M., Self-quantized wavelet subtrees: A wavelet based theory of fractal image compression, *In H. H. SZU, editor, wavelet applications II, 2491 of SPIE proceedings*, April 1995, vol. 2491, p.141-152.

[135] KRUPNIK, H., MALAH, D. and KARNIN E., Fractal representation of images via the discrete wavelet transform, *In proceeding of the 18th IEEE convention of Electrical and Electronics Engineers*, 2.2.2./1-5, March 1995, p. 1-5.

[136] VAN DE WALL A., Merging fractal image compression and wavelet transform methods. *In Nato ASI on Fractal Image Encoding and Analysis*, July 1995, p. 8-17.

[137] SIMON, B., "Explicit link between local fractal transform and multiresolution transform". *IEEE international conference on Image Processing*, October 1995, vol.1, p.278-281.

[138] RINALDO, R. and CALVAGNO, G., Image coding by block prediction of multiresolution subimages, *IEEE Trans. On Image Processing*, July 1995, vol.4, n°7, p.909-920.

[139] DAVIS, G. M., A Wavelet Based Analysis of Fractal Image Compression, *IEEE Transaction on Image Processing*, 1998, vol.7, n°2, p. 141-154.

[140] LI, J. and KUO, C.C.J., Image compression with a hybrid wavelet-fractal coder, *IEEE Trans. on Image. Processing*, June 1999, vol. 8, n° 6, p. 868-873.

[141] ZHANG, L. and XI, L., Hybrid image compression using fractal-wavelet prediction, *In proceedings of the 5th WSEAS International conference on Information Security and Privacy*, 2006, p.112-117.

[142] KIM, T., VAN DYCK, R.E. and MILLER, D.J., Hybrid Fractal Zerotree Wavelet Image Coding, *Elsevier science, Signal Processing. Image Communication*, 2002, vol. 17, n°4, p.347-360.

[143] IANO, Y., DA SILVA, F.S. and CRUS, A.L., A Fast and Efficient Hybrid Fractal-Wavelet Image Coder, *IEEE Transaction Image Processing*, 2006, vol. 15, n°1, p.98-105.

[144] DURAISAMY, R., VALARMATHI, L. and AYYAPPAN, J., Iteration Free Hybrid Fractal-Wavelet Image Coder, *International Journal of Computational Cognition*, 2008, vol.6, n° 4, p.34-40.

[145] CHANG, H.T., KUO, C.J., Iteration free fractal image coding based on efficient domain pool design, *IEEE Trans. On. Image Process*, 2000, vol. 9, n° 3, p.329-339.

[146] MEKHALFA F., AVANAKI, M.R.N. and BERKANI, D., A Lossless Hybrid Wavelet-Fractal Image Compression for Welding Radiographic Images, *Journal of X-Ray Science and Technology*, Feb 2016, vol. 24, n° 1, p.107-118.

[147] MEKHALFA, F. and BERKANI D., Hybrid Wavelet-Fractal Image Coder Applied to Radiographic images of Weld Defects, *Digital Information and Communication Technology and Its Applications Volume 166 of the series Communications in Computer and Information Science, Dijon, 2011, p.753-761.*

[148] COSMAN, P.C., GRAY, R.M. and OLSHEN, R.A., Evaluating quality of compressed medic images: SNR, subjective rating, and diagnostic accuracy, *Proceedings of the IEEE*, 1994, vol. 82, n° 6, p. 919-932.

[149] ROGERSON, J. H., Defects in welds: Their prevention and their significance, Applied science publishers, 1983.

[150] DA SILVA, N., CALÖLA, L., SIQUEIRA, M. and RBELLO, J., Pattern Recognition of Weld Defects Detected by Radiographic Test, *NDTE International journal*, 2004, vol.37, n° 6, p. 461-470.

[151] MEKHALFA, F., NACEREDDINE, N. and GOUMEIDANE, A.B., Unsupervised Algorithm for Radiographic Image Segmentation Based on the Gaussian Mixture Model, *The International Conference on Computer as a Tool (EUROCON)*, pp.289-293, Warsaw, Sep 2007.

[152] MEKHALFA, F. and NACEREDDINE, N., Multiclass Classification of Weld Defects in Radiographic Images based on Support Vector Machines, *10th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS)*, Marrakach, Nov 2014, p.1-6.

[153] MEKHALFA, F. and BERKANI, D., Comparison of Fractal Compression Methods Impact on Radiographic Images of Weld Defects, *IEEE International Conference on Communications, Signals and Coding (MIC-CSC2)*, Istanbul, Oct 2012, p.41-46.

[154] NACEREDDINE Nafaa, Mise au point d'une méthode de détection et classification automatisé des défauts de soudure en radiographie industrielle. Thèse de magister, 2004, Université de Boumerdès.

[155] MEKHALFA, F. and BERKANI, D., Application of Hybrid Wavelet-Fractal Compression algorithm for Radiographic Images of Weld Defects, International Journal of New Computer Architectures and their Applications (IJNCAA), 2011vol. 1, n° 1, p.123-132.

[156] MEKHALFA, F. and BERKANI, D., Compression of radiographic images of weld defects based on fractal and wavelet techniques, *Journées d'Etudes Algéro-Françaises de Doctorants en Signal-image and Applications (JEAFD)*, Dec 2012, ENP, p.120-125.

[157] DEMPSTER, A. P., LAIRD, N. M., RUBIN, D. B., Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Ser. B*, 1977, vol. 39, p. 1-38.