Mémoire de master de recherche
« Architectures logicielles distribuées »

# Quantification et codage après transformée en ondelettes orientées

## *Application à la compression d'images fixes*

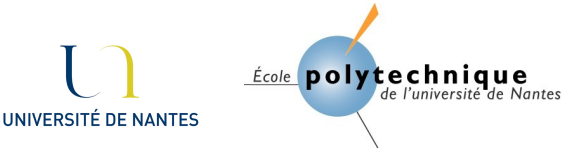
Matthieu MOINARD

4 Septembre 2007

encadré par Vincent RICORDEL

— IVC —

INSTITUT DE RECHERCHE EN COMMUNICATIONS ET EN CYBERNÉTIQUE DE NANTES



# Quantification et codage après transformée en ondelettes orientées

## Application à la compression d'images fixes

Matthieu MOINARD

**Résumé**

Ce rapport expose les travaux réalisés portant sur les stratégies de quantification et de codage, dans un contexte de transformée en ondelettes orientées. Dans le but d'adapter la quantification des coefficients au contenu local de l'image, nous proposons une étape de segmentation et de classification de l'image. L'information supplémentaire issue de cette étape est codée selon une méthode que nous exposons. Les résultats des stratégies de quantification et de codage que nous expliquons en détails sont fournis à la fin du rapport. La qualité visuelles des images reconstruites est mesurée par la métrique de qualité objective Komparator, développée au sein de l'équipe IVC.

Catégories et descripteurs de sujets : I.4.2 [**Compression (Coding)**]: Approximate methods; I.4.6 [**Segmentation**]: Pixel classification; I.4.10 [**Image Representation**]: Hierarchical

Termes généraux : Compression d'images, optimisation visuelle, modèle perceptuel.

Mots-clés additionnels et phrases : Transformée en ondelettes, lifting, quantification, codage, SPIHT, système visuel humain, lifting orienté basé contours, JPEG2000, masquage visuel.

# Remerciements

BLABLABLA

# Table des matières

# Chapitre 1

# Introduction

## Sommaire



La représentation numérique des images est un domaine de recherche à part entière qui suscite désormais beaucoup d'attention. L'image est un média à fort contenu sémantique, elle est devenue un moyen de communication à part entière de plus en plus présent dans notre vie quotidienne. L'intérêt récent du grand public pour l'imagerie numérique, au travers des appareils photos numériques, des téléphones portables ou même le partage et la divulgation sur internet, montre que les problémes liées à sa représentation, son stockage et sa transmission sont des sujets fort d'actualité.

Le stage de recherche se déroule au sein de l'équipe IVC du laboratoire IRCCyN, entouré par tout le personnel enseignant-chercheur. L'équipe IVC est à ce jour sous la direction de Patrick Le Callet, et mon stage est encadré par Vincent Ricordel, Guillaume Jeannic et Dominique Barba. Le stage a une durée globale de six mois, de début mars à début septembre 2007.

## 1.1 Présentation de la problématique

Le traitement des images fixes numériques est un domaine de recherche très actif depuis l'apparition des premiers procédés d'acquisition d'image. Depuis, les recherches se sont étendues et de nombreux axes de recherches plus ou moins récents se sont développés :

- Tatouage d'image
- Reconnaissance des formes et des images
- Segmentation et classification des images
- Compression avec ou sans pertes

- Transformation des images
- Transmission des images
- ...

Le sujet de stage qui m'a été proposé concerne la compression d'images fixes. Cet axe de recherche captive de nombreux chercheurs de par le monde, et a abouti récemment à l'élaboration conjointe de la norme JPEG2000 pour les applications industrielles. Grâce à cela, de nombreuses recherches connexes ont permis d'améliorer le procédé de la compression d'images. Notamment, Guillaume Jeannic et Vincent Ricordel ont mis au point une nouvelle méthode de transformation d'images très prometteuse, puisqu'elle concentre l'énergie du signal d'origine (l'image). Ce procédé permet de hiérarchiser l'information de façon plus pertinente que JPEG2000. Les travaux de Guillaume Jeannic et Vincent Ricordel sur la transformation en ondelettes orientées d'images fixes offrent des possibilités de compression très avantageuses. Cependant, cette étape de transformation représente que la première partie du schéma de global de compression. S'en suivent ensuite les étapes de quantification et de codage (i.e. figure 2.1). Mon stage consiste donc à concevoir et à réaliser ces deux étapes, pour une compression basée sur la transformation en ondelettes orientées précédemment citées. Pour cela, les recherches réalisées pendant le stage devront porter sur les techniques actuelles de quantification et de codage. Le but sera d'adapter ce qui existe dans le cadre de la nouvelle transformation en tirant profit des modifications qu'elle représente.

L'étude du système visuel humain constitue un axe de recherche récent, et laisse entrevoir des applications concrètes dans le domaine de la compression d'images. Plusieurs chercheurs reconnus ont d'ores et déjà proposé des procédés de compression basés sur ces études. Dans le cadre du stage, cette nouvelle approche sera exploitée pour améliorer les performances en terme de réduction d'information.

La problématique même du sujet qui nous concerne est de combiner la transformée en ondelette orientée, les méthodes issues de l'étude du SVH et l'information *a priori* que nous avons sur le contenu du l'image. En effet, un des intérêts de la transformée orientée présentée par Guillaume Jeannic est qu'elle s'appuie sur le contenu local de l'image. De ce fait, nous pouvons nous demander s'il est possible d'interpréter ces informations pour les étapes de quantification et de codage.

## 1.2 Objectifs poursuivis

La compression d'images fixes consiste à réduire le débit, c'est l'objectif principal que nous poursuivrons. Pour cela, nous allons devoir travailler sur de nouveaux algorithmes issus des récentes recherches dans le domaine. Nous devons également travailler sur les services à fournir, principalement la scalabilité du flux de données qui permet d'obtenir, par un unique processus d'encodage, un train binaire décodable à de multiples niveaux de qualité et à différentes résolutions. Cette fonctionnalité correspond aux nouveaux besoins des applications émergentes et étoffe la gamme des services disponibles dans les nouveaux systèmes de compression. Enfin, nous devrons

être vigilant quant à la complexité des algorithmes, pour proposer des solutions peu coûteuses en temps d'exécution. En effet, quand bien même les algorithmes que nous proposons sont très performants en terme de compression, ils deviennent inutilisables si le procédé de compression/décompression est trop long.

## 1.3 Travail à réaliser

Nous travaillons dans un contexte de compression d'images fixes. Pour réduire le débit, les algorithmes sont de plus en plus complexes, et se basent désormais sur des caractéristiques propres au SVH[1] pour supprimer l'information non perceptible ou peu perceptible. Ces récentes études vont nous inspirer pour proposer un schéma de compression différent.

Le schéma de compression classique est décomposé en trois étapes successives :

1. La transformation pour passer d'une représentation où chaque pixel de l'image apporte la même quantité d'information à un modèle hiérarchisé.
2. La quantification qui consiste à approcher une valeur continue par une valeur comprise dans un ensemble fini d'assez petite taille. C'est au niveau de la quantification que s'effectue la réduction de la précision des coefficients ondelettes, ce qui rend l'opération irréversible.
3. Le codage qui génère le flux binaire selon différentes stratégies en fonction des services requis.

À la phase de décodage, le parcours inverse est effectué, à savoir le décodage du flux binaire, la quantification inverse et la transformation inverse. Ces opérations sont schématisées sur la figure 2.1.

Or, le travail réalisé par Guillaume Jeannic a modifié la méthode de transformation, pour qu'elle s'adapte localement à l'image. De ce fait, de nouvelles possibilités s'offrent à nous pour la quantification et le codage.

Notre travail peut se résume ainsi : après une transformation *lifting* orienté, adapter une étape de quantification et de codage en s'inspirant des concepts existants tout en exploitant les caractéristiques intrinsèques de la transformée.

## 1.4 Contribution

La contribution que j'ai apportée au cours de ce stage porte sur plusieurs aspects. En reprenant l'ordre chronologique établi par le schéma de compression, j'ai travaillé sur :

- le codage de l'information additionnelle nécessaire au procédé de décompression ;
- la quantification selon le contenu local de l'image et l'étude du SVH ;
- l'amélioration du codeur SPIHT pour le cas qui nous concerne.

Pour chacun de ces points, j'ai implanté les algorithmes qui ont permis d'expérimenter et de valider les résultats.

[1] Système Visuel Humain

## 1.5 Plan de l'étude

Ce rapport est décomposé en trois parties majeures. Dans un premier temps, nous allons nous intéresser à l'état de l'art du domaine, en se focalisant sur les procédés qui vont nous inspirer par la suite. La deuxième partie est consacrée aux propositions que nous avons formulé, c'est-à-dire au nouveau schéma de compression que nous avons élaboré. Enfin, nous exposerons les résultats de notre travail.

# Chapitre 2

# État de l'art

## Sommaire



Dans ce chapitre, nous faisons part de nos recherche bibliographiques qui vont nous concerner par la suite. La première partie est dédiée à l'introduction au domaine de la compression d'images fixes, et plus particulièrement à JPEG2000. Ensuite, nous étudierons l'algorithme SPIHT sur lequel nous allons entre autre baser nos propositions. Dans un deuxième temps, nous analyserons les techniques avancées pour la compression d'images fixes, apparues plus récemment.

Nous nous focaliserons sur des méthodes qui vont exclusivement nous servir pour notre propre schéma de compression. Pour une étude complète du travail de bibliographie ou pour un complément d'information sur les notions évoquées dans ce chapitre, vous pouvez vous référer à un rapport bibliographique constitué précédemment.

## 2.1 Introduction

Grand successeur de la DCT[1] utilisée dans la norme JPEG, la technologie de compression à base d'ondelettes offre une plus grande finesse au niveau de l'analyse du signal et permet de mieux s'adapter aux propriétés locales de l'image. C'est un axe de recherche très prometteur puisqu'il permet d'obtenir, à qualité d'image

[1] Discrete Cosine Transform

identique, des taux de compression variables bien supérieurs au JPEG standard (de l'ordre de 50 :1 par rapport à l'image d'origine). La compression à base d'ondelettes est la base du système de compression JPEG2000.

JPEG 2000 ou ISO/CEI 15444-1 est une norme commune de l'ISO (l'organisme international de normalisation) et de l'UIT-T (l'union internationale des télécommunications). C'est un standard de compression d'images défini par le comité *Joint Photographic Experts Groups*. En plus d'être plus performant en terme de compression d'images fixes que JPEG, JPEG 2000 offre de nombreux services supplémentaires, tels que [Saadane, 2000] la transmission progressive par qualité ou par résolution, l'accés et le traitement aléatoire du flux binaire (accès aux régions d'intérêt, rotation du domaine compressé, etc.), la robustesse aux erreurs, etc.

La norme JPEG 2000 est divisée en douze sous-parties, comme ceci :

1. Part 1 « Core Coding System » : le standard avec un minimum d'options pour pouvoir être intégré dans différents dispositifs : imprimantes, appareil photo, etc. ;
2. Part 2 « Extensions » : pour toutes les applications de compression où l'échange est moins important ;
3. Part 3 « Motion JPEG2000 » : extensions aux séquences d'images.
4. Part 4 Conformité : normalisation des différentes étapes du décodeur ;
5. Part 5 « Reference software » : il existe trois outils de développement de référence pour le format JPEG2000, à savoir JJ2000, JasPer et Kakadu ;
6. Part 6 « Compound image file format » : format additionnel pour les documents composés ;
7. Part 7 (inutilisée) ;
8. Part 8 JPSEC : gère l'aspect sécurité ;
9. Part 9 JPIP : protocoles intéractifs et API ;
10. Part 10 JP3D : étend aux images tri-dimensionnelles ;
11. Part 11 JPWL : pour les applications sans-fils ;
12. Part 12 « ISO Base Media File Format » : commun avec MPEG-4.

Comme toute architecture standard d'un système de compression, le codage JPEG2000 se décompose en trois étapes successives :

1. La transformation pour passer d'une représentation où chaque pixel de l'image apporte la même quantité d'information à un modèle hiérarchisé.
2. La quantification qui consiste à approcher une valeur continue par une valeur comprise dans un ensemble fini d'assez petite taille. C'est au niveau de la quantification que s'effectue la réduction de la précision des coefficients ondelettes, ce qui rend l'opération irréversible.
3. Le codage qui génère le flux binaire selon différentes stratégies en fonction des services requis.

À la phase de décodage, le parcours inverse est effectué, à savoir le décodage du flux binaire, la quantification inverse et la transformation inverse. Ces opérations sont schématisées sur la figure 2.1.

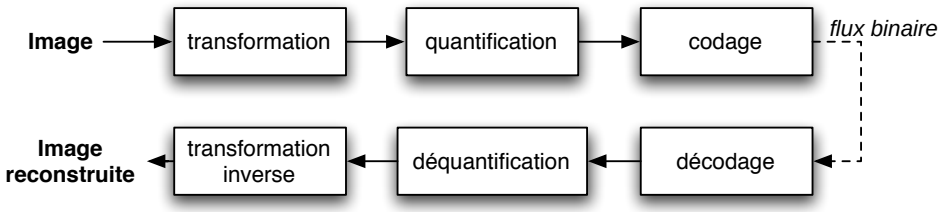



FIG. 2.1 – Schéma de compression/décompression d'images fixes classique

## 2.2 Algorithme SPIHT

Après la quantification des coefficients ondelettes, la phase suivante du système de compression des données est le codage entropique. Cette étape a pour rôle d'organiser le flux des données compressées afin de hiérarchiser l'information. Une caractéristique intéressante des codeurs est la notion de scalabilité (ou graduabilité), qui permet d'obtenir, par un unique processus d'encodage, un train binaire décodable à de multiples niveaux de qualité et à différentes résolutions. Cette fonctionnalité correspond aux nouveaux besoins des applications émergentes et étoffe la gamme des services disponibles dans les nouveaux systèmes de compression. Il existe deux algorithmes de codage très répandus nommés SPIHT [Said et Pearlman, 1996] et EBCOT [Taubman, 1998], qui correspondent à deux stratégies différentes. Nous allons étudier SPIHT en détail.

L'algorithme SPIHT [2] [Said et Pearlman, 1996] ordonnance de façon partielle l'information tout en ajoutant peu d'information supplémentaire.

L'algorithme SPIHT propose une amélioration de l'algorithme EZW [3] [Shapiro, 1993] tout en conservant ses propriétés qui sont :

- de bonnes performances ;
- si le flux binaire produit est interrompu ou tronqué, la reconstruction de l'image de façon partielle est possible.

SPIHT est basé sur un ordonnancement partiel par amplitude via un algorithme de tri de partitions, et en exploitant les similarités présentes à différents niveaux de l'image transformée en ondelettes.

### 2.2.1 Transmission progressive de l'image

L'objectif de la transmission progressive de l'image est de sélectionner l'information la plus importante pour la transmettre en premier. L'information la plus importante et celle qui apportera, lors de sa réception, la correction la plus importante de la distorsion du signal. SPIHT utilise l'erreur quadratique moyenne comme métrique (MSE [4]). Ainsi le coefficient de la transformée de plus large amplitude doit être transmis en premier, de même son bit de poids le plus fort avant les bits de poids plus faibles.

[2] Set Partioning in Hierarchical Trees
[3] Embedded Zerotree Wavelet
[4] Mean Square Error

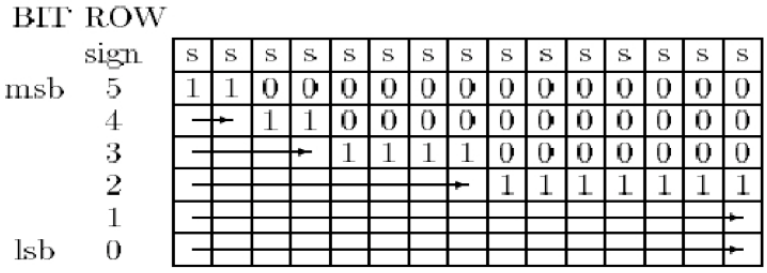



FIG. 2.2 – Représentation binaire des coefficients triés selon leurs amplitudes ([Said et Pearlman, 1996]).

Cependant, la relation entre MSE et qualité d'image perçue n'est pas systématique. Il faut noter que les coefficients de la transformée peuvent avoir une valeur importante sans que ces derniers n'apportent de détails importants au niveau de la perception visuelle de l'image. C'est le cas par exemple lorsqu'un coefficient important se situe dans une zone fortement bruitée ; il n'apporte pas beaucoup d'information pour améliorer la qualité de l'image.

### 2.2.2 Transmission des indices ondelettes quantifiés

SPIHT propose de trier les coefficients suivant leurs amplitudes afin de transmettre l'information de façon progressive. Les bits de poids forts seront transmis en premier puis par étapes de raffinement successives, les bits de poids plus faibles seront ensuite transmis. La figure 2.2 montre ces coefficients (codés sur 7 bits, un bit de signe et 6 bits pour la valeur) en colonnes triés. Le bit de poids le plus fort (MSB) est ici noté 5, et le bit de poids le plus faible (LSB) par 0.

Dans cet exemple, les deux indices de plus fortes amplitudes, correspondant aux deux premières colonnes, sont transmis en priorité dans le flux. Leurs signes sont d'abord codés, puis intervient une phase de raffinement qui consiste à transmettre le bit le plus significatif de l'indice pas encore codé.

Le problème lié à l'ordonnancement des indices selon leurs amplitudes est qu'il faut transmettre de l'information supplémentaire pour localiser spatiallement chaque indice transmis dans un espace à deux dimensions, ce qui, pour $i$ indices, représente $2i$ entiers. Pour pallier à ce problème, l'algorithme de tri des partitions d'ensembles ne transmet pas explicitement l'ordre des données. Le principe est basé sur le fait que le décodeur est capable de reproduire le chemin d'exécution de l'encodeur puisque cet ordre est défini par des règles communes. Avec un tel algorithme, il n'est pas nécessaire de trier complètement les coefficients mais seulement de détecter ceux compris entre $|2^n|$ et $|2^{n+1}|$, $n$ étant le rang du bit transmis.

### 2.2.3 Arbres spatiaux orientés

L'information la plus importante d'une image est contenue dans les basses fréquences. En conséquence, la variance décroît plus on se trouve dans les bas niveaux de la pyramide des sous-bandes. De plus il a été observé qu'il existe une dépendance entre les niveaux des sous-bandes, et plus précisément qu'un indice a généralement

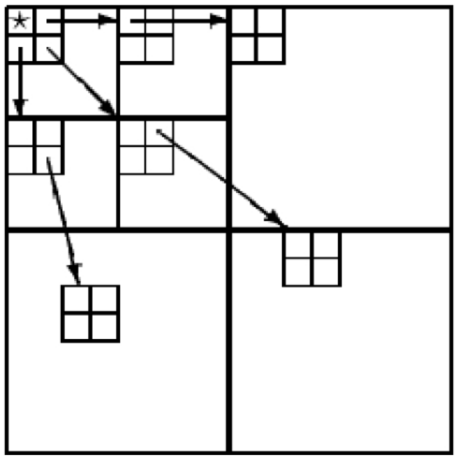


FIG. 2.3 – Exemple de dépendances spatiales dans l'arbre

une plus petite amplitude que l'indice du niveau supérieur à la même localisation spatiale dans l'image. L'utilisation d'un arbre spatial permet de hiérarchiser les données, les indices des plus bas niveaux de résolutions seront transmis avant les autres. La figure 2.3 illuste un arbre utilisé dans le cadre de l'algorithme SPIHT.

La représentation de l'arbre spatial est la suivante : un noeud est soit une feuille soit pointe vers quatre fils, un carré de $2 \times 2$ pixels adjacents. Les pixels du plus haut niveau de la hiérarchie sont les racines. Un fils d'un noeud correspond aux pixels à la même position spatiale mais avec un niveau plus fin dans la pyramide.

Cet algorithme repose sur la gestion de trois listes : les coefficients significatifs (LSP), les coefficients insignifiants (LIP) et les ensembles insignifiants (LIS). La liste des coefficients significatifs est initialement vide, tandis que la liste de coefficients insignifiants contient les racines de chaque arbre (coefficients de la bande basse fréquence) et la liste des ensembles insignifiants contient l'ensemble des descendants de chaque arbre.

Cette partition initiale est segmentée récursivement au moyen de deux règles. Si un ensemble de descendants d'un noeud est significatif, il est séparé en quatre coefficients fils directs de ce noeud, et l'ensemble des autres descendants. Les fils directs sont ajoutés à la LIP ou à la LSP en fonction de leur signifiance. Si au moins un élément de l'ensemble des autres descendants est significatif, cet ensemble est séparé en quatre ensembles insignifiants ajoutés à la LIS. Le fait de traiter les coefficients par groupes de quatre permet d'effectuer un codage entropique efficace par la suite. Comme dans EZW, la passe de raffinement consiste à coder progressivement les bits de poids plus faibles des coefficients significatifs.

L'article [Said et Pearlman, 1996] expose en détail l'algorithme utilisé.

## 2.3 Transformée orientée lifting EDOWT

La méthode de *lifting* classique appliquée à une image 2-D consiste à effectuer deux *liftings* successifs selon les lignes et les colonnes pour extraire les coefficients ondelettes. De ce fait, les coefficients ondelettes qui représentent les contours qui

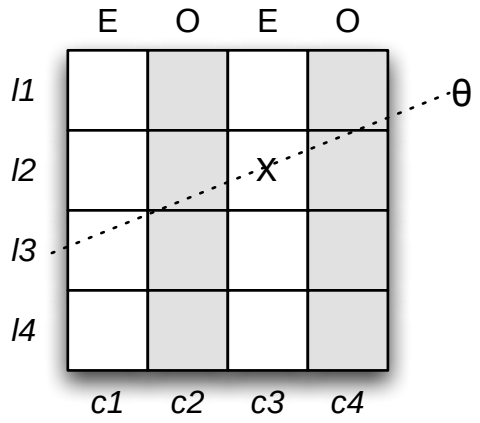



FIG. 2.4 – Exemple de lifting orienté basé-colonne selon un angle $\theta$

ne sont pas horizontaux ou verticaux se retrouveront dans plusieurs sous-bandes différentes à plusieurs niveaux de résolutions différents.

– **Le lifting orienté**

Avec l'approche du *lifting* orienté, l'idée est de réaliser une décomposition en ondelettes suivant un angle $\theta$, selon le contenu local de l'image. Pour cela, un premier lifting est effectué selon un angle $\theta$, avec $\theta \in [0, \pi[$, puis un second selon l'angle orthogonal $\theta \pm \frac{\pi}{2}$. L'opération a pour but de concentrer l'énergie des coefficients dans une sous-bande particulière. La prédiction et la mise à jour sont calculées grâce à une interpolation linéaire entre les voisins d'un coefficient selon la direction de filtrage.

Par exemple, l'opération de prédiction d'un *lifting* basé sur une colonne va consister à estimer un coefficient grâce à une interpolation linéaire de ses voisins placés respectivement sur la colonne adjacente à gauche et à droite. La figure 2.4 illustre la phase de prédiction d'un lifting orienté basé-colonne selon l'angle $\theta$. Le coefficient $X$ est prédit à partir de l'interpolation linéaire des coefficients aux coordonnées $(l2, c2)$ et $(l3, c2)$, et de l'interpolation linéaire des coefficients $(l1, c4)$ et $(l2, c4)$.

Le choix des directions de filtrage est basé sur le contenu de l'image, le but étant d'effectuer un filtrage anisotropique des contours rectilignes et des textures orientées. Les deux *liftings* basé-ligne et basé-colonne ne sont pas commutatifs. Le choix a donc été fait d'appliquer le lifting d'abord selon la régularité pour que la sous-bande haute fréquence ne contienne pas de grands coefficients et que le bruit soit minimal dans la sous-bande basse fréquence.

Il a été démontré [Jeannic *et al.*, 2007] que cette transformation permet une reconstruction parfaite du signal.

– **Détection des orientations de l'image et codage**

Grâce au *lifting* orienté, il est maintenant possible d'adapter la transformation en ondelette en fonction des orientations des contours et des textures de l'image. Cependant, une image « naturelle » ne possède pas une orientation principale unique, mais une multitude d'orientations réparties spatialement. L'image peut donc être décomposée en *blocs* présentant une orientation principale. Les blocs et leurs orientations associées sont transmis en priorité dans le flux binaire pour permettre la transformation inverse des coefficients.

## 2.4 Étude sur la perception visuelle et ses applications dans le domaine de la compression d'image

Les stratégies de compression modernes exploîtent le fait que le système visuel humain est un capteur « imparfait ». Dans la plupart des cas, une représentation exacte bit-par-bit de l'image n'est pas nécessaire, les données peuvent donc être codées de manière non-inversible. Les compressions d'images avec pertes peuvent être classifiées de la façon suivante : (1) *visuellement sans-perte* où l'image reconstruite est visuellement indifférenciable de l'originale ; ou (2) *visuellement dégradée*, qui produit des images contenant des distorsions visibles. Pour maximiser l'efficacité de la compression, ces deux catégories doivent exploîter les propriétés du système visuel humain (SVH).

Nous allons nous intéresser aux nouveaux procédés, dont l'objectif est de « calquer » au mieux le SVH afin d'augmenter la compression sans perdre la qualité visuelle de l'image.

Plusieurs types d'outils permettent une optimisation visuelle de la compression, et sont basés sur le modèle du système visuel humain. La sensibilité visuelle varie en fonction des caractéristiques propres de l'image, telles que la localisation spatiale des fréquences, le contraste local de l'image ou la couleur, mais également en fonction des conditions de visualisation qui sont entre autres la luminosité, la distance et l'angle de visualisation.

Concrètement, deux propriétés du système visuel humain sont très largement exploités par les nouveaux systèmes de compression d'images fixes :

- Le capteur visuel humain est sensible aux contrastes en fonction de la fréquence. Désormais, nous savons qu'un signal très basse fréquence ou très haute fréquence n'est pas perçu par le système visuel humain si le contraste est trop faible.
- Le capteur visuel humain est sujet aux effets de masquage, ce qui se produit lorsqu'un signal fort dissimule ou "cache" un autre signal.

Nous allons nous focaliser sur cette dernière propriété, puisqu'elle inspire le schéma de compression que nous proposons. Pour cela, les modifications du schéma classique de compression peuvent concerner l'étape de quantification comme l'étape de codage. Ce choix n'est pas anodin, puisqu'il implique des conséquences différentes :

- La modification de l'étape de quantification a pour objectif de supprimer l'information inutile, c'est-à-dire l'information qui n'est pas perçue par le système visuel humain. La conséquence directe est que le gain en terme de compression n'est pas négligeable. Cependant, les algorithmes d'optimisation n'étant pas idéalement efficaces, la suppression d'information peut occasionner une perte de qualité visuelle.
- La modification de l'étape de codage va coder les mêmes données, mais dans un ordre différent. Ainsi, il n'y a pas de gain de compression observé, mais l'information la plus pertinente visuellement sera codée en premier. De ce fait, le décodage partiel de la même quantité d'information restituera une image de meilleur qualité.

Cependant, l'utilisation d'un codeur de type SPIHT ne permet pas d'utiliser ce dernier procédé. Nous allons donc nous focaliser sur les techniques d'optimisation par modification des lois de quantification. Dans les sections suivantes, nous répertorions les différents procédés basés sur l'étude du système visuel humain, appliqués à la compression d'images fixes.

### 2.4.1 Masquage visuel

Le terme de *masquage visuel* est employé pour désigner le phénomène perceptuel lorsque un signal est localement « masqué » (caché) par un autre signal « masquant » (typiquement une texture de fond)[Zeng *et al.*, 2002]. En effet, l'image agit comme un fond qui réduit la visibilité des signaux « gênants » générés par la distorsion (le bruit).

#### Masquage engendré par les signaux eux-mêmes

Il a été déterminé expérimentalement [Zeng *et al.*, 2002] que le coefficient de masquage est directement proportionnel à la valeur du coefficient d'ondelette. L'idée est donc d'appliquer une quantification non-uniforme en fonction de l'activité et de la localisation spatiale des fréquences. Pour cela, on utilise une fonction de non-linéarité décroissante appelée fonction de transfert, puis on applique une quantification uniforme. La fonction de transfert est par défaut une fonction puissance, à savoir :

$$y = x^{\alpha}, 0 < \alpha \leq 1. \tag{2.1}$$

Les meilleurs résultats sont obtenus avec $\alpha = 0{,}7$ [Zeng *et al.*, 2002].
Grâce à ce procédé, on obtient une quantification non-uniforme illustrée à la figure 2.5, avec un pas de quantification préalablement défini à 1 et $\alpha = 0,5$.

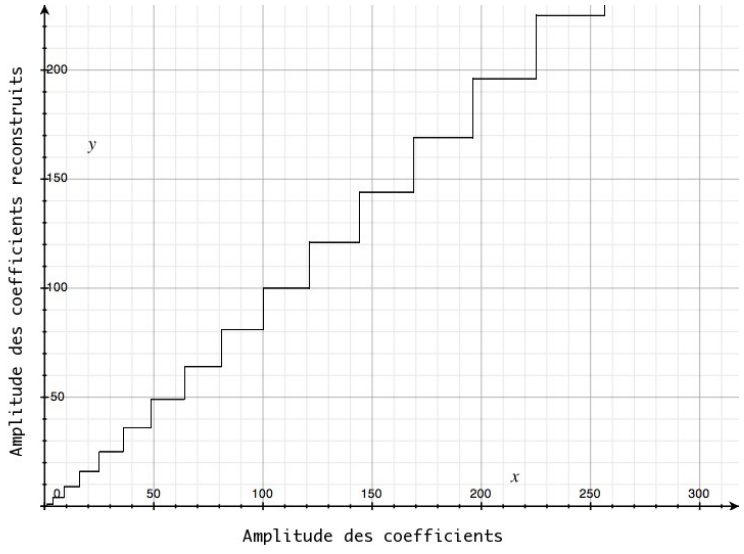



FIG. 2.5 – Quantification non-uniforme

Cependant, le *self-masking* soulève des problèmes intéressant liés à la nature différente du système visuel humain et de la compression basée sur la décomposition ondelette :

**a.** La sous-bande diagonale contient les détails orientés à 45ř et -45ř. Or, pour le système visuel humain, les détails orientés à 45ř ne masquent pas ceux qui sont orientés à -45ř et inversement, ce qui n'est pas le cas en compression image.

**b.** L'empiétement des sous-bandes horizontales et verticales sur la sous-bande diagonale : les contours diagonaux peuvent se retrouver dans les sous-bandes horizontales et verticales comme illustré sur la figure 2.6. Cette figure représente une décomposition ondelette classique, et sa représentation dans l'espace fréquentiel correspondante. Par exemple, la sous-bande diagonale *1HH* de la décomposition ondelette est représentée par les quatre blocs *1HH* dans l'espace fréquentiel. Ce constat soulève deux problèmes majeurs :

- Lorsque l'amplitude des coefficients diagonaux présents dans les sous-bandes verticales et horizontales est assez grande, l'effet de masquage va produire un pas de quantification important, ce qui peut causer des distorsions horizontales et verticales sur les contours diagonaux.
- L'énergie déplacée des sous-bandes diagonales vers les sous-bandes horizontales et verticales ne sera pas prise en compte par l'effet de masquage des structures diagonales. Il en résulte que le débit sera plus conséquent pour les sous-bandes diagonales et donc que la compression sera moins efficace.

**c.** Sensibilité de la phase : différence entre le SVH et le *visual masking* implémenté dans JPEG2000. Dans JPEG2000, le *self-masking* est restreint aux coefficients qui ont une largeur de phase proche des degrés nuls, donc le masquage résultant de l'énergie proche du passage par zéro n'est pas pris en compte.

Cette étape de masquage visuel s'intègre dans le schéma de compression de JPEG 2000 comme illustré sur la figure 2.7.

### *Point-wise extended masking*

Ce procédé prend en compte l'effet de masquage et d'addition spatiale des coefficients voisins ([Zeng *et al.*, 2000]). Chaque coefficient va donc être quantifié en fonction de sa propre valeur, mais aussi en fonction du voisinage causal. En effet, plus l'activité locale de l'image est importante, plus nous pouvons constater d'effets de masquage. La mise en œuvre consiste à combiner les deux processus séparément :

1. On applique tout d'abord la fonction puissance sur les coefficients (technique issue du *self-masking*).

$$x_i \rightarrow y_i = sign(x_i)|x_i|^{\alpha} \tag{2.2}$$

2. Puis on normalise par un facteur de masquage spatial en fonction de l'amplitude des signaux voisins.

$$z_i = y_i/(1 + a \sum_{\text{k voisin de i}} |\hat{x_k}|^{\beta}/|\phi_i|) \tag{2.3}$$

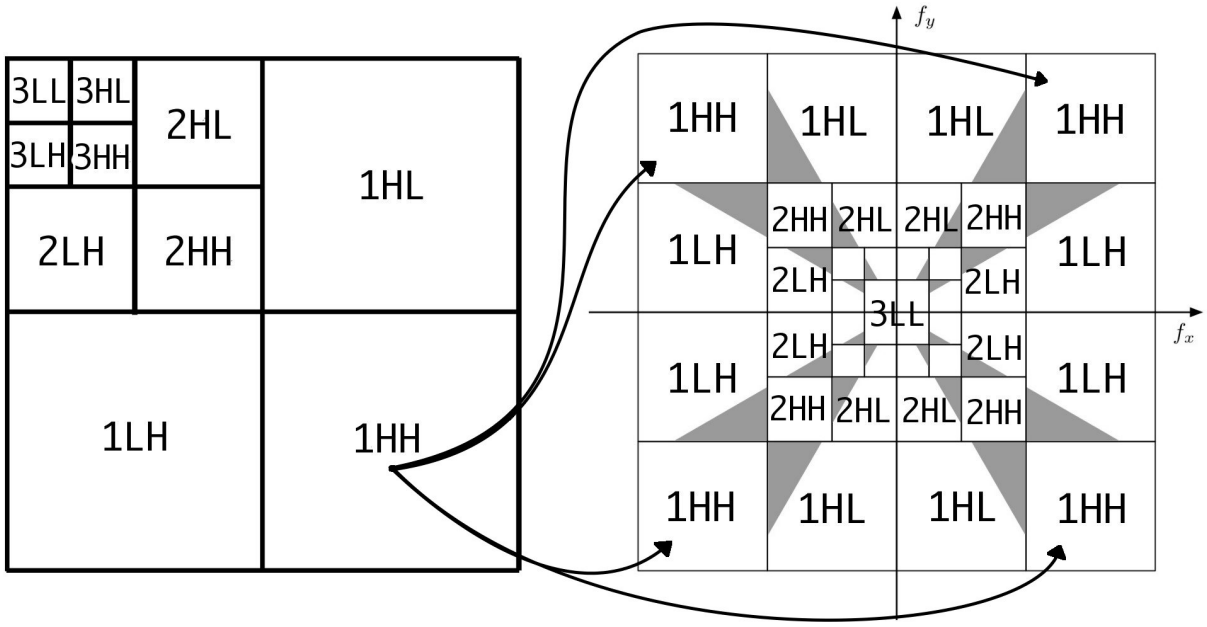



FIG. 2.6 – À gauche : décomposition ondelette classique en sous-bandes représentées dans un espace spatio-fréquentiel. À droite : empiétement horizontal et vertical sur les sous-bandes diagonales dans un espace fréquentiel, avec correspondance des sous-bandes de la figure de gauche.

où $|\phi_i|$ est la taille du voisinage causal, $a$ est un facteur de normalisation avec une valeur constante de $(10000/2^{bitdepth-1})^{\beta}$ et *bitdepht* est le nombre de bits par pixel de l'image d'origine, $\hat{x_k}$ est la valeur quantifiée des coefficients voisins. Le voisinage contient les coefficients déjà traités (précédents) de la même sous-bande dans une fenêtre de taille $N \times N$ centrée sur le coefficient courant, comme illustré sur la figure 2.8.

Le paramètre $\alpha$ détermine le degré de *self-masking* (typiquement à 0,7). Le paramètre $\beta$ et $N$ détermine le degré de masquage visuel. Ensuite, une quantification uniforme sur $z_i$ est appliquée.

Les avantages de cette méthode sont multiples. Tout d'abord, elle distingue les coefficients de grandes amplitudes qui se situent dans une zone texturée ou dans une structure de contours, qui n'ont pas la même importance du point de vue de la qualité visuelle de l'image reconstruite. De plus, elle résout le problème du *self-masking* que nous avons vu concernant les contours diagonaux.

Cependant, cette technique est limitée, puisque pour prendre en compte le voisinage, les coefficients doivent obligatoirement appartenir au même bloc. De plus, de l'information supplémentaire doit être codée pour avertir le décodeur de la façon dont est pris en compte le voisinage. Enfin, un problème dû à la quantification et qui pose problème pour le calcul du facteur du masquage visuel exprimé par l'équation 2.3 est que les coefficients du voisinage causal n'auront pas les même valeurs au décodage qu'au codage. Une méthode a été développée pour contrer ce phénomène. Elle consiste à ne retenir que les $n$ bits les plus signifiants de chaque coefficient voisin

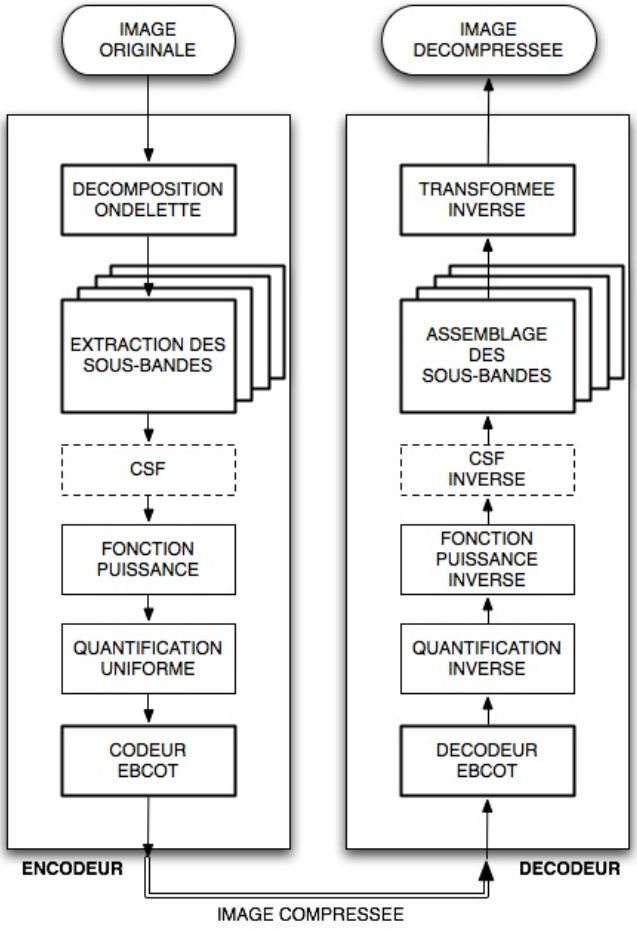



FIG. 2.7 – Schéma de compression du JPEG 2000 intégrant le procédé du *self-masking* (*via* la fonction puissance)

pour calculer le facteur de masquage visuel pour ainsi disposer des mêmes valeurs au codage et au décodage (la quantification n'affecte que les bits de poids les plus faibles). Cependant, ce procédé affecte légèrement les résultats.

### Masquage selon la texture locale

Matthew Gaubatz et son équipe ont expérimenté un procédé [Gaubatz *et al.*, 2006] qui analyse l'activité locale d'une image pour estimer l'effet de masquage et ainsi déterminer un pas de quantification adapté. Leur constat fut identique à celui de l'équipe de W. Zeng 2.4.1, mais la mise en œuvre est différente. L'idée proposée requiert une carte des contrastes codée qui sera transmise au décodeur. Grâce à cette carte des contrastes, la pas de quantification est calculé de la façon suivante :

$$c'_k = \frac{1}{Q_k^{seuil}} \cdot c_k,$$

où $c'_k$ est la valeur du coefficient quantifié de $c_k$, et $Q_k^{seuil}$ est le pas de quantification au seuil de visibilité basé sur le contraste.

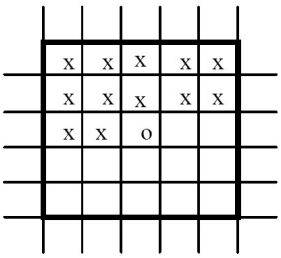



FIG. 2.8 – Exemple d'un voisinage causal ($N = 5, |\phi_i| = 12$). o : coefficient courant, x : coefficient appartenant au voisinage causal

La carte des contrastes est obtenue en calculant le contraste pour chaque valeur de pixel de l'image. Pour une image de taille $N \times N$, la carte des contrastes aura donc une taille $N \times N$. Ensuite, deux approches sont utilisées pour réduire la quantité d'information nécessaire à transmettre.

1. Réduction explicite du surplus d'information

   La carte des contrastes est réduite en applicant un filtre moyenneur et en sous-échantillonant le résultat, afin de diminuer la quantité d'information à transmettre. Un compromis doit être déterminé entre débit et qualité visuelle. La carte des contrastes subit une transformation en ondelettes, une quantification "zone-morte" et un codage arithmétique afin de diminuer sa taille dans le flux binaire généré. Au codage et au décodage, il est nécessaire d'adapter la carte des contrastes à la taille de chaque sous-bande en effectuant une interpolation ou un sous-échantillonnage selon la taille de la sous-bande, comme illustré à la figure 2.9.

2. Réduction implicite du surplus d'information

   Il existe une corrélation entre l'indice maximum d'une sous-bande quantifiée et le pas de quantification de cette sous-bande. Cependant, cette corrélation n'implique pas forcément qu'il existe une corrélation entre la valeur du coefficient ondelette quantifié actuel et la carte des quantifieurs (issue de la carte des contrastes). Pourtant, pour des images naturelles respectant un modèle « raisonnable » de distribution d'amplitudes des coefficients, cette corrélation existe bien. En effet, les zones très texturées, correspondant à des amplitudes de coefficients grands, seront quantifiée grossièrement avec pas large pour compenser l'effet de masquage, alors que les zones « plates » (amplitude de coefficient petite) seront finement quantifiées. Plus le pas de quantification pour le coefficient courant est large, plus la valeur quantifiée a de probabilités d'être grande. Le lien entre la carte des quantifieurs et les coefficients quantifiés se révèle encore plus important entre la carte des quantifieurs et la carte des bits significatifs des coefficients. L'idée est d'estimer la valeur des bits significatifs des coefficients quantifiés en fonction de la carte des quantifieurs (auparavant transmise au décodeur).

   L'étape de quantification (et notamment la « zone-morte ») provoque l'augmentation de coefficients à valeur nulle. De ce fait, la localisation des coefficients nuls permet d'éviter une estimation trompeuse de ces coefficients. C'est

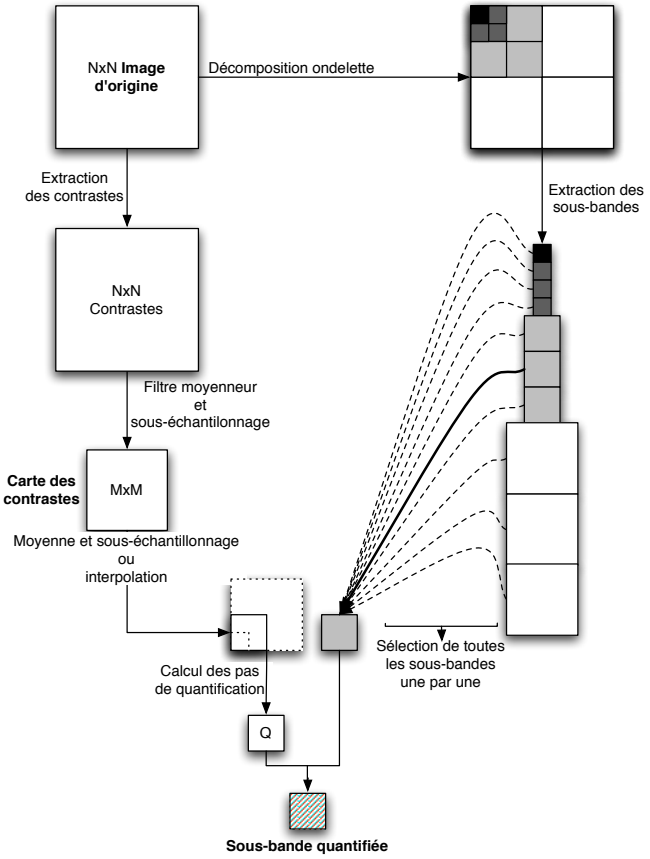



FIG. 2.9 – Schéma de quantification des coefficients ondelettes selon la carte des contrastes

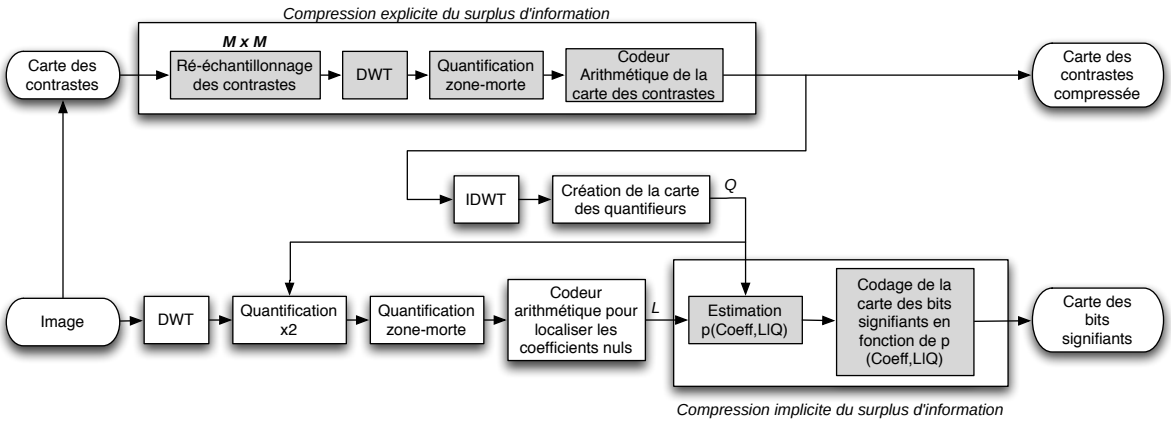



FIG. 2.10 – Diagramme du codeur d'une image fixe utilisant le contraste local pour déterminer le pas de quantification

pourquoi, dans le schéma présenté par Matthew D. Gaubatz et son équipe, la localisation des coefficients nuls est d'abord codée, suivi des bits significatifs des coefficients non nuls. Cette astuce permet également de mieux estimer les coefficients proches des coefficients nuls, puisque la valeur de coefficients non quantifiés voisins est sensiblement identique dans une grande partie des cas, donc l'indice proche d'un indice nul peut être estimé à la limite de la « zone morte ».

Le diagramme 2.9 schématise le codeur décrit.

### 2.4.2 Masquage structurel

Le masquage structurel [Gaubatz *et al.*, 2006] détermine le pas de quantification local en fonction du contraste et du type de l'image. En particulier, trois types ont été imaginés : les textures, les structures et les contours. Un classifieur est utilisé pour déterminer à laquelle de ces trois catégories une région de l'image appartient, et un pas de quantification est calculé à partir de ces informations. Les cartes des contrastes et des types sont envoyés au codeur qui les transmet dans le flux binaire de façon explicite.

La stratégie de codage procède en deux étapes : (1) la segmentation de l'image et la classification des régions, et (2) le codage de l'information supplémentaire et des coefficients ondelettes quantifiés. La classification s'effectue de la façon suivante :

1. Chaque bloc est étiqueté comme étant une texture si la valeur moyenne au carré des coefficients du bloc (l'énergie) dépasse un seuil donné par une constante $C_1$ calculée en fonction l'activité globale de l'image.
2. Tous les blocs non étiquetés sont analysés par un détecteur de contour Canny [Canny, 1986]. Chaque bloc est étiqueté « contour » si le résultat est au dessus d'un seuil calculé en fonction de l'activité globale de l'image.

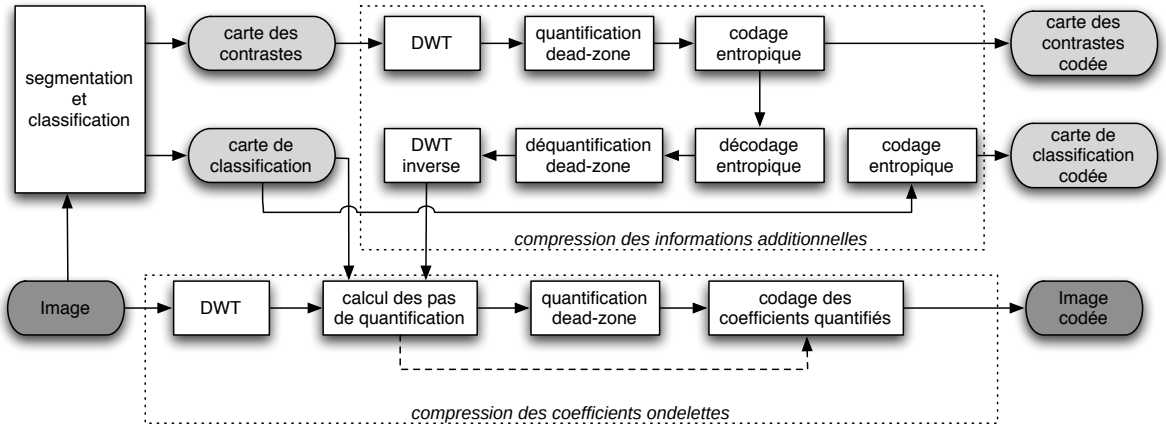



FIG. 2.11 – Diagramme du codeur pour le masquage structurel

3. Les blocs non étiquetés sont analysés par un autre détecteur de contour, l'opérateur « Laplacian of Gaussian ». De la même façon, les blocs dont le résultat surpasse un certain seuil sont étiquetés « contour ».
4. Les blocs restants dont l'énergie moyenne dépasse un certain seuil sont étiquetés comme étant des contours.
5. Tous les blocs non étiquetés sont classés « structure ».

Le schéma de la figure 2.11 illustre le codeur pour le masquage dit « structurel ». La carte des contrastes, qui contient un grand ensemble de valeurs, est codée selon un schéma classique de compression, c'est-à-dire une transformation en ondelette classique, une quantification et un codage entropique. La carte des classifications est codée avec un codeur entropique.

La carte des contrastes et la carte de classification doivent être transmises en priorité dans le flux binaire pour permettre la reconstruction. Les blocs de coefficients quantifiés de l'image peuvent être codés au choix de façon imbriquée (codeur du type EBCOT) ou non (codeur du type SPIHT). D'après les résultats de l'application de ce procédé, les images possèdent une meilleure qualité visuelle que par la méthode classique pour des taux de compression similaires, même si l'information supplémentaire ajoutée représente une part importante du flux binaire codé, ce qui rend cette méthode inappropriée pour des compressions à très bas débit.

# Chapitre 3

# Proposition

## Sommaire



## 3.1 Introduction

Dans ce chapitre, nous allons exposer plusieurs procédés qui vont nous permettre d'améliorer les performances de compression en terme de débit. Ces procédés reposent sur l'étude bibliographique préalable et sur les aspects particuliers que présentent la décomposition en ondelette *via* le *lifting* orienté. En effet, cette dernière offre des possibilités intéressantes que nous nous devons d'exploiter tant au niveau de la quantification que du codage. En particulier, la répartition présumée de l'énergie à travers les sous-bandes d'un même niveau de décomposition permet d'affiner la représentation hiérarchique de l'information.

Nous allons maintenant étudier trois aspects différents. Dans un premier temps, nous allons présenter une méthode de segmentation et de classification des images.

Cette classification et les connaissances *a priori* que nous disposons vont nous permettre d'exposer une nouvelle approche pour la quantification et le codage de l'information.

## 3.2 Segmentation et classification

La classification que nous proposons va directement influencer la stratégie de quantification et de codage. Pour commencer, l'idée est d'adapter le pas de quantification en fonction du contenu local de l'image. Par exemple, pour une zone de l'image où il y a peu d'activité, les informations contenues dans les sous-bandes hautes-fréquences n'apporteront que très peu voire aucune amélioration visuelle. Pour ses raisons, il peut être intéressant de quantifier ces données "grossièrement", ou de ne pas les coder du tout dans le flux de sortie. Les stratégies de quantification et de codage sont expliquées en détail dans les sections suivantes. Nous allons nous intéresser ici à la méthode retenue pour segmenter et étiqueter chaque zone distincte de l'image, ainsi qu'au codage de l'information supplémentaire nécessaire à l'étape de décompression.

### 3.2.1 Principe

La segmentation et la classification est la première étape de schéma de compression que nous proposons. Pour la suite, deux types d'information sont précieux, à savoir :

- la direction privilégiée de la zone de l'image, qui renseignera sur l'orientation du lifting orienté de la transformée en ondelettes ;
- l'activité locale de l'image, pour adapter au mieux la quantification et la codage de l'information.

Pour ce qui concerne l'activité locale de l'image, nous avons sélectionné quatre étiquettes, correspondant à quatre classes différentes :

- classe 1 : zone uniforme où l'activité est réduite ;
- classe 2 : zone mono-orientée présentant une unique direction privilégiée ;
- classe 3 : zone multi-orientée présentant plus d'une direction priviliégiée ;
- classe 4 : zone texturée où l'activité est importante et ne présentant pas de direction privilégiée.

Pour les zones uniformes et texturées qui ne présentent pas de directions privilégiées, nous avons choisi une orientation $\theta = 0°$ pour le lifting orienté, qui a pour avantage de ne pas introduire d'erreur d'interpolation. Pour les zones mono-orientées et multi-orientées, la direction privilégiée est sélectionnée parmis $N$ classes d'angle, réparties uniformément entre 0° et 180°. Dans nos expérimentations, nous avons déterminé que le bon compromis entre résultat visuel et codage de l'information est obtenu avec $N = 8$.

### 3.2.2 Méthode et algorithme

Tout d'abord, l'image est découpée en blocs de tailles identiques. Nous utilisons par défaut des blocs de tailles $8 \times 8$ pixels pour une résolution de $512 \times 512$ pixels

et $256 \times 256$ pixels, et des blocs de tailles $4 \times 4$ pixels pour les résolutions d'image inférieures. Dans un premier temps, nous allons décrire la méthode utilisée pour étiqueter chaque bloc $\mathcal{B}$, ensuite nous verrons la représentation que nous avons choisi pour manipuler ces données.

### Méthode de classification

Chaque bloc $\mathcal{B}$ est étiqueté selon le principe décrit précédemment, en utilisant une métrique qui doit être simple, peu coûteuse et cohérente en terme de résultat et par rapport à la définition de nos classes. Notre métrique s'appuie sur la mesure du gradient grâce aux filtres de Sobel. Pour chaque pixel du bloc, le gradient est défini par :

$$\nabla I_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * I \quad \text{et} \quad \nabla I_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * I \ , \tag{3.1}$$

où $I$ représente l'image d'origine. $G_x$ et $G_y$ sont deux images qui contiennent en chaque point des approximations respectivement de la dérivée horizontale et verticale. Ensuite, nous pouvons combiner les gradients horizontaux et verticaux pour approximer la norme et la direction du gradient :

$$\|\nabla I\| = \sqrt{\nabla I_x^2 + \nabla I_y^2}. \tag{3.2}$$

$$\Theta = \arctan\left(\frac{\nabla I_y}{\nabla I_x}\right). \tag{3.3}$$

Ces deux valeurs permettent d'évaleur l'intensité du contour et son orientation.

La probabilité de la mesure est modélisée selon [3.5]. Si le gradient $\nabla I$ n'est pas le vecteur nul, la probabilité que la direction othogonale de $\nabla I(x, y)$ soit $\theta$ est modélisée par la loi normale de moyenne $\theta_I + \frac{\pi}{2}$, d'écart-type $\frac{\sigma}{\|\nabla I\|}$ :

$$P(x, y, \cdot) \sim \mathcal{N}\left(\theta_I + \frac{\pi}{2}, \frac{\sigma}{\|\nabla I\|}\right), \tag{3.4}$$

où $\sigma$ est l'écart-type de l'erreur sur le gradient, $\theta_I$ est l'orientation du gradient mesurée par arctan. Avec $K$ une constante de normalisation, nous avons donc :

$$P(x, y, \theta) = \frac{1}{K} e^{-dist^2(\theta, \theta_I(x,y)+\frac{\pi}{2}) \cdot \frac{\|\nabla I\|^2}{\sigma^2}}. \tag{3.5}$$

Si $\nabla I = \vec{0}$, $P(x, y, \cdot) \sim \mathcal{U}$, c'est à dire :

$$P(x, y, \theta) = \frac{1}{K} \cdot \frac{1}{2\pi}. \tag{3.6}$$

La mesure $M$ utilisée pour la classification est définie par la mixture des lois de probabilité $P(x, y, \theta)$ sur un bloc $\mathcal{B}$.

$$M(\theta) = \frac{1}{card(\mathcal{B})} \sum_{(x,y)\in\mathcal{B}} P(x, y, \theta). \tag{3.7}$$

Un premier test consiste à mesurer la distribution de $M$ par rapport à une distribution uniforme :

$$d = \sum_{\theta \in \Theta} \left( M(\theta) - \frac{1}{N} \right)^2 , \tag{3.8}$$

où $N$ est le nombre de classes d'orientations étudiées. Si l'écart $d$ est inférieur à un certain seuil, alors nous pouvons en conclure qu'aucune orientation privilégiée dans le bloc ne se dégage, et donc que le bloc $\mathcal{B}$ appartient à la classe 1 ou 4 (zone uniforme ou texturée). Dans ce cas, un seuillage par rapport à $\tau$ sur la moyenne de la norme du gradient permet de distinguer si $\mathcal{B}$ appartient à la classe 1 ou 4. Dans le cas où au moins une orientation privilégiée se dégage, le nombre de maxima locaux où $M(\theta)$ est supérieur à $\frac{1}{N}$ donne le nombre d'orientations priviliégiées détectées.

### Représentation des données issues de la classification

La structure utilisée doit fournir deux indications pour chaque zones de l'image, à savoir la direction privilégiée si elle existe et le type de la zone (uniforme, mono-orientée, multi-orientée ou texturée). Pour cela, nous utilisons deux quadtrees différents, notés $\mathcal{Q}_T$ pour représenter les types de zone et $\mathcal{Q}_O$ pour les orientations privilégiées. Un quatree est un arbre quaternaire composé de noeuds et de feuilles, correspondant à une zone carré de l'image. Un noeud comporte quatre fils contenus spatiallement dans le noeud. Chaque feuille contient une valeur, qui indique selon le quadtree soit la classe d'orientations, soit le type. Un exemple de quadtree est illustré sur la figure 3.1.

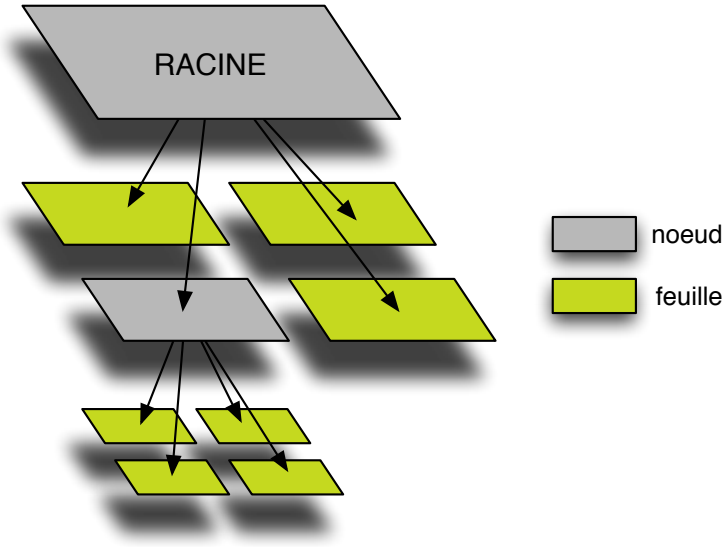



FIG. 3.1 – Exemple de quadtree

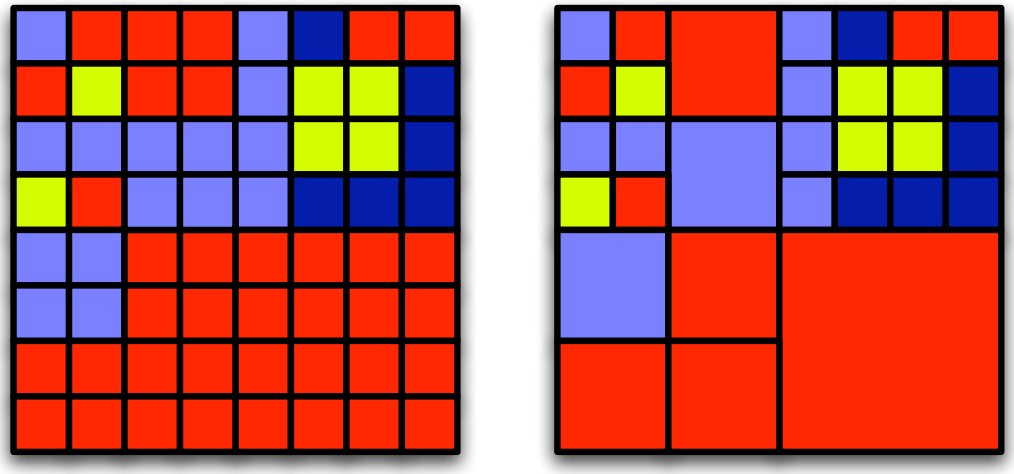
Pour construire les deux quatrees $\mathcal{Q}_T$ et $\mathcal{Q}_O$, nous commençons par les initialiser en faisant correspondre la taille de chaque feuille avec la taille des blocs $\mathcal{B}$. Ensuite, lorsque les quatre fils d'un même noeud ont la même valeur, l'arbre quaternaire est simplifié en remplaçant le noeud par une feuille de valeur égale aux fils. La figure

3.2 illustre ce traitement. L'algorithme pour réduire d'un niveau les feuilles d'un quadtree est exposé en 1, il faut l'appliquer plusieurs fois itérativement pour obtenir le quadtree simplifié.

**Algorithme 1** : Algorithme de simplification d'un quadtree

**Entrées** : cellule *root*

**si** *root est un noeud* **alors**
  **si** $root \rightarrow filsHautGauche$ *est une feuille* $\wedge root \rightarrow filsHautDroit$ *est une feuille* $\wedge root \rightarrow filsBasGauche$ *est une feuille* $\wedge root \rightarrow filsBasDroit$ *est une feuille* **alors**
    $value \leftarrow \{root \rightarrow filsHautGauche \rightarrow valeur\}$;
    **si** $\{root \rightarrow filsHautDroit \rightarrow valeur\} = value \wedge \{root \rightarrow filsBasGauche \rightarrow valeur\} = value \wedge \{root \rightarrow filsBasDroit \rightarrow valeur = value\}$ **alors**
      simplifie($root$);
      $\{root \rightarrow valeur\} \leftarrow value$;
    **fin**
  **sinon**
    appel récursif $root \rightarrow filsHautGauche$;
    appel récursif $root \rightarrow filsHautDroit$;
    appel récursif $root \rightarrow filsBasGauche$;
    appel récursif $root \rightarrow filsBasDroit$;
  **fin**
**fin**



(a) Quadtree d'origine (b) Quadtree modifié

FIG. 3.2 – Optimisation d'un quadtree constitué de quatre types de feuilles

En général, les images naturelles sont composées de zones similaires relativement étendues, par exemple un ciel uni, une forêt, une chemise rayée, etc. De ce fait, les

blocs $\mathcal{B}$ voisins ont une probabilité plus accrue d'être étiqueté de façon identique. La structure quadtree est adapté à ce style de disposition, réduisant ainsi la hauteur moyenne de l'arbre, le nombre de noeuds intermédiaires et le nombre de feuilles. Le codage du quadtree, décrit dans la prochaine section, s'en retrouve simplifié.

### 3.2.3 Encodage

Dans la section précédente, nous avons vu que l'information additionnelle nécessaire au codage et au décodage de l'image était caractérisée par deux quadtrees $\mathcal{Q}_T$ et $\mathcal{Q}_O$, dont le but est d'étiqueter chaque zone de l'image selon sa direction privilégiée et son type d'activité. L'opération de codage/décodage des quadtrees a pour contrainte une reconstruction rigoureusement identique du quadtree, avec comme objectif, dans un contexte de compression de données, une taille du flux binaire la plus petite possible.

Nous allons étudier trois solutions différentes, de complexités et de performances en terme de compression croissantes.

#### Méthode classique

La méthode, dite "classique", d'encodage des quadtrees consiste à séparer le codage de la structure du quadtree du codage des valeurs des feuilles. Pour le codage de la structure, nous utilisons un algorithme récursif qui, partant de la racine de l'arbre (c'est-à-dire le noeud englobant tous les autres), code un symbole "0" pour indiquer un noeud et un symbôle "1" pour indiquer une feuille. L'algorithme est présenté au numéro 2. Cet algorithme est appelé avec en paramètre la cellule racine de l'arbre (une cellule est soit un noeud, soit une feuille).

**Algorithme 2** : Algorithme d'encodage de la structure d'un quadtree

**Entrées** : cellule *root*

```
1 si root est une feuille alors
2 |  coder le symbole 1;
3 sinon
4 |  coder le symbole 0;
5 |  appel récursif root → filsHautGauche;
6 |  appel récursif root → filsHautDroit;
7 |  appel récursif root → filsBasGauche;
8 |  appel récursif root → filsBasDroit;
9 fin
```

Par exemple, l'encodage de la structure du quadtree illustré sur la figure 3.1 donne le train binaire suivant : 011011111. Nous pouvons constater qu'il y a autant de bits que de cellules dans l'arbre. Prenons maintenant l'exemple d'une image de taille $512 \times 512$ pixels, avec des blocs $\mathcal{B}$ de tailles $8 \times 8$ minimum. Si nous nous trouvons dans le pire cas où toutes les feuilles correspondent à un bloc $\mathcal{B}$, il y en a en tout 4096. Au niveau supérieur de l'arbre, il y a quatre fois moins de noeuds que de feuilles, c'est-à-dire 1024, puis 256 au niveau suivant, etc. Au total, il y a

5461 cellules dans l'arbre, soit 5461 bits pour coder la structure, ce qui correspond à environ 0.02 bits/pixels. Nous pouvons en conclure que le codage de la structure représente un débit assez faible.

Pour l'encodage des valeurs des feuilles du quadtree, il suffit ensuite de parcourir l'arbre récursivement en profondeur et de coder les symbole de toutes les feuilles les unes après les autres. Au décodage, le parcours identique de l'arbre permettra d'attribuer correctement chaque symbole décodé à une feuille. La taille en bits d'un symbole dépend du nombre de valeurs possibles pour chaque feuilles. Si nous nous plaçons dans le cas où nous encodons le quadtree $\mathcal{Q}_T$, il y a au total quatre symboles : zone uniforme, mono-orientée, multi-orientée et texturée. Ces quatre symboles peuvent donc être codés sur au minimum deux bits. Nous présentons l'algorithme 3 qui illustre l'encodage des valeurs d'un quadtree.

**Algorithme 3** : Algorithme d'encodage des valeurs des feuilles d'un quadtree

**Entrées** : cellule *root*
**si** *root est une feuille* **alors**
  coder le symbole $root \rightarrow valeur$;
**sinon**
  appel récursif $root \rightarrow filsHautGauche$;
  appel récursif $root \rightarrow filsHautDroit$;
  appel récursif $root \rightarrow filsBasGauche$;
  appel récursif $root \rightarrow filsBasDroit$;
**fin**

Pour reprendre l'exemple précédent, nous avons un arbre constitué de 4096 feuilles. La valeur de chaque feuille étant codée sur deux bits, l'encodage des valeurs de ces feuilles représente 8192 bits, soit un peu plus que 0.03 bits/pixel. Pour $\mathcal{Q}_O$, si nous définissons huit orientations possibles, chaque symbole est codé sur trois bits. L'encodage des valeurs de ce quadtree a donc une taille de 12288 bits, soit environ 0.05 bits/pixel. Au total, si nous cumulons l'encodage des quadtrees $\mathcal{Q}_T$ et $\mathcal{Q}_O$, nous obtenons une taille de 31402 bits, soit 0.12 bits/pixel. Pour rappel, nous nous plaçons dans le cas improbable où chaque feuille correspond à un bloc de taille $8 \times 8$ pixels, ce chiffre est donc la valeur maximale possible. Dans ce cas-ci, les deux arbres ont la même structure, ce n'est pas toujours le cas. Par exemple, si quatre feuilles d'un même noeud sont de type mono-orientées, l'arbre $\mathcal{Q}_T$ est simplifiable. D'un autre côté, la direction privilégiées n'est pas nécessairement identique pour ces quatre feuilles, donc $\mathcal{Q}_O$ n'est pas simplifiable.

L'avantage de cette méthode d'encodage des quadtrees est qu'elle est extrêmement simple et peu coûteuse. Nous allons maintenant voir comment il est possible d'adapter cet algorithme générique pour améliorer les performances.

### Amélioration de la méthode d'encodage des quadtrees

Même si la taille d'un quadtree codé selon la méthode précédemment décrite semble dérisoire par rapport à la taille du flux binaire d'une image encodée, il faut garder à l'esprit que pour chaque niveau de décomposition en ondelettes, il faut

encoder deux quadtrees, soit huit quadtrees au total lorsque nous utilisons quatre niveaux de décomposition. Au final, l'encodage de toutes ces données peut atteindre un débit conséquent.

Dans un contexte où nous recherchons en priorité à réduire le débit, nous allons utiliser les informations que nous avons en amont pour encoder les quadtrees plus efficacement. Nous connaissons *a priori* la hauteur maximale de l'arbre, déterminée par la résolution de l'image et la taille minimale d'un bloc. Par exemple, pour une image de résolution $512 \times 512$ pixels et pour des blocs de taille minimale $8 \times 8$ pixels, la hauteur maximale de l'arbre est 7. Ainsi, lors de l'encodage de la structure, le symbole "1" codé pour une feuille au niveau 7 est superflu. L'algorithme modifié est numéroté 4. L'algorithme est appelé avec en paramètre la cellule racine et la hauteur maximum de l'arbre.

**Algorithme 4** : Algorithme amélioré d'encodage de la structure d'un quadtree

**Entrées** : cellule *root*, $hauteur \in N$

```
si hauteur > 1 alors
    si root est une feuille alors
        coder le symbole 1;
    sinon
        coder le symbole 0;
        appel récursif (root → filsHautGauche, hauteur − 1);
        appel récursif (root → filsHautDroit, hauteur − 1);
        appel récursif (root → filsBasGauche, hauteur − 1);
        appel récursif (root → filsBasDroit, hauteur − 1);
    fin
fin
```

En reprenant l'exemple illustrant la pire cas possible (avec une image de $512 \times 512$ pixels, une taille de bloc minimale de $8 \times 8$ pixels et un quadtree composé uniquement de feuilles au dernier niveau de l'arbre), nous obtenons, pour l'encodage de la structure, une taille du flux binaire de 1365 bits, puisque les 4096 bits nécessaires à l'encodage du dernier niveau de l'arbre sont inutiles, soit 0.005 bits/pixels. La réduction du débit de l'encodage de la structure est de l'ordre 75% par rapport à la méthode classique, elle correspond au gain maximum que nous pouvons observer puisque toutes les feuilles sont à la hauteur maximale de l'arbre.

La seconde astuce pour réduire le débit consiste à prendre en compte la corrélation entre le quadtree $\mathcal{Q}_T$ et $\mathcal{Q}_O$. En effet, nous savons que les feuilles étiquetées "uniforme" ou "texture" ont également une direction choisie arbitrairement à 0°. De ce fait, l'encodage du symbole correspondant à 0° est inutile. L'opération d'encodage des valeurs réuni désormais les deux quadtrees $\mathcal{Q}_T$ et $\mathcal{Q}_O$. Nous présentons

l'algorithme 5 correspondant.

**Algorithme 5** : Algorithme amélioré d'encodage des valeurs des feuilles des quadtrees

**Entrées** : cellule $root_{type}$, cellule $root_{orientation}$

/* $root_{type}$ correspond au noeud racine du quadtree représentant les types de zone,$root_{orientation}$ au noeud racine du quadtree représentant les directions privilégiées */

**si** $root_{type}$ *est une feuille* **alors**
  coder $root_{type} \rightarrow valeur$;
  **si** $root_{type} \rightarrow valeur \neq UNIFORME$ *et* $root_{type} \rightarrow valeur \neq TEXTURE$ **alors**
    coder $root_{orientation} \rightarrow valeur$;
  **fin**
**sinon**
  appel récursif $(root_{type} \rightarrow filsHautGauche, root_{orientation} \rightarrow filsHautGauche)$;
  appel récursif $(root_{type} \rightarrow filsHautDroit, root_{orientation} \rightarrow filsHautDroit)$;
  appel récursif $(root_{type} \rightarrow filsBasGauche, root_{orientation} \rightarrow filsBasGauche)$;
  appel récursif $(root_{type} \rightarrow filsBasDroit, root_{orientation} \rightarrow filsBasDroit)$;
**fin**

Cet algorithme requiert au préalable une condition, les deux quadtrees $\mathcal{Q}_T$ et $\mathcal{Q}_O$ doivent posséder la même structure. Or, nous avons vu que, même si elles se ressemblent beaucoup, ce n'est pas toujours le cas. Une opération est donc effectuée au préalable pour obtenir la structure minimum recouvrant les deux quadtrees, l'algorithme est présenté en 6. La figure 3.3 illustre le principe de la structure minimum de $\mathcal{Q}_T$ et $\mathcal{Q}_O$. Au décodage des quadtrees, l'opération de simplification précédemment étudiée et illustrée sur la figure 3.2 permet d'obtenir les quadtrees d'origine. Cette opération, en plus d'offrir l'avantage de pouvoir utiliser l'algorithme précédent améliorant la compression, permet également de ne coder qu'une seule structure puisqu'elle est commune aux deux quadtrees. La réduction de débit grâce à cet algorithme dépend de la proportion de zone uniformes et texturées dans l'image. Plus ces zones sont présentes, meilleure est la compression. En pratique, nous constatons un gain moyen d'environ 25% pour l'encodage des deux quadtrees par rapport à

l'approche classique.

**Algorithme 6** : Algorithme d'équilibrage entre deux quadtrees selon la structure minimum

**Entrées** : cellule $root_{type}$, cellule $root_{orientation}$

**si** $root_{type}$ *est une feuille* $\wedge root_{orientation}$ *est un noeud* **alors**
  $temp \leftarrow \{root_{type} \rightarrow valeur\}$;
  subdiviser($root_{type}$);
  $\{root_{type} \rightarrow filsHautGauche \rightarrow valeur\} \leftarrow temp$;
  $\{root_{type} \rightarrow filsHautDroit \rightarrow valeur\} \leftarrow temp$;
  $\{root_{type} \rightarrow filsBasGauche \rightarrow valeur\} \leftarrow temp$;
  $\{root_{type} \rightarrow filsBasDroit \rightarrow valeur\} \leftarrow temp$;
**fin**
**sinon si** $root_{type}$ *est un noeud* $\wedge root_{orientation}$ *est une feuille* **alors**
  $temp \leftarrow \{root_{orientation} \rightarrow valeur\}$;
  subdiviser($root_{orientation}$);
  $\{root_{orientation} \rightarrow filsHautGauche \rightarrow valeur\} \leftarrow temp$;
  $\{root_{orientation} \rightarrow filsHautDroit \rightarrow valeur\} \leftarrow temp$;
  $\{root_{orientation} \rightarrow filsBasGauche \rightarrow valeur\} \leftarrow temp$;
  $\{root_{orientation} \rightarrow filsBasDroit \rightarrow valeur\} \leftarrow temp$;
**fin**
**si** $root_{type}$ *est un noeud* $\wedge root_{orientation}$ *est un noeud* **alors**
  appel récursif
  $(root_{type} \rightarrow filsHautGauche, root_{orientation} \rightarrow filsHautGauche)$;
  appel récursif
  $(root_{type} \rightarrow filsHautDroit, root_{orientation} \rightarrow filsHautDroit)$;
  appel récursif
  $(root_{type} \rightarrow filsBasGauche, root_{orientation} \rightarrow filsBasGauche)$;
  appel récursif $(root_{type} \rightarrow filsBasDroit, root_{orientation} \rightarrow filsBasDroit)$;
**fin**

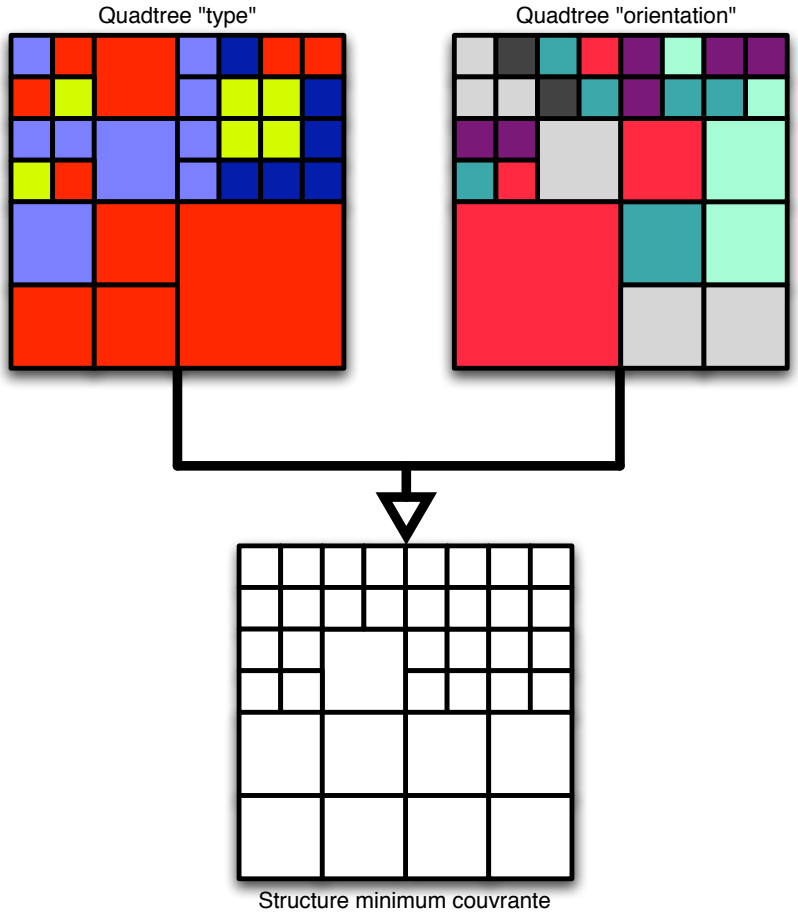



FIG. 3.3 – Exemple de structure minimum couvrante entre deux quadtrees

### Utilisation d'un codeur arithmétique

Dans un contexte de compression des données, le recourt à un codeur arithmétique de type adaptatif [Howard et Vitter, 1992] permet une reconstruction exacte de la séquence d'origine tout en offrant une réduction de débit importante. Dans cette partie, nous conservons les idées développées précédemment :

- en connaissant la hauteur maximum de l'arbre quaternaire, le codage pour indiquer le type des cellules du dernier niveau est inutile ;
- la corrélation entre les deux quadtrees permet de ne pas coder toutes les valeurs de l'arbre des directions privilégiées, en prenant soin au préalable d'harmoniser leurs structures.

Le codage arithmétique permet, à partir de la probabilité d'apparition des symboles d'une source de créer un seul mot-code qui soit associé à une séquence de longueur arbitraire de symboles. Ceci diffère du code de Huffman qui attribue des mots-codes de longueurs variables à chaque symbole de la source. Le codage arithmétique permet de coder chaque symbole sur un nombre de bits non nécessairement entier. Cet avantage permet de réaliser une compression plus performante. Les performances du codage arithmétique peuvent être améliorées dès lors que les tables de probabilités des symboles ne sont pas fixées, mais adaptées à la séquence courante et fonction des

séquences précédentes. Ce procédé que nous utilisons est appelé adaptatif. De base, le code associé à une séquence est un nombre réel de l'intervalle $[0, 1[$. Pour éviter les problèmes liés à la représentation des nombres flottants au sein des machines, nous utilisons un codeur arithmétique basé sur les entiers, très largement employé.

L'article [Said, 2004] expose les algorithmes utilisés en détail. Il faut surtout retenir les informations suivantes :

- à chaque symbole est attribué une probabilité d'apparition ;
- plus la probabilité d'apparition d'un symbole est grande, moins il faut de bits pour le coder ;
- dans un système adaptatif, les probabilités attribuées à chaque symbole évoluent de façon identique au codage/décodage en fonction des symboles déjà traités.

De plus, nous utilisons une connaissance *a priori* pour coder plus efficacement les symboles, c'est le codage avec prise en compte du "passé". Pour mettre en place ce système, nous manipulons plusieurs ensemble de données probabilistes, un ensemble correspond à une situation particulière du symbole à coder. Cette connaissance du "passé" va différer entre le codage de la structure et des valeurs du quadtree.

En ce qui concerne le codage de la structure, l'idée que nous avons développée est que la probabilité qu'une cellule soit un noeud ou une feuille dépend de la hauteur de la cellule dans l'arbre quaternaire. En effet, nous constatons que plus la cellule est proche de la racine de l'arbre, moins elle a de "chance" d'être une feuille. Nous élaborons donc des classes en fonction de la hauteur des cellules auxquelles nous attribuons une ensemble de deux probabilités correspondant aux chances d'être un noeud ou une feuille. Prenons l'exemple déjà cité dans lequel une image de $512 \times 512$ pixels est décomposée en blocs de tailles minimales $8 \times 8$ pixels. Nous savons que l'arbre à au maximum 7 niveaux. L'information sur les cellules du dernier niveau de l'arbre est superflue puisqu'elles sont obligatoirement des feuilles. Nous initialisons donc six ensembles de probabilités, qui peuvent par exemple être données sur le tableau 3.1. La classe 0 correspond aux cellules du premier niveau de l'arbre, c'est-à-dire à la racine. Dans l'exemple précédent, la probabilité que la racine de l'arbre soit une feuille est 1%, c'est évidemment la tendance que nous observons dans la réalité. Plus nous descendons dans les branches de l'arbre, plus la probabilité entre les deux types de cellule s'équilibrent. À l'avant dernier niveau de l'arbre, les chances sont équiprobables. Ces probabilités sont données à titre d'exemple et nous verrons dans la partie expérimentation quels chiffres donnent les meilleurs résultats à l'initialisation des ensemble de probabilités.
Le recourt à un système adaptatif va permettre de faire évoluer ces probabilités en fonction des symboles déjà traités. Par exemple, une image très détaillée va entraîner une faible corrélation des étiquettes entre blocs adjacents, contrairement à une image flou. De ce fait, l'arbre quaternaire associé sera composé de feuilles de tailles réduites et donc les classes de 1 à 5 auront tendance à favoriser les probabilités d'avoir un noeud au fur et à mesure du traitement (la classe 0 ne concerne qu'une seule cellule, la racine, donc le système adaptatif est ici dénué d'intérêt).

Pour le codage des valeurs des quadtrees $\mathcal{Q}_T$ et $\mathcal{Q}_O$, la prise en compte de l'environnement des valeurs des feuilles pour la réalisation des classes passe par l'analyse des feuilles adjacentes déjà traitées. En effet, il existe une forte corrélation entre les

| Classe | Probabilités | |
|---|---|---|
| | **d'une feuille** | **d'un noeud** |
| 0 | 0.01 | 0.99 |
| 1 | 0.05 | 0.95 |
| 2 | 0.10 | 0.90 |
| 3 | 0.20 | 0.80 |
| 4 | 0.35 | 0.65 |
| 5 | 0.50 | 0.50 |

TAB. 3.1 – Exemple d'ensembles de probabilités en fonction de la classe des cellules d'un quadtree

valeurs de feuilles géographiquement voisines puisque pour une image naturelle, les zones de même nature (de type et d'orientation similaires) ne se cantonnent pas à une forme strictement carrée, et sont décomposées en une multitude de plus petits carrés. Cependant, pour une feuille $f$ donnée, il existe un nombre limité de feuilles adjacentes déjà traitées. Pour un parcourt en profondeur de l'arbre de gauche à droite et de haut en bas, seules les feuilles voisines en haut et à gauche (si elles existent) de la feuille $f$ peuvent renseigner sur la valeur de $f$. Un exemple est disponible sur la figure 3.4, où les cases blanches représentent les feuilles pas encore traitées. En analysant le voisinage de la feuille $f$, nous pouvons en déduire qu'elle a de forte probabilités d'être étiquetée bleue.

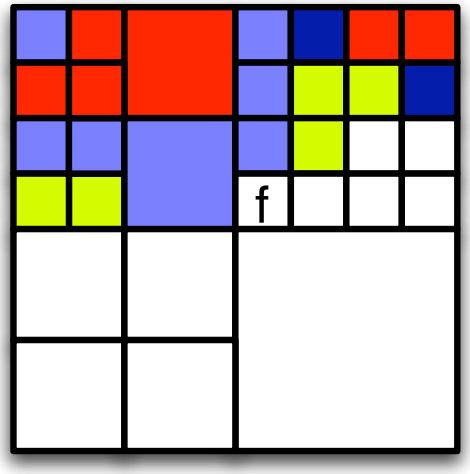



FIG. 3.4 – Exemple de prise en compte du voisinage de la feuille $f$ pour estimer sa valeur

La méthode utilisée consiste à définir autant de classes qu'il y a d'étiquettes possibles. Pour le codage des étiquettes de $\mathcal{Q}_T$, quatre classes sont donc définie. Pour chacune des classes, nous attribuons une probabilité d'apparition à chaque symbole. Par exemple, pour le quadtree $\mathcal{Q}_T$, nous pouvons avoir quelque chose qui ressemble à 3.2. Enfin, la classe de la feuille à coder est définie en prenant la valeur la plus représentée des pixels voisins de la feuille déjà traités, montré par la figure

3.5. Dans cet exemple, le nombre de feuilles voisines les plus représentées sont de classe 1 et 2, mais les pixels adjacents sont majoritairement de classe 0. La classe 0 sera donc retenue pour prédire la valeur de $f$.

| **Classe** | **Probabilités** | | | |
|---|---|---|---|---|
| | **Uniforme** | **Mono-orienté** | **Multi-orienté** | **Texture** |
| 0 *uniforme* | 0.4 | 0.2 | 0.2 | 0.2 |
| 1 *mono-orienté* | 0.2 | 0.4 | 0.2 | 0.2 |
| 2 *multi-orienté* | 0.2 | 0.2 | 0.4 | 0.2 |
| 3 *texture* | 0.2 | 0.2 | 0.2 | 0.4 |

TAB. 3.2 – Exemple d'ensembles de probabilités pour coder les valeurs de $\mathcal{Q}_T$

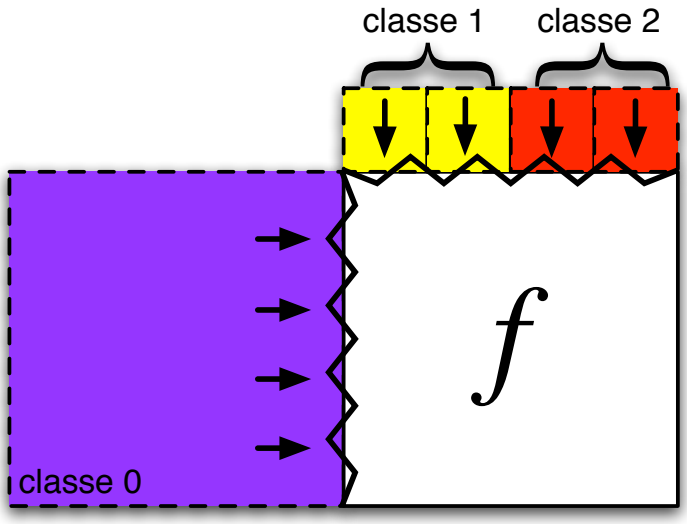



FIG. 3.5 – Exemple de prise en compte du voisinage de la feuille $f$ pour estimer sa valeur

Pour le quadtree $\mathcal{Q}_T$, la feuille située dans le coin en haut à gauche de l'image n'aura pas de connaissance sur son environnement, les chances sont donc équiprobables. Pour les valeurs du quadtree $\mathcal{Q}_T$, le cas est plus complexe, puisqu'une feuille peut être entourée de feuilles uniformes ou texturées étiquetées de façon arbitraire par le symbole "0°", qui ne correspond forcément à la tendance locale de l'image. Il peut donc s'avérer que plusieurs feuilles du quadtree soient privées de connaissance sur leurs environnements, nous avons alors recourt à un système équiprobable.

## 3.3 Lois de quantification en fonction du contenu local

Nous disposons d'informations concernant les zones d'activités de l'image. Nous allons maintenant nous employer à utiliser ces données à bon escient afin d'adapter le pas de quantification en fonction de la localisation spatiale des coefficients. La section précédente montre que l'image est segmentée et la classification s'opère selon quatre classes, à savoir les zones uniformes, les zones orientée selon une direction privilégiée, les zones orientées selon plus d'une direction et les zones texturées, où l'activité est élevée mais sans direction particulière.

### 3.3.1 Les zones uniformes

Les zones uniformes représentent les régions où l'activité est faible, c'est-à-dire où le contraste est minime. Un exemple de zone uniforme est illustrée sur la figure 3.6. Dans ces régions, les signaux hautes fréquences sont inexistants. L'idée est d'utiliser un pas de quantification très large, à l'exception de la sous-bande basse fréquence. Ainsi, les coefficients ondelettes, initialement petit en vue de la faible activité, deviennent des indices à 0, ce qui engendre un coup de codage moindre. La sous-bande basse fréquence suffira à elle seule à restituer un signal visuellement correct.

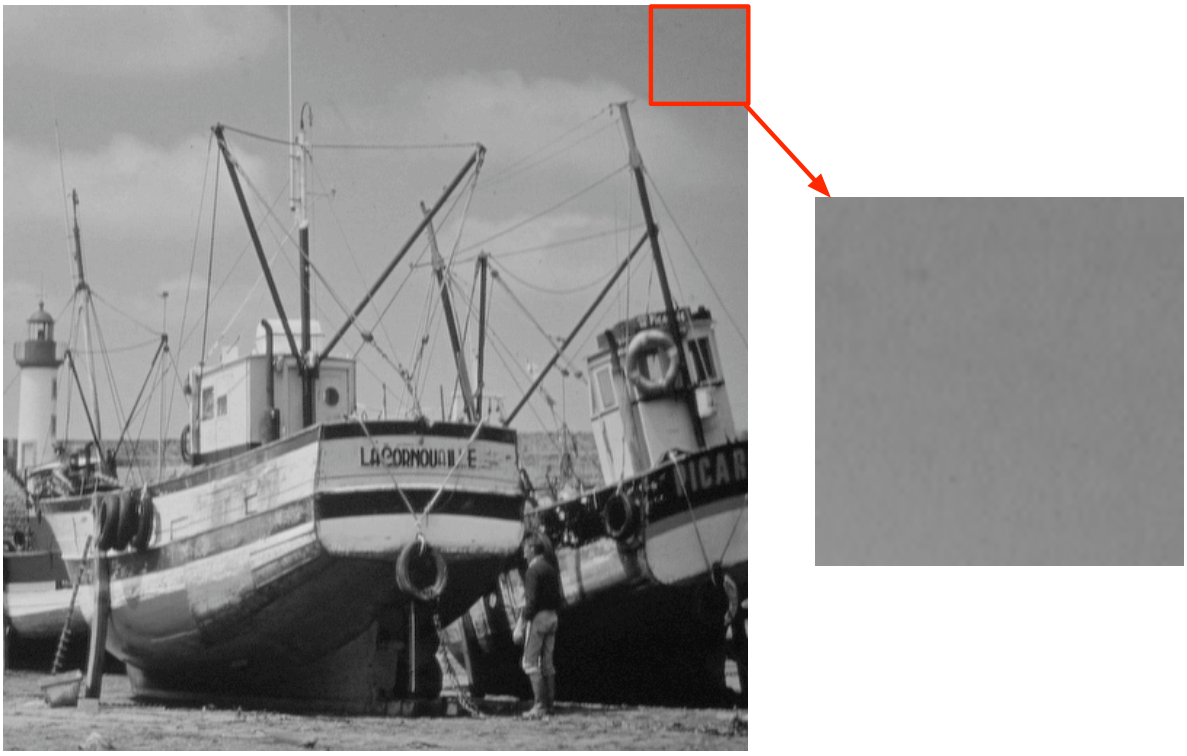


FIG. 3.6 – Exemple de zone uniforme avec l'image "boats"

### 3.3.2 Les zones mono-orientées

Les zones mono-orientées sont constituées de signaux orientés dans la même direction. Un exemple est illustré 3.7 avec la photo “barbara”. Pour ces zones, l’intention que nous avons est de quantifier plutôt finement la sous-bande en particulier qui contient l’information des signaux (*HFortho*, et d’élargir le pas de quantification pour les deux autres sous-bandes *HFreg* et *HFBiOrtho*. En effet, la sous-bande *HFOrtho* est issue du *lifting* orienté réalisé dans la direction orthogonale aux contours, c’est donc dans celle-ci que se concentre l’information.

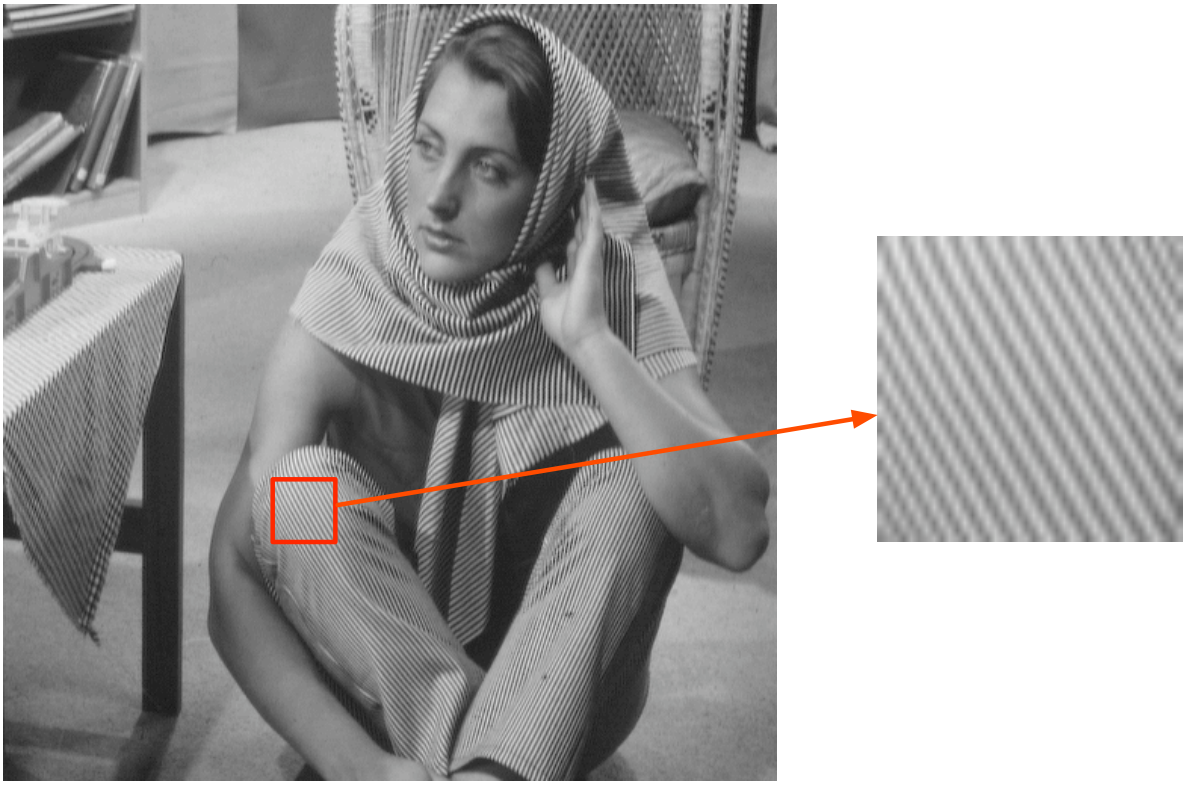


Fig. 3.7 – Exemple de zone mono-orientée avec l’image “barbara”

### 3.3.3 Les zones multi-orientées

Dans ces zone se trouvent des contours marqués orientés dans différentes directions. La figure 3.8 montre ce cas, où les mâts et les cordages des bateaux se croisent. Ces contours nets constituent des signaux forts. La quantification de ces zones est délicate puisque l’information est répartie dans toutes les sous-bandes. L’orientation du contour le plus marqué indique la direction du *lifting* réalisé, donc la sous-bande *HFortho* contient une partie assez conséquente de l’énergie. Pour le reste, nous disposons d’aucune information sur l’orientation des autres contours de la zone, qui peut est proche du contour principal (dans ce cas l’énergie du contour est principalement dans *HFortho*), orthogonale au contour principal (l’information

se retrouve en grande partie dans *HFreg*) où ni l'un ni l'autre (l'information est donc dans *HFBiOrtho*. La quantification des ces zones va donc être classique, avec un pas de quantification assez fin et identique pour toutes les sous-bandes.
Heureusement, ces zones sont plutôt rares puisque la segmentation très fine de l'image engendre des blocs de petites tailles qui ne contiennent rarement plus d'un contour.

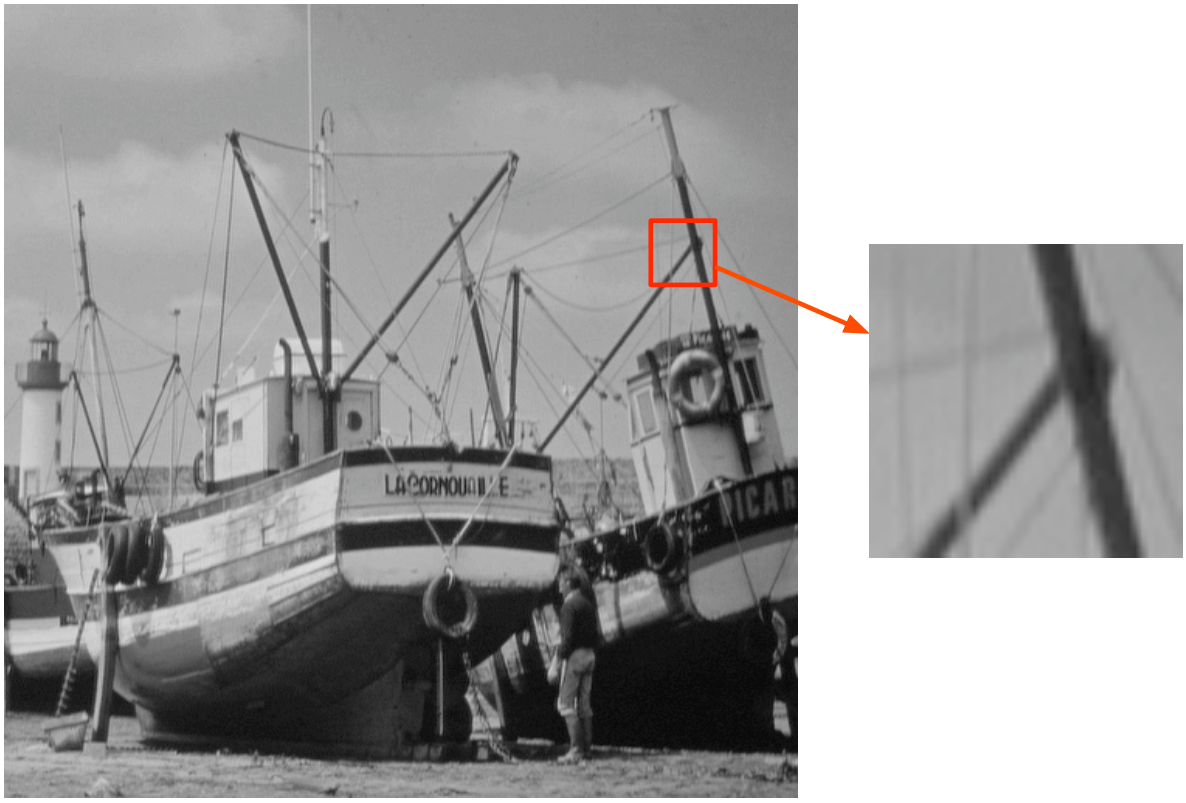


FIG. 3.8 – Exemple de zone multi-orientée avec l'image "boats"

### 3.3.4 Les zones texturées

Enfin, les zones texturées ont un contraste élevé mais aucune direction n'est privilégiée, comme illustré sur la figure 3.9. L'énergie de ces régions est donc réparti sur les trois sous-bandes de façon identique. Cependant, du fait de leurs fortes activités, ces régions sont sujettes aux effets de masquage spatial. En reprenant le concept de Zeng Daly ([Zeng *et al.*, 2000]) et de la méthode *Point-wise extended visual masking*, nous pouvons quantifier ces zones moins finement sans occasionner de pertes visuelles.

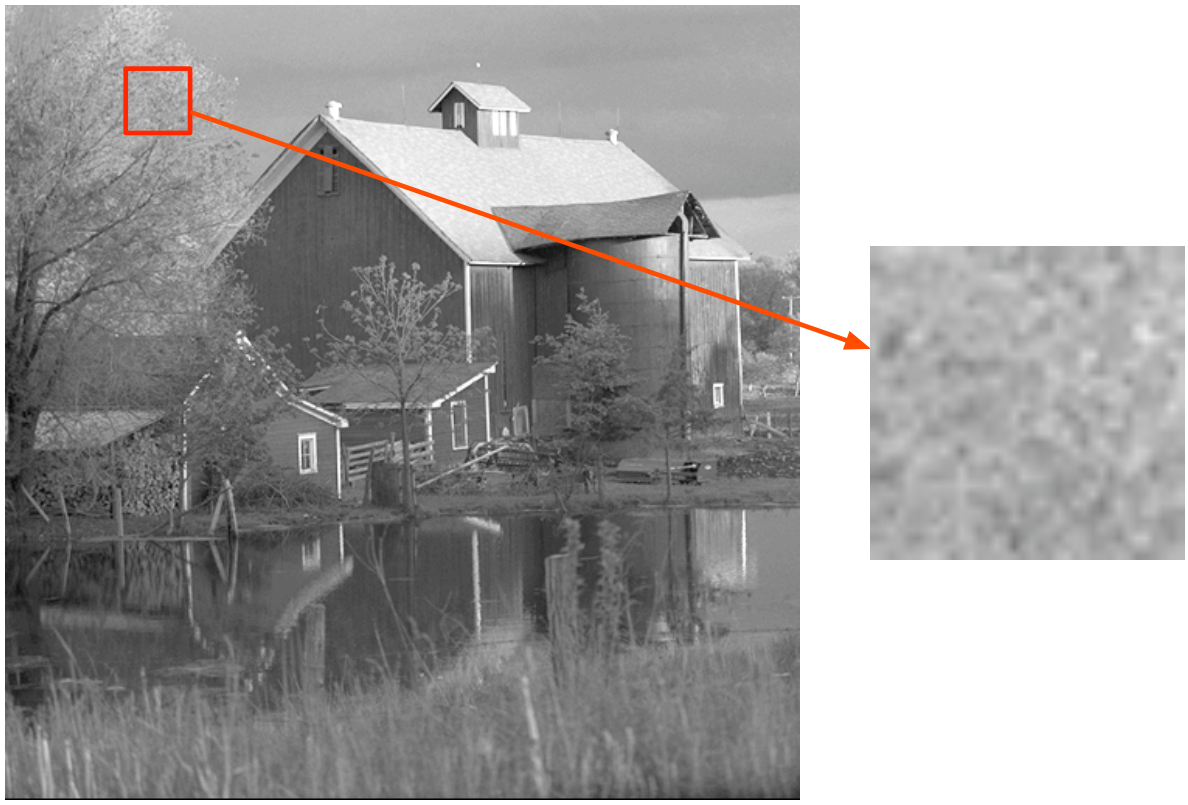


FIG. 3.9 – Exemple de zone texturée avec l'image "farm"

## 3.4 Stratégie de codage

Dans la partie bibliographique, nous avons étudié l'algorithme du codeur SPIHT. La démarche qui va suivre consiste à utiliser les informations que nous avons en amont pour améliorer les performances du codeur en terme de débit. Ces informations concernent la segmentation et la classification que nous avons décrit précédemment. Dans un premier temps, nous allons mentionner les possibilités qui s'offrent à nous pour optimiser SPIHT. Ensuite, nous étudierons la façon de construire le "masque" sur lequel repose la méthode et enfin nous proposerons une nouvelle version de l'algorithme.

### 3.4.1 Principe

L'origine de notre démarche est venue du constat que nous avons fait en analysant la décomposition en ondelette des zones uniformes. L'énergie des sous-bandes contenant les signaux à moyenne et haute fréquence est quasi-nulle pour ces zones. Ajouté à cela une quantification grossière, les indices qui s'y trouvent sont majoritairement égaux à "0". De ce fait, l'algorithme SPIHT tel qu'il est décrit par [Said et Pearlman, 1996] transmet ces coefficients en toute fin dans le flux binaire. Pourtant, même si ces indices ne sont pas transmis si le train binaire est tronqué, une quantité non négligeable de bits est forcément transmise pour indiquer au décodeur

le chemin parcouru dans l'arbre spatial orienté. Or, grâce au quadtree $\mathcal{Q}_T$, les zones où il y a peu d'activité sont connues à l'avance par le codeur et le décodeur. Les zones en question sont :

- les zones uniformes dans les sous-bandes *HFReg*, *HFOrtho* et *HFBiOrtho* ;
- les zones mono-orientées dans les sous-bandes *HFReg* et *HFBiOrtho* (l'énergie est concentrée dans la sous-bande *HFOrtho*).

Cependant, un point qu'il faudra éclaircir pendant les phases de tests portera sur le nombre de niveaux de décomposition que seront concernés par cette astuce. Il y aura un compromis entre débit et qualité qu'il faudra trouver, puisque plus il y a aura de niveaux de décomposition visés, plus le rendu visuel sera susceptible d'être dégradé. La première étape consiste à élaborer un masque en fonction des quadtrees $\mathcal{Q}_T$ pour chaque niveau de décomposition ondelette. Ensuite, nous présenterons le nouvel algorithme.

### 3.4.2 Construction du masque

La construction du masque va suivre plusieurs règles :

- le masque est de type booléen, il indique les indices à coder ou non ;
- le masque est une matrice qui se base sur l'arbre spatial de SPIHT, donc sur l'image transformée ;
- les niveaux de décomposition sur lesquels le masque est calculé sont connus du codeur et du décodeur ;
- un indice volontairement non codé ne peut pas avoir pour descendance des indices codés.

Ce dernier point s'explique par le déroulement de l'algorithme SPIHT qui se base sur la structure hiérarchique de l'arbre spatial. Pourtant, une zone uniforme du quadtree correspondant à une résolution particulière peut être étiquetée comme une zone mono-orientée pour la résolution supérieure. Pour illustrer ce cas, prenons par exemple une zone de l'image composée uniquement de signaux orientés à très haute fréquence. L'énergie de ces signaux se retrouve dans le premier niveau de décomposition ondelette, les décompositions suivantes seront uniformes. Nous devrons donc être vigilant sur la construction du masque pour prendre en compte ce détail.

Pour illustrer la construction de notre masque, nous prenons comme exemple les deux quatrees de la figure 3.10 qui correspondent à une transformée en ondelette sur deux niveaux de décomposition. La structure du masque pour une décomposition sur deux niveaux est illustrée sur la figure 3.11.

Ensuite, pour construire le masque, nous commençons par initialiser les trois sous-bandes de la résolution la plus grande. Dans l'exemple précédent, il s'agit de $HFReg_1$, $HFOrtho_1$ et $HFBiOrtho_1$. Cela donne la figure 3.12, les zones en gris sont "ignorées" par SPIHT. Pour la sous-bande $HFOrtho_1$, les indices ignorés appartiennent aux zones uniformes et mono-orientées. Pour les deux autres sous-bandes, seules les indices des zones uniformes sont ignorés.

Ensuite, la figure 3.13 illustre la construction du masque sur le deuxième niveau de décomposition. Nous ajoutons une contrainte par rapport au niveau précédent : les indices peuvent être éventuellement ignorés uniquement si les indices correspondant au niveau inférieur sont ignorés. La zone uniforme marquée d'une croix évoque ce

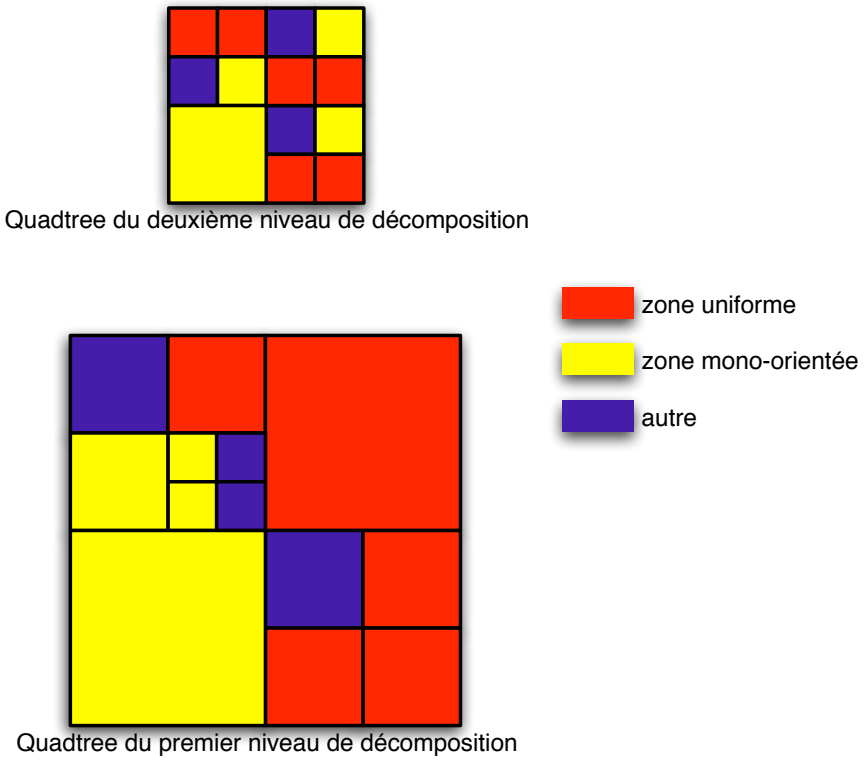



FIG. 3.10 – Exemple de quadtrees sur deux niveaux de décomposition

cas précis. En effet, les indices ne sont pas ignorés par le codeur SPIHT puisque la zone correspondante au niveau inférieur n'est pas ignorée.

Une fois le masque créé, nous l'incorporons à l'algorithme SPIHT. Le nouvel algorithme est présenté dans la section suivante.

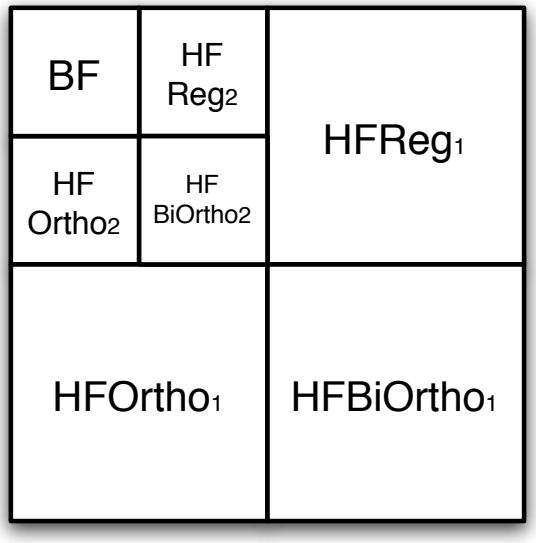



FIG. 3.11 – Structure du masque pour une décomposition en ondelette sur deux niveaux

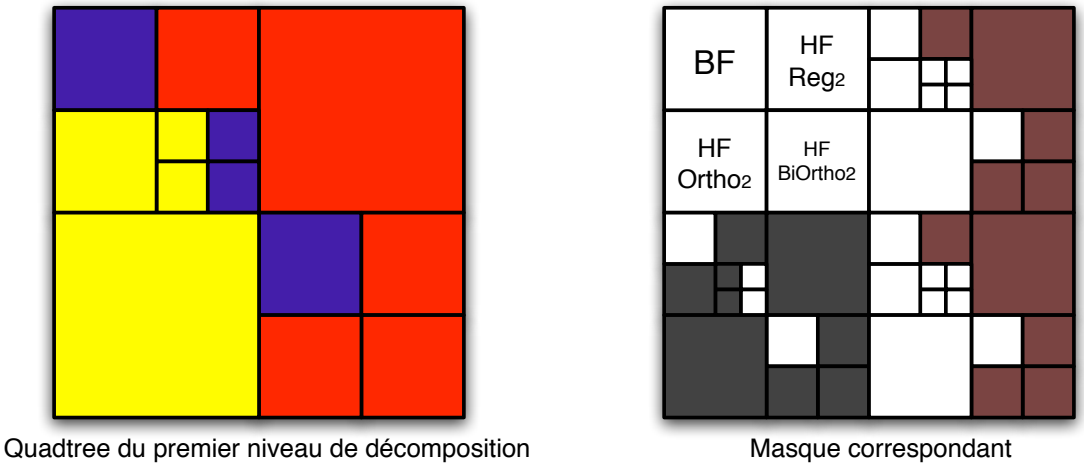



FIG. 3.12 – Construction du masque sur le premier niveau de décomposition

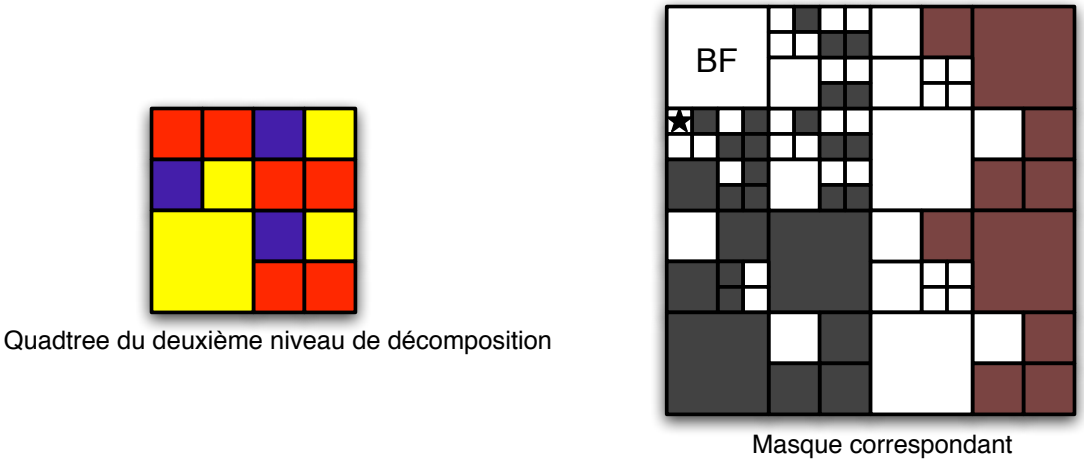



FIG. 3.13 – Construction du masque sur le deuxième niveau de décomposition

### 3.4.3 Algorithme proposé

Pour rappel, SPIHT est basé sur un algorithme de tri des partitions d'ensemble (voir partie bibliographie). Trois listes sont manipulées :

- LSP : liste des pixels significatifs ;
- LIP : liste des pixels non significatifs ;
- LIS : liste des ensembles non significatifs.

C'est dans cette dernière liste qu'un traitement supplémentaire nous permet d'ignorer les indices en fonction du masque. Avant de traiter les noeuds de type $A$ et de transmettre le bit de signification pour l'indice correspondant et ces quatre fils (comme il est fait dans l'algorithme standard), nous testons son éligibilité, à savoir si ses fils ne se situent pas dans une zone ignorée par le masque. Les algorithmes 7, 8, 9 et 10 présentent la nouvelle version de SPIHT. La partie modifiée de l'algorithme est soulignée.

**Algorithme 7** : Algorithme SPIHT proposé

$n = \lfloor \log_2(\max_{(i,j)} \{|c_{i,j}|\} \rfloor$;
$LSP = \emptyset$;
$LIP = H$;
`/* les noeuds racines */`
$LIS = \{(A, (i,j)), (i,j) \in H \wedge D(i,j) \neq \emptyset\}$;
`/* les noeuds racines ayant des descendants */`
$n = n - 1$;
Réitérer l'algorithme jusqu'à $n = 0$;

**Algorithme 8** : Traitement de la liste $LIP$

**Entrées** : liste de noeuds $LIP$
**pour chaque** $(i,j) \in LIP$ **faire**
    Envoie $S_n(\{i,j\})$;
    **si** $S_n(\{i,j\}) = 1$ **alors**
        ajout($(i,j)$,$LSP$);
        supprime($(i,j)$,$LIP$);
        Envoie signe de $c_{i,j}$;
    **fin**
**fin**

Bien entendu, la nouvelle version va avoir comme effet l'impossibilité d'une reconstruction parfaite, mais nous estimons que le débit gagné sera conséquent puisque non seulement certains indices ne sont pas codés, mais en plus le nombre de bits pour indiquer le parcourt de SPIHT dans l'arbre spatial sera réduit. D'un autre côté, le choix judicieux des indices ignorés permet de réduire au minimum la dégradation visuelle.

**Algorithme 9** : Traitement de la liste $LIS$

```
   Entrées : liste de noeuds LIS
 1 pour chaque (T,(i,j)) ∈ LIS faire
 2     si T = A alors
 3         si ∀(k,l) ∈ O(i,j), c_{k,l} est ignoré alors
 4             supprime((i,j),LIS);
 5         sinon
 6             Envoie S_n(D(i,j));
 7             si S_n(D(i,j)) = 1 alors
 8                 pour chaque (k,l) ∈ O(i,j) faire
 9                     Envoie S_n({k,l});
10                     si S_n({k,l}) = 1 alors
11                         ajout((k,l),LSP);
12                         Envoie signe de c_{i,j};
13                     sinon
14                         ajout_en_fin((k,l),LIP);
15                     fin
16                 fin
17                 si L(i,j) ≠ ∅ alors
18                     ajout_en_fin((B,(i,j)),LIS);
19                 fin
20                 supprime((A,(i,j)),LIS);
21             fin
22         fin
23     fin
24     sinon si T = B alors
25         Envoie S_n(L(i,j));
26         si S_n(L(i,j)) = 1 alors
27             pour chaque (k,l) ∈ O(i,j) faire
28                 ajout_en_fin((A,(k,l)),LIS);
29             fin
30             supprime((B,(i,j)),LIS);
31         fin
32     fin
33 fin
```

**Algorithme 10** : Traitement de la liste $LSP$

**Entrées** : liste de noeuds $LSP$

`/* Étape de raffinement */`

1 **pour chaque** $(i, j) \in LSP \wedge c_{i,j} > 2^{n+1}$ **faire**

2 | Envoie du bit de poids $n$;

3 **fin**

## 3.5 Conclusion et mode opératoire

Nous avons vu dans ce chapitre les points majeurs sur lesquels nous allons nous focaliser. Nous interagirons sur les trois domaines suivants :

- les informations additionnelles qui nous permettront de supprimer les données qui ne représente qu'un faible intérêt visuel ;
- les stratégies de quantification que nous pourrons améliorer grâce au premier point ;
- l'algorithme de codage SPIHT qui peut également utilisé les informations additionnelles pour améliorer son fonctionnement.

Le nouveau schéma de compression/décompression que nous proposons est illustré sur les figures 3.14 et 3.15.

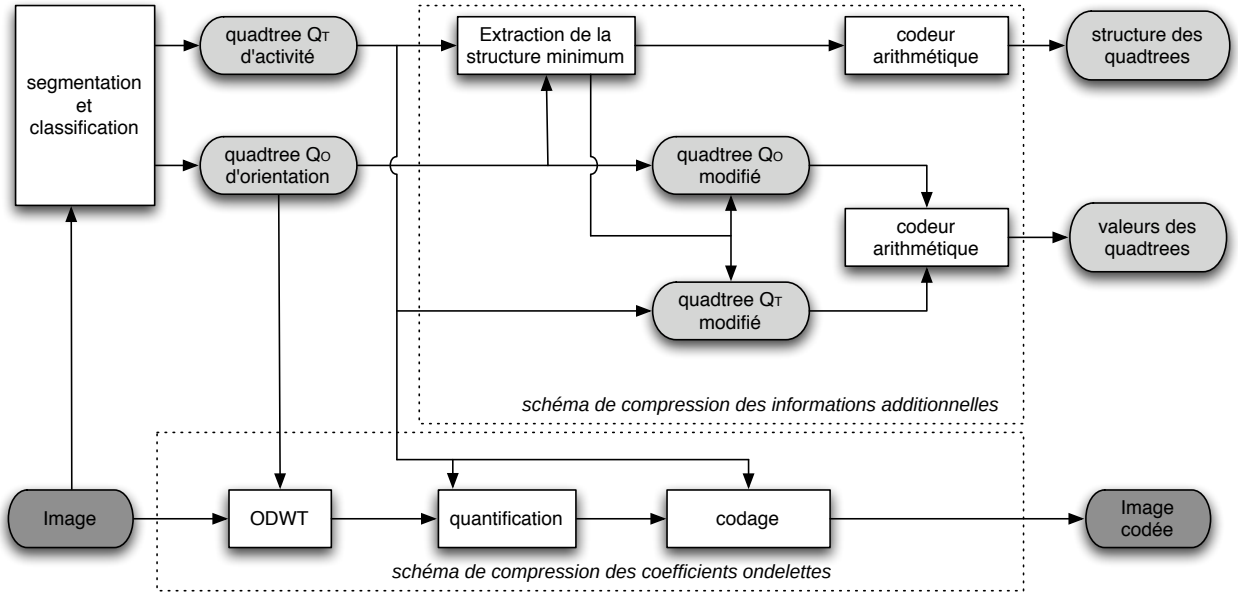



FIG. 3.14 – Récapitulatif du schéma de compression global proposé

Nous avons exposé notre démarche globale à travers ce chapitre. Cependant, plusieurs paramètres ne pourront être déterminés précisément qu'après la phase expérimentale, *via* des tests psychovisuels. Il s'agit en particulier des pas de quantifications en fonction de la classification, du nombre et du type de décompositions ondelettes

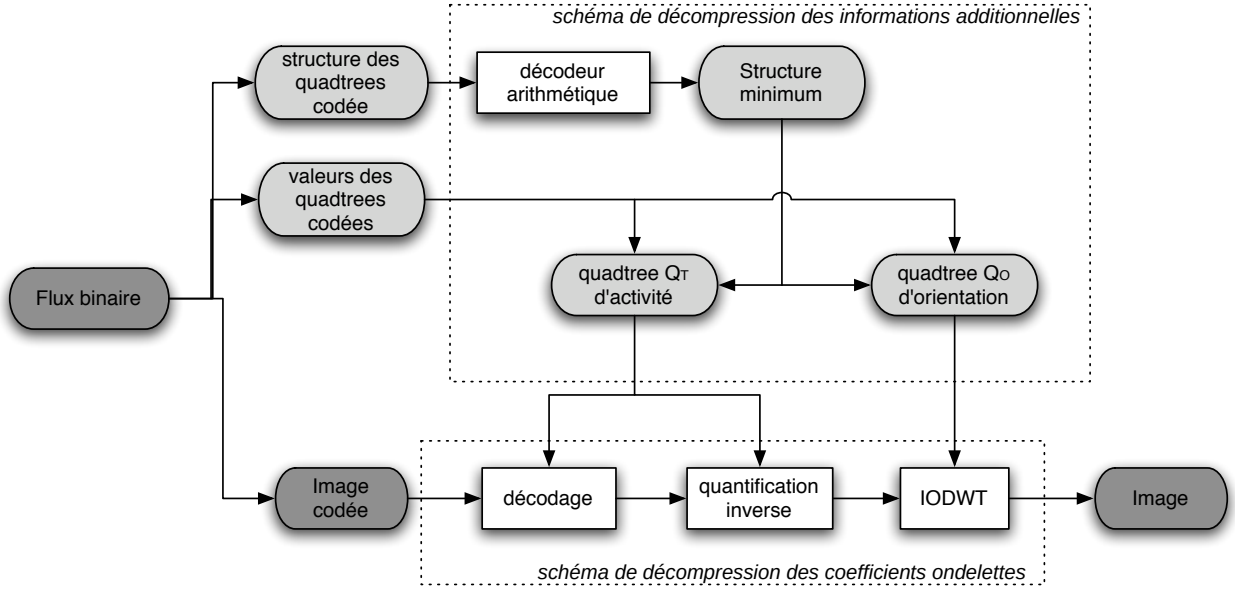



FIG. 3.15 – Récapitulatif du schéma de decompression global proposé

concernées par l'optimisation de SPIHT, etc. L'objectif principal sera de trouver le bon compromis entre débit et qualité visuelle. Pour évaluer la qualité visuelle d'une image, il est largement admis que le PSNR[1] se révèle particulièrement inadapté. En revanche, nous avons à notre disposition la métrique de qualité objective Komparator développée au sein du laboratoire de l'IRCCyN ([Barba et Callet, 2003]). Komparator travaille en *full reference*, c'est-à-dire qu'il considère l'image d'origine et l'image dégradée pour évaluer une note de qualité. Cette métrique est autrement adaptée aux problématiques de codage. Enfin, en ce qui concerne l'optimisation de SPIHT que nous proposons, nous disposons de la librairie *open source* Qccpack et de son codeur SPIHT pour comparer les performances sur deux points : le débit obtenu et la vitesse d'exécution.

[1]Peak Signal to Noise Ratio

# Chapitre 4

# Expérimentations et résultats

## Sommaire



## 4.1 Introduction

Le chapitre est décomposée en trois parties se référant aux propositions précédemment exposées. La première section abordée présente les résultats obtenus pour la segmentation et la classification des images. Dans un deuxième temps, nous exposerons les résultats selon les diverses stratégies de quantification adoptées, ce qui nous permettra d'en retenir la ou les meilleure(s). Enfin, nous présenterons le fruit de notre travail sur l'algorithme SPIHT, que nous comparerons à celui existant.

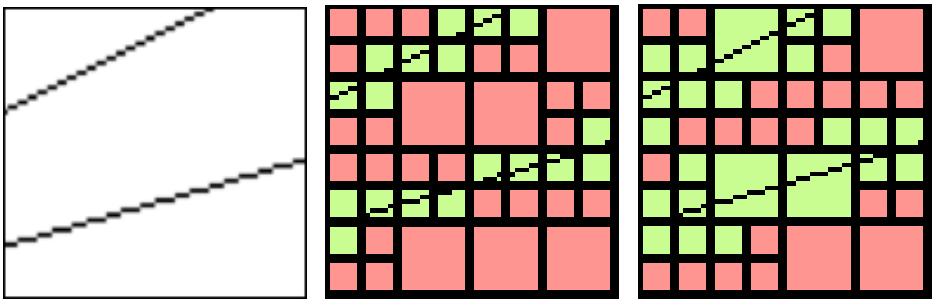
Pour réaliser les tests, nous possédons un ordinateur Macbook dont le processeur de type Core2Duo est cadencé à 2GHz, avec 2Go de mémoire RAM. Également, une station de travail de type Macpro est mis à notre disposition par l'équipe IVC. Les tests de performance en terme de rapidité d'exécution sont bien évidemment réalisés sur la même machine. L'ensemble des implémentations est réalisé avec l'environnement de développement Xcode.

## 4.2 Segmentation et classification

### 4.2.1 Résultats expérimentaux pour la classification

Avant de présenter les résultats finaux, nous allons exposer une modification que nous avons apporté pour la classification, car les résultats précédents ne nous satisfaisaient pas pleinement. L'optimisation consiste à élargir la fenêtre des pixels pour étiqueter chaque bloc. À l'origine, seuls les pixels d'un bloc étaient pris en compte pour mesurer l'activité du bloc. Cependant, la décomposition ondelette a pour conséquence de légèrement "étaler" l'énergie des signaux (imputé aux filtrages). Par exemple, un contour net et fin sera représenter par des coefficients ondelette localisés sur le contour et légèrement connexes.

Le problème que ce constat soulève est que des coefficients sont quantifiés selon l'étiquette d'un bloc (par exemple "uniforme"), alors qu'ils représentent un signal d'une autre nature (par exemple un contour net). Ce cas est illustré sur la figure 4.1. Pour remédier à ce problème, nous élargissons la fenêtre des pixels pour chaque bloc, sur laquelle est mesurée l'activité locale. Ainsi, si un contour est situé à proximité d'un bloc où l'activité est nulle, celui-ci sera étiqueté en fonction (c'est-à-dire mono-orienté au lieu d'uniforme). L'élargissement en nombre de pixels est dépendant du banc de filtre utilisé pour la décomposition ondelette. Pour des filtres 9/7, nous prévoyons un élargissement de de 4 pixels de la fenêtre. La figure 4.1 illustre les modifications.



(a) Image d'origine (b) Classification d'origine (c) Nouvelle classification

FIG. 4.1 – Illustration de la modification du procédé de classification.

Pour la classification, l'interprétation des résultats est difficile puisqu'elle repose uniquement sur des critères d'évaluation subjectifs. Cependant, nous pouvons dire que les résultats obtenus caractérisent assez bien les images, tant au niveau de la détection des orientations que de l'activité. Les résultats pour l'image "lena", d'une résolution de $512 \times 512$ pixels et avec une taille de bloc $\mathcal{B}$ de $8 \times 8$ pixels, sont illustrés sur la figure 4.2. Pour cet exemple comme pour tous les exemples suivants, le nombre de classes d'orientations retenu est huit. Les images originales sont fournies en annexe.

Nous pouvons constater que les deux classes les plus présentes sont les zones uniformes et mono-orientées, comme nous l'avions présumé en vue de la petite tailles de blocs $\mathcal{B}$. En conséquence, ceci représente un avantage certain puisque les stratégies

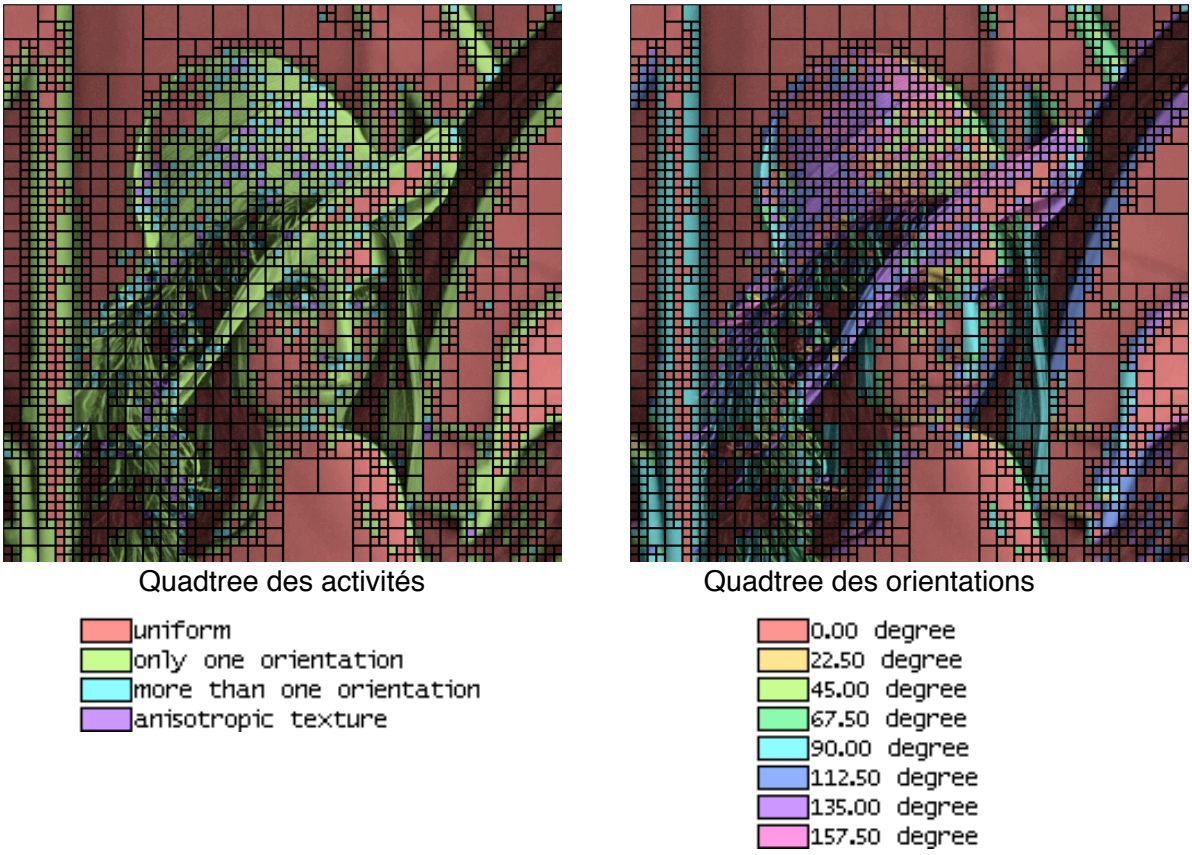



FIG. 4.2 – Résultat de la classification sur l'image "lena"

de quantification et de codage que nous avons élaboré reposent en grande partie sur ces types de régions. Ensuite, la segmentation et la classification réalisées sur la sous-bande $BF$ obtenue, donc une image de $256 \times 256$ pixels, est illustrée sur la figure 4.3. La légende, comme pour les autres résultats illustrées ci-après, est identique à la figure 4.2.

Nous pouvons effectuer le même constat que sur la figure 4.2 : les zones uniformes et mono-orientées sont majoritaires. Pour les sous-bandes basse fréquence du deuxième niveau de décomposition et plus, la composition des structures de l'image est fortement altérée dû aux sous-échantillonnage successifs. La classification des régions par activité ne donne pas de résultats exploitables (ce qui n'est pas forcément le cas pour la classification des orientations utilisée pour le *lifting*), c'est pourquoi nous avons décidé de nous limiter au premier niveau de décomposition ondelette des images de $512 \times 512$ pixels pour appliquer nos stratégies de quantification et de codage.

Nous allons maintenant nous focaliser sur le quadtree des régions par activité pour l'image entière et le premier niveau de décomposition, puisque ce sont ces derniers qui vont servir par la suite. Les figures 4.4, 4.5, 4.6 et 4.7 nous exposent les résultats pour diverses images.

De ces exemples, nous pouvons constaté plusieurs faits. Tout d'abord, la propor-

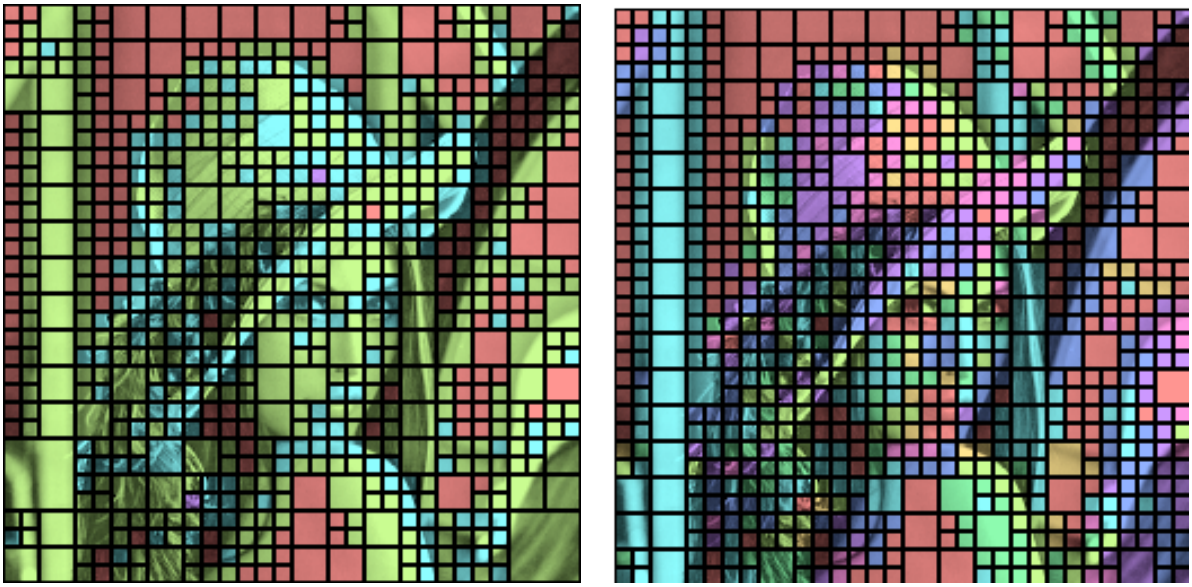



FIG. 4.3 – Résultat de la classification sur la sous-bande $BF$ de la première décomposition en ondelette de l'image "lena"

tion de zones uniformes est en net recul sur le quadtree de la $BF$ que sur l'image d'origine. Ce constat est dû au fait que la taille des blocs $\mathcal{B}$ est identique pour les deux quadtrees, alors qu'une image est deux fois plus grande que l'autre. Les zones uniformes s'en retrouvent donc diminuées pour la sous-bande $BF$, les plus petites sont étiquetées en fonction de l'activité des zones voisines. Cependant, même si la correspondance entre les deux quadtrees est du coup faible, ce résultat est cohérent pour les traitements de quantification et de codage ultérieurs.
Les zones représentant la végétation, notamment les arbre et les buissons, sont étiquetées en tant que texture. D'un autre côté, les zones des arrières plans flous et du ciel sont étiqueté comme uniformes. Ces résultats sont visuellement et objectivement cohérents.

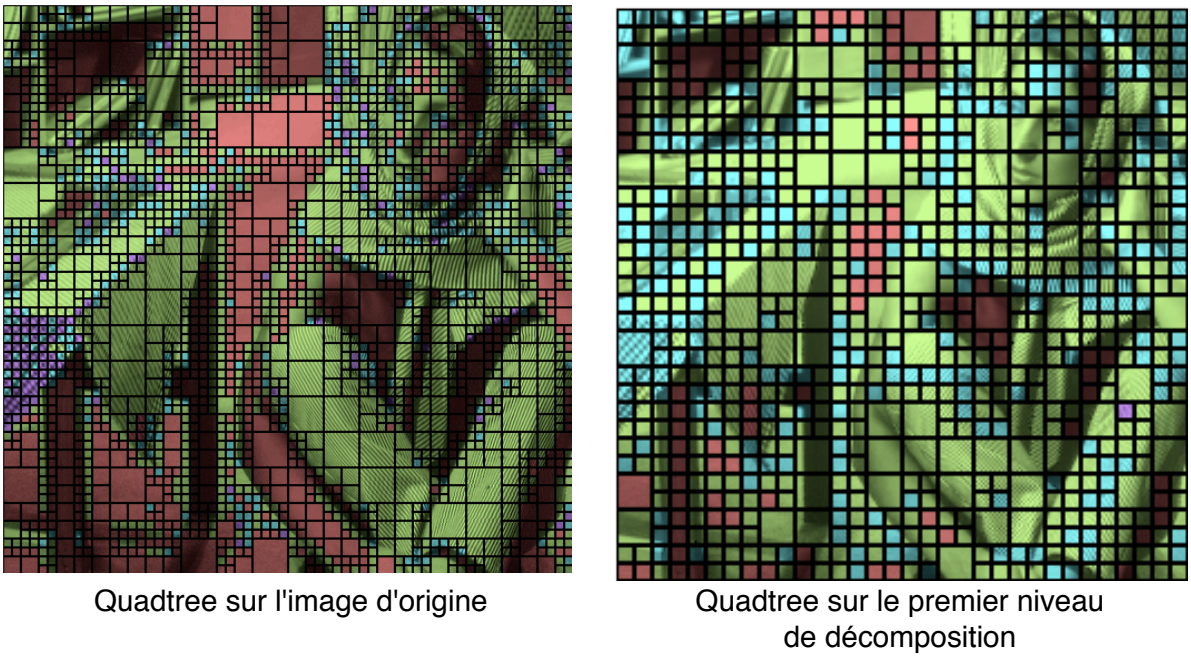



FIG. 4.4 – Résultats de la classification sur l'image "barbara" pour l'image d'origine et la sous-bande $BF$ obtenue après une décomposition en ondelette orientée

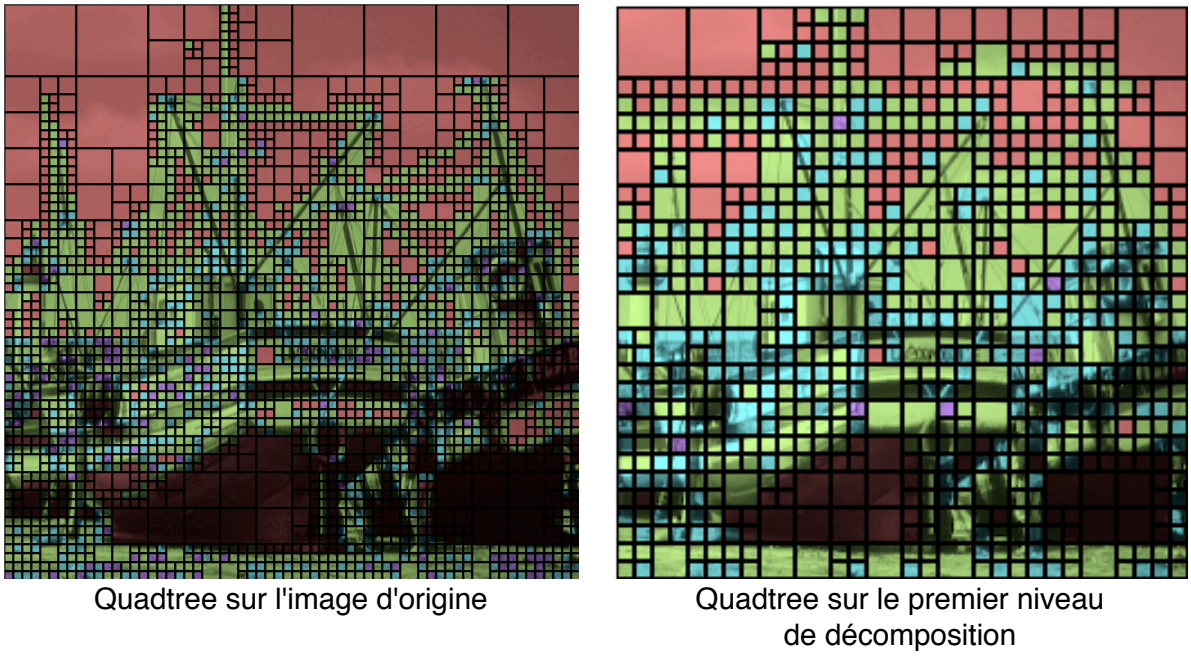



FIG. 4.5 – Résultats de la classification sur l'image "boats" pour l'image d'origine et la sous-bande $BF$ obtenue après une décomposition en ondelette orientée

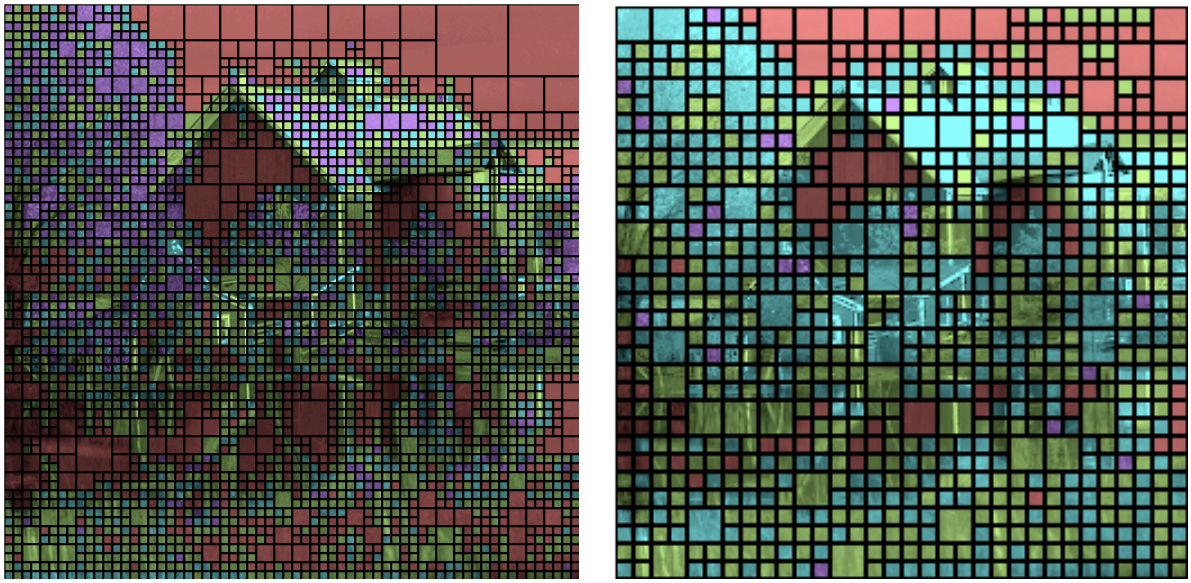



FIG. 4.6 – Résultats de la classification sur l'image "farm" pour l'image d'origine et la sous-bande $BF$ obtenue après une décomposition en ondelette orientée

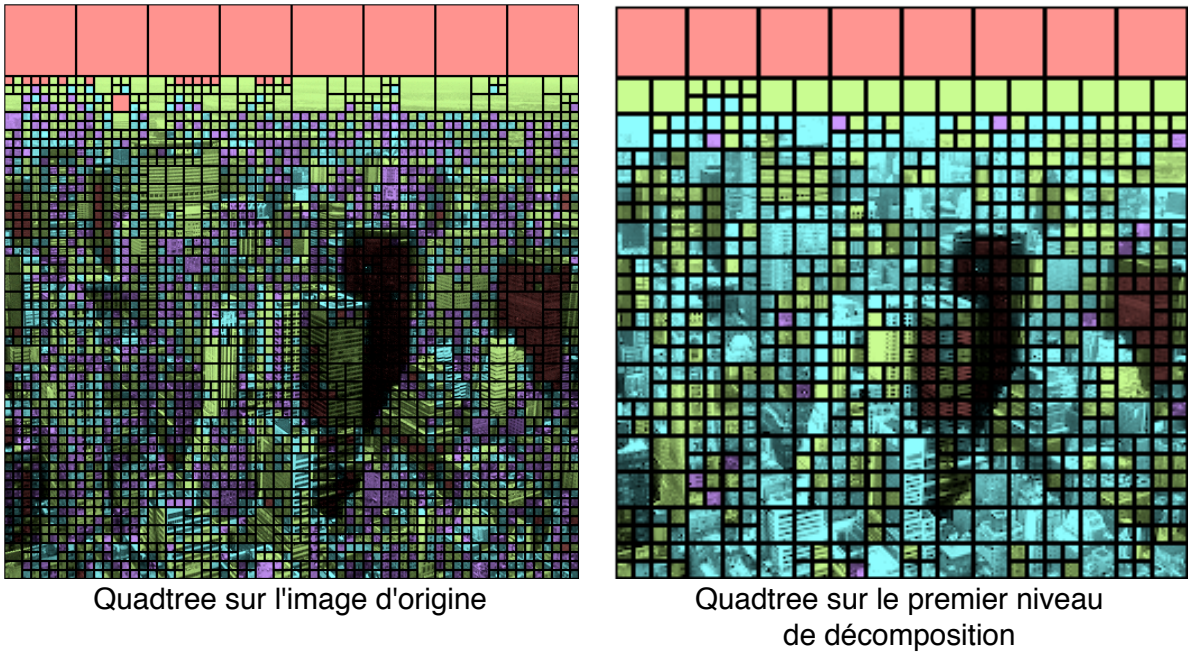



FIG. 4.7 – Résultats de la classification sur l'image "newyork" pour l'image d'origine et la sous-bande $BF$ obtenue après une décomposition en ondelette orientée

### 4.2.2 Résultats expérimentaux pour le codage de l'information additionnelle

Pour expérimenter les performances de codage des arbre quaternaires, nous avons réalisé trois séries de test se référant aux trois propositions précédemment exposées, de telle façon que :

- la première série utilise un codage classique des quadtrees ;
- pour la seconde série, nous avons utilisé les connaissances *a priori* et la corrélation entre les deux quatrees ;
- la troisième série utilise un codeur arithmétique adaptatif tel qu'il a été décrit.

Les résultats sont donnés par le tableau 4.1. Le débit en *bits/pixel* correspond au codage binaire des huit quadtrees nécessaires pour le compression/décompression des images (deux quatrees orientation et activité pour chacun des quatre niveaux de décomposition). Le temps donné en secondes coïncide au temps nécessaire pour coder/décoder les huit quadtrees.

| **Image** | **Série 1** | | **Série 2** | | **Série 3** | |
|---|---|---|---|---|---|---|
| | **débit (bpp)** | **durée (s)** | **débit (bpp)** | **durée (s)** | **débit (bpp)** | **durée (s)** |
| baboon | 0.165 | 0.015 | 0.123 | 0.040 | 0.093 | 0.079 |
| barbara | 0.131 | 0.016 | 0.092 | 0.036 | 0.069 | 0.067 |
| boats | 0.124 | 0.014 | 0.083 | 0.032 | 0.064 | 0.062 |
| farm | 0.145 | 0.015 | 0.096 | 0.038 | 0.077 | 0.071 |
| fruits | 0.151 | 0.016 | 0.098 | 0.040 | 0.078 | 0.073 |
| lena | 0.128 | 0.015 | 0.082 | 0.033 | 0.061 | 0.063 |
| newyork | 0.131 | 0.015 | 0.097 | 0.037 | 0.068 | 0.069 |
| plane | 0.131 | 0.259 | 0.089 | 0.038 | 0.068 | 0.067 |
| raft | 0.153 | 0.016 | 0.109 | 0.047 | 0.081 | 0.082 |
| train | 0.140 | 0.014 | 0.101 | 0.041 | 0.077 | 0.075 |
| yosemite | 0.130 | 0.015 | 0.084 | 0.037 | 0.067 | 0.067 |
| **MOYENNE** | 0.139 | 0.015 | 0.096 | 0.038 | 0.073 | 0.070 |
| **ECART-TYPE** | 0.013 | 0.001 | 0.012 | 0.004 | 0.009 | 0.006 |
| **GAIN MOYEN** | | | 31.1% | | 47.5% | |

TAB. 4.1 – Résultats du codage des arbres quaternaires

Nous pouvons constater que les gains moyens en terme de taille sont significatifs (47.5% pour le codeur arithmétique adaptatif par rapport au codage normal), mais cela au détriment du temps d'exécution. L'algorithme du codeur arithmétique étant le plus complexe, il lui revient la moins bonne performance en terme de temps. Dans un contexte où la compression prime sur le temps d'exécution, ces résultats sont satisfaisants. Nous pouvons également constater que les performances, que ce soit au niveau de la taille ou de la durée d'exécution, sont très similaires pour toutes les images de chaque série.

## 4.3 Qualité objective et subjective de l'image après quantification

Nous allons maintenant exposer les résultats des différentes stratégies de quantification présentées dans le chapitre précédent. Pour rappel, nous proposons une quantification différente pour les zones uniformes, mono-orientées et texturées. Les zones multi-orientées n'offrent pas de possibilités particulières, elles sont quantifiées de manière classique.
Nous allons nous focaliser sur les deux premiers niveaux de décomposition, puisque les niveaux suivants n'apportent pas d'information exploitable sur leurs activités pour des images d'origines de $512 \times 512$ pixels.

### 4.3.1 Les zones uniformes

L'expérimentation consiste à faire varier la pas de quantification uniquement pour les zones uniformes. Le calcul de l'indice de quantification est donné par 4.1.

$$Q(x) = sign(x) \left\lfloor \lambda \frac{|x|}{\Delta_U} \right\rfloor, \Delta_U \geq 1, \tag{4.1}$$

où $\lambda$ représente une constante de quantification. Plus $\lambda$ est grand, plus l'image reconstruite est détériorée. $\Delta_U$ est le paramètre qui nous permet de faire varier le pas de quantification. L'opération inverse de la quantification est donné par l'équation 4.2.

$$\overline{Q^{-1}}(q) = \begin{cases} 0 & q = 0 \\ sign(q)\frac{(|q|+\delta)\Delta_U}{\lambda} & q \neq 0 \end{cases} \tag{4.2}$$

où $\delta$ est un paramètre compris entre $0 \leq \delta < 1$ (typiquement $\delta = 1/2$). Avec $\delta = 1/2$, on obtient une valeur déquantifiée au milieu de l'intervalle $\Delta$ concerné, au lieu d'obtenir une valeur reconstruite sur une des bornes de l'intervalle.

#### Premier niveau de décomposition

Les coefficients ondelettes des autres zones sont conservés tels quel. Pour chaque image reconstruite, l'entropie et la distorsion du signal sont calculés. La mesure de la distorsion est donnée soit par le PSNR soit par Komparator (figure 4.8). Avec Komparator, plus la note donnée est proche de zéro, moins l'image est dégradée. Nous avons réalisé les tests avec quatre images, en prenant uniquement en compte les zones uniformes du premier niveau de décomposition. Le calcul de l'entropie, quant à lui, concerne uniquement les zones unifromes du premier niveau de décomposition, puisque les autres coefficients ne sont pas quantifiés.

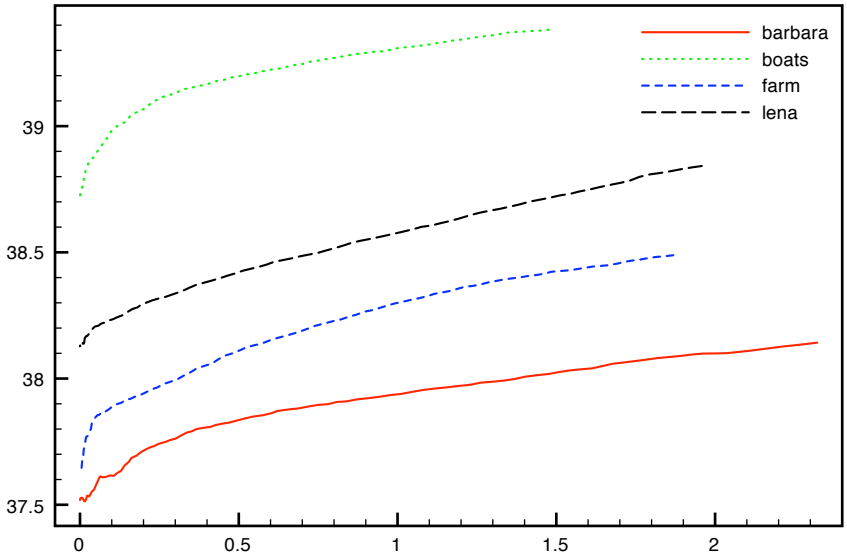
Nous intéressons à l'évolution des courbes. De ce fait, nous pouvons constater une contradiction entre les résultats du PSNR et de Komparator. Les résultats donnés par le PSNR s'accordent pour indiquer une chute de la qualité de l'image en quantifiant plus grossièrement les zones uniformes. D'un autre côté, les notes délivrées par Komparator sont très constantes, et elles sont les plus proches de ce que nous avons pu constater visuellement. En faisant varier *lambda*, les résultats sont

tous semblables.
La stratégie de quantification pour les zones uniformes du premier niveau de décomposition passe par l'interprétation de ces tests. Pour cela, les résultats donnés par Komparator nous confirment que les zones uniformes peuvent être quantifiées grossièrement, voir ignorées. En effet, l'entropie du signal est ainsi réduite, mais la qualité visuelle demeure identique.

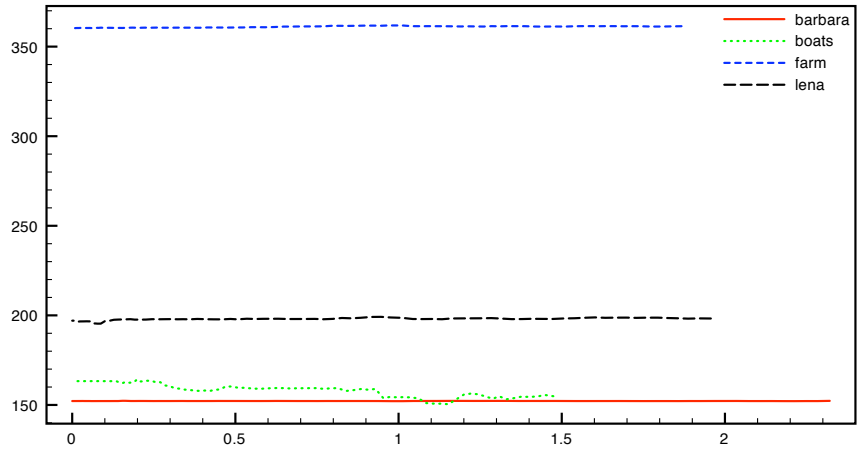
### Deuxième niveau de décomposition

Nous réalisons des tests identiques pour le second niveau de décomposition. La figure correspondante est 4.9. L'entropie est calculée uniquement sur le deuxième niveau de quantification. En ayant connaissance des précédents résultats nous quantifions les zones uniformes du premier niveau de décomposition très grossièrement.

Le constat est identique, la suppression d'information pour les zones uniformes n'affecte pas la qualité visuelle de l'image (excepté pour l'image "boats"). Ainsi, les zones uniformes des deux premiers niveaux de décomposition peuvent être ignorées sans occasionner de dégradations visuelles. Pour l'image "boats", nous avons pu analyser et comprendre l'origine des résultats. Les dégradations visuelles sont dues à la classification des zones. Lorsqu'un contour fin est à proximité d'un bloc étiqueté uniforme, les coefficients du bloc contribue à la restitution du contour même s'ils ne se situent pas exactement au niveau du contour. Or, ces coefficients sont quantifiés grossièrement et donc le contour n'est pas fidèlement reconstruit, ce qui occasionne des gênes visuelles. Une modification de la méthode de classification permet d'améliorer les résultats. Pour cela, chaque bloc est étiqueté en fonction des pixels qui le compose, mais aussi des pixels voisins. Cette méthode revient donc à élargir la fenêtre de calcul d'activité pour chaque bloc. Ainsi, si une zone uniforme est située à proximité d'un contour, le bloc sera étiqueté différemment.

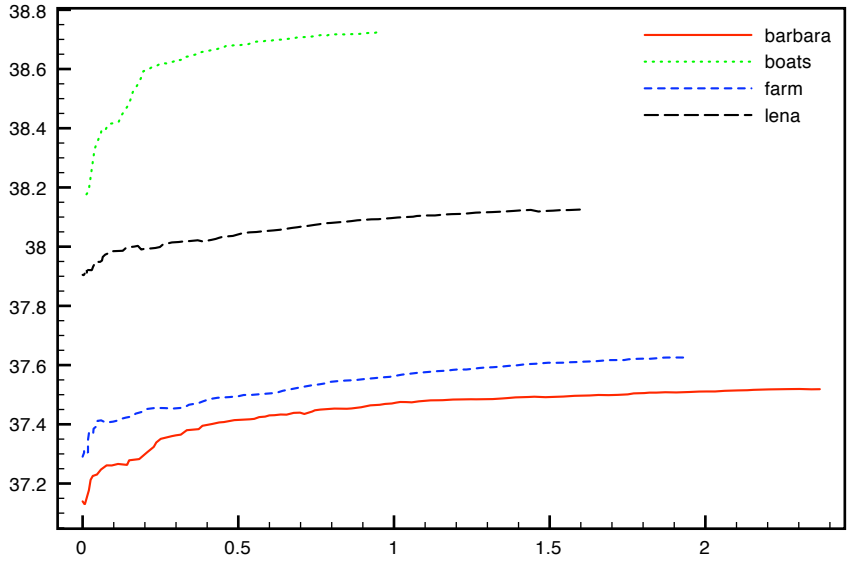



(a) PSNR

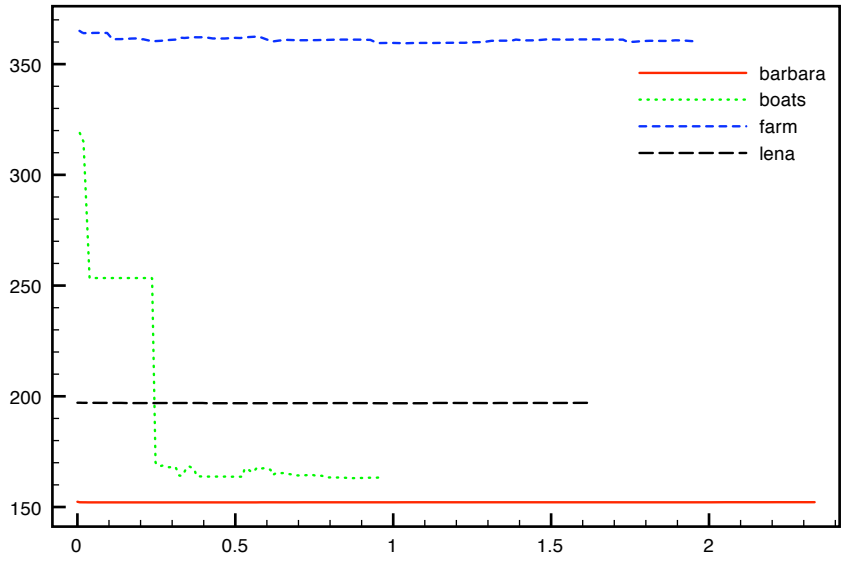



(b) Komparator

FIG. 4.8 – Résultat des courbes entropie-distorsion en prenant en compte les zones uniformes du premier niveau de décomposition, avec $\lambda = 0.1$.




(a) PSNR




(b) Komparator

FIG. 4.9 – Résultat des courbes entropie-distorsion en prenant en compte les zones uniformes du deuxième niveau de décomposition, avec $\lambda = 0.1$.

### 4.3.2 Les zones mono-orientées

#### Premier niveau de décomposition

Pour les zones mono-orientées, notre démarche va être semblable que précédemment, hormis le fait que le pas de quantification varie uniquement pour les sous-bande $HFReg$ et $HFBiOrtho$, puisque l'information est concentrée dans la sous-bande $HFOrtho$. De même, nous illustrons les résultats en prenant en compte le PSNR ou Komparator pour mesurer la distorsion. Pour commencer, nous nous focalisons sur la quantification des sous-bandes de la première décomposition en ondelette, avec la figure 4.10.

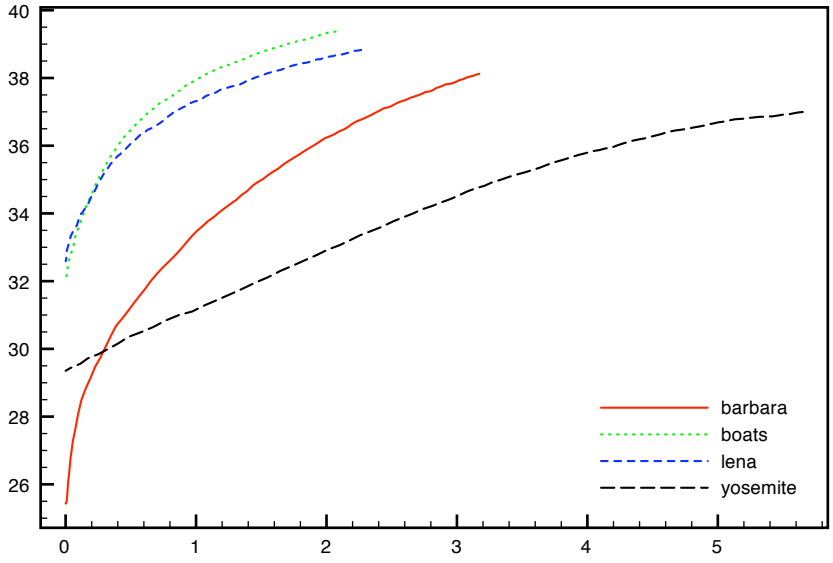
La courbe entropie-PSNR nous signale que l'image reconstruite est de plus en plus altérée lorsque l'entropie diminue. La courbe entropie-Komparator nous donne les résultats subjectifs, et nous pouvons constater que les notes de qualité sont assez disparates. Cependant, la tendance générale des courbes montre que la qualité des images est assez constante, même lorsque l'entropie est faible.

Dans l'ensemble, les résultats indiquent qu'une quantification grossière des sous-bandes $HFReg$ et $HFBiOrtho$,dans le cas des zones mono-orientées, n'affecte pas la qualité de l'image reconstruite.

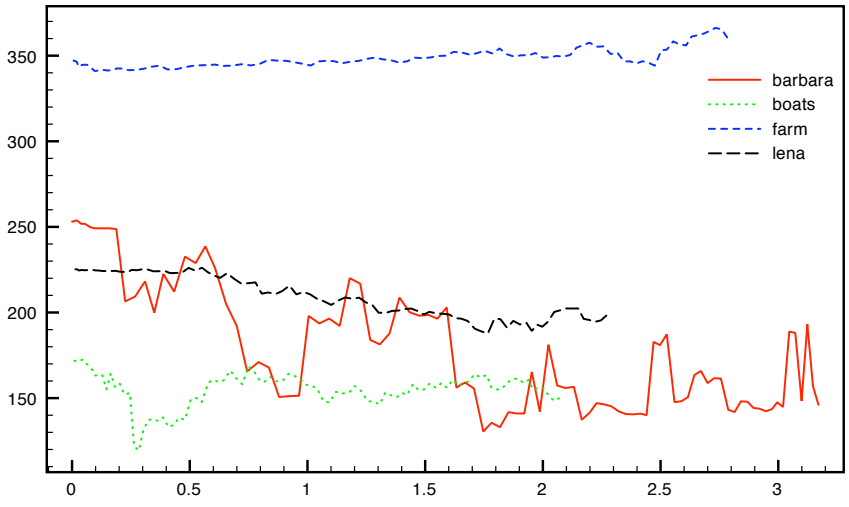
#### Deuxième niveau de décomposition

Dans un deuxième temps, nous avons mesuré les dégradations visuelles obtenues lors d'une quantification dynamique du deuxième niveau de décomposition, uniquement pour $HFReg$ et $HFBiOrtho$. Pour le premier niveau de décomposition, nous avons retenu la solution de ne pas prendre en compte les éléments des sous-bandes $HFReg$ et $HFBiOrtho$ pour les zones mono-orienées. En effet, tel que le montre le test précédent, nous n'avons pas jugé visuellement important ces informations. Les résultats obtenus sont illustrés sur la figure 4.11.

L'allure générale de ces courbes montre une forte dégradation des images lorsque l'entropie diminue. Pour conserver une qualité acceptable, nous devrons donc conserver ces coefficients, en les quantifiant avec un pas de quantification relativement petit. Les tests psycho-visuels réalisés par la suite vont nous permettre d'établir le seuil à partir duquel les dégradations sont visibles.

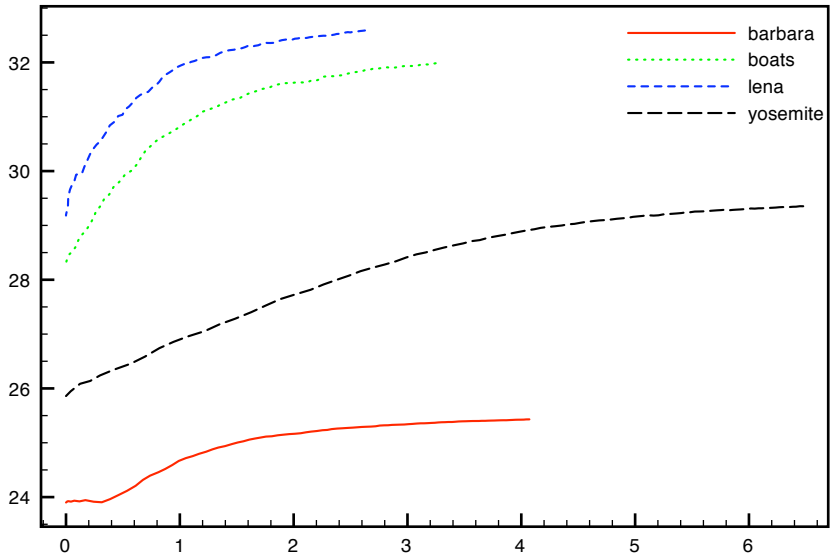



(a) PSNR

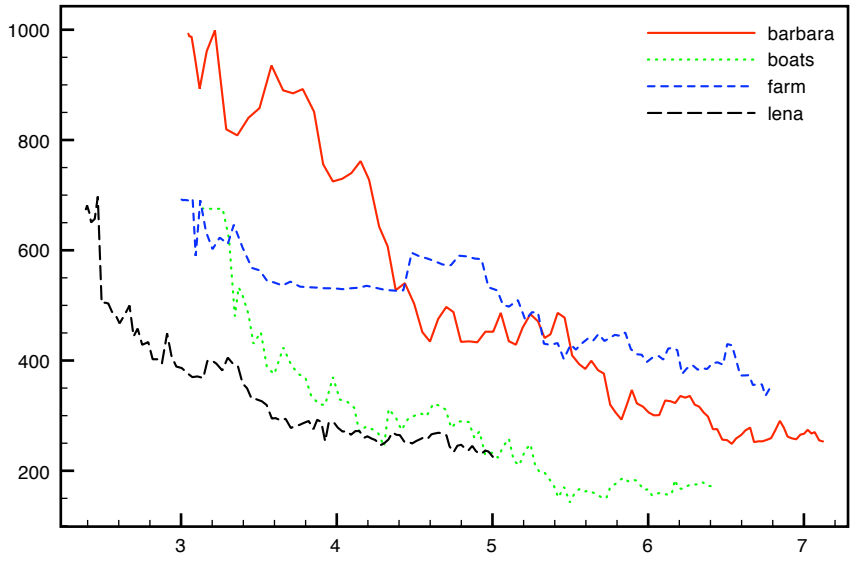



(b) Komparator

FIG. 4.10 – Résultat des courbes entropie-distorsion en prenant en compte les zones mono-orientées du premier niveau de décomposition, avec $\lambda = 0.1$.




(a) PSNR




(b) Komparator

FIG. 4.11 – Résultat des courbes entropie-distorsion en prenant en compte les zones mono-orientées du deuxième niveau de décomposition, avec $\lambda = 0.1$.

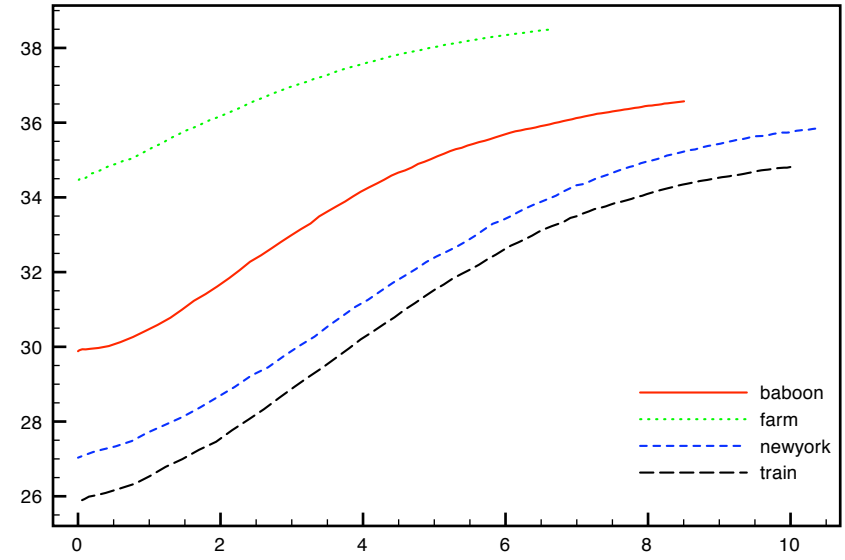
### 4.3.3 Les zones de textures

#### Premier niveau de décomposition

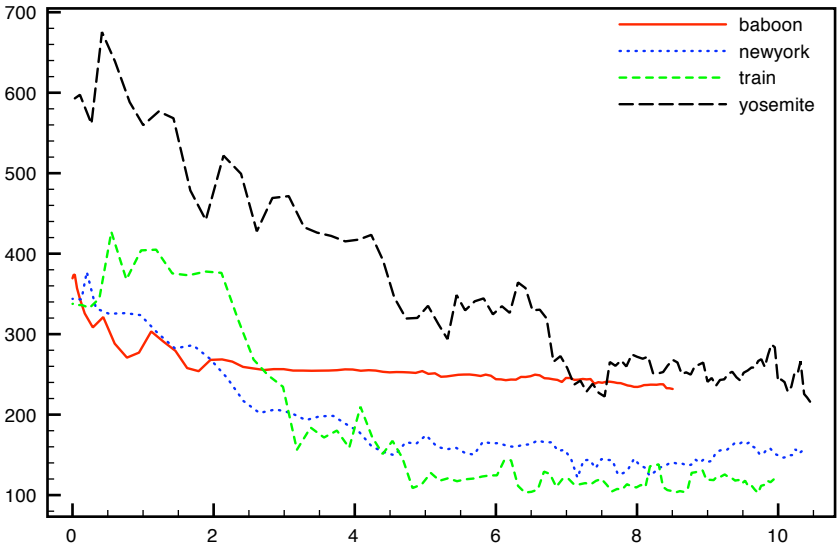
Nous appliquons le même procédé pour les textures. Le groupe d'images utilisé pour réaliser ces tests est différents des précédents, pour avoir des images constituées de plus de zones de textures. La variation d'entropie du premier niveau de décomposition s'en retrouve ainsi plus importante, et donc les résultats plus significatifs. La figure 4.12, basée sur le PSNR pour la mesure des distorsions, met en relief la détérioration flagrante de l'image en fonction de la quantification des zones de textures. Ce résultat était prévisible puisque la valeur des pixels reconstruits est très différente de l'originale. Cependant, les résultats fournis par Komparator confirment nos prédictions, puisque les images sont relativement bien conservées "visuellement" alors que l'entropie des sous-bandes est diminuée de façon significative.

#### Deuxième niveau de décomposition

Après ces résultats, nous nous intéressons aux zones de texture du deuxième niveau de décomposition. En appliquant le même procédé, nous obtenons la figure 4.13. Le constat est encore plus flagrant que pour le premier niveau de décomposition. La qualité des images est d'autant plus diminuée que l'entropie est faible. Nous pouvons cependant nous questionner sur les résultats donnés pas le PSNR, puisque les courbes sont assez constantes alors que l'entropie diminue. Pour comprendre ces résultats, nous devons nous remémorer que tous les indices des zones de texture du premier niveau niveau de décomposition sont quantifiés à 0, donc l'image est déjà fortement altérée. Le fait de quantifier plus grossièrement les indices du deuxième niveau de décomposition n'ajoute pas de distorsion au niveau du PSNR.

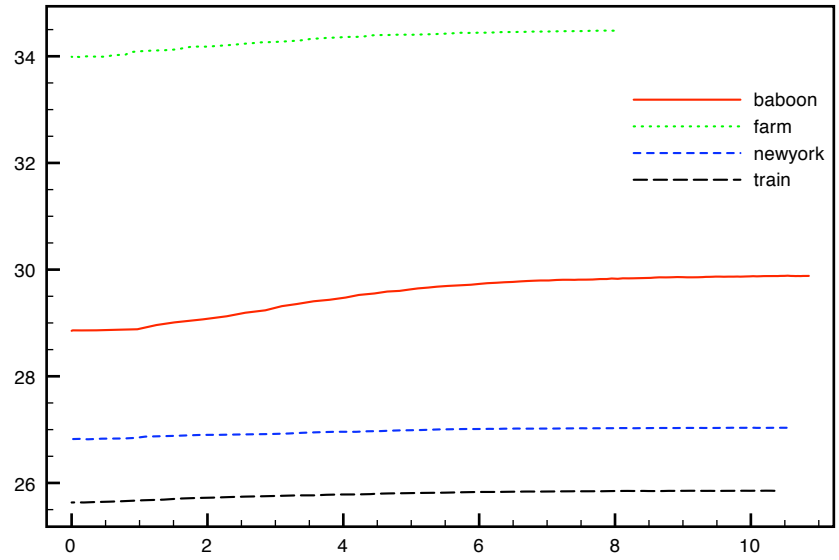



(a) PSNR

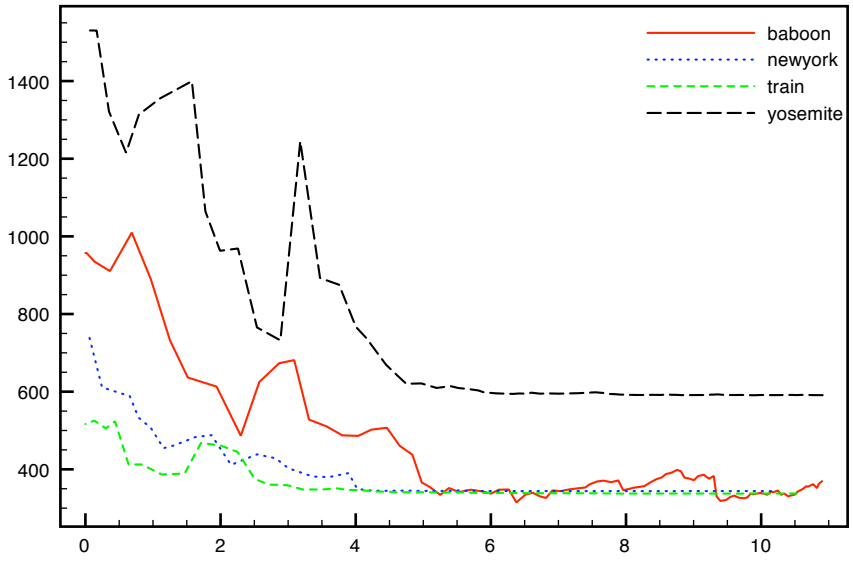



(b) Komparator

FIG. 4.12 – Résultat des courbes entropie-distorsion en prenant en compte les zones de textures du premier niveau de décomposition, avec $\lambda = 0.1$.




(a) PSNR




(b) Komparator

FIG. 4.13 – Résultat des courbes entropie-distorsion en prenant en compte les zones de textures du deuxième niveau de décomposition, avec $\lambda = 0.1$.

### 4.3.4 Les zones multi-orientées

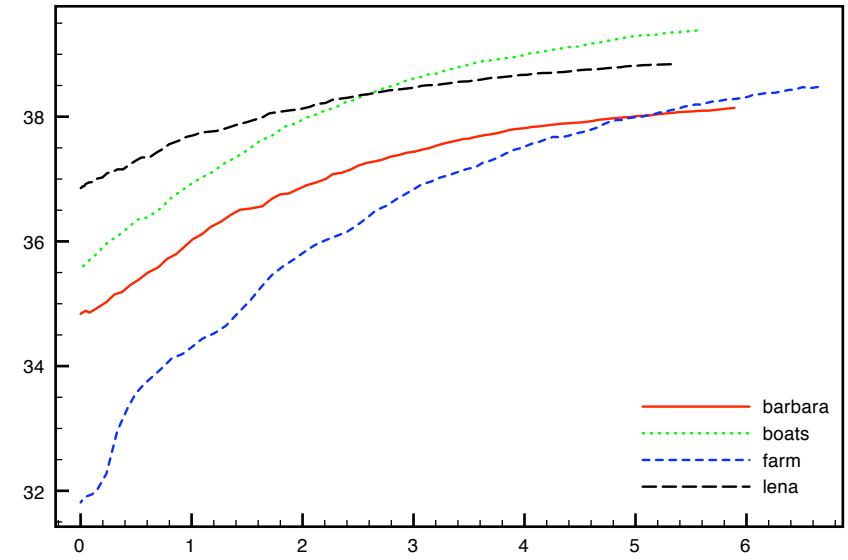
#### Premier niveau de décomposition

Les zones multi-orientées sont les plus difficiles à appréhender. En effet, leurs contenus sont très différents d'une image à l'autre. Ce constat recommande donc la plus grande prudence pour la stratégie de quantification. Nous avons donc choisi d'utiliser un pas de quantification petit pour ces zones. Les résultats de la figure 4.15 montrent que pour deux images différentes, l'évolution de leurs qualités en fonction de l'entropie varie. L'image "lena", par exemple, conserve une qualité identique, alors que l'image "barbara" est fortement dégradée lorsque l'entropie diminue.

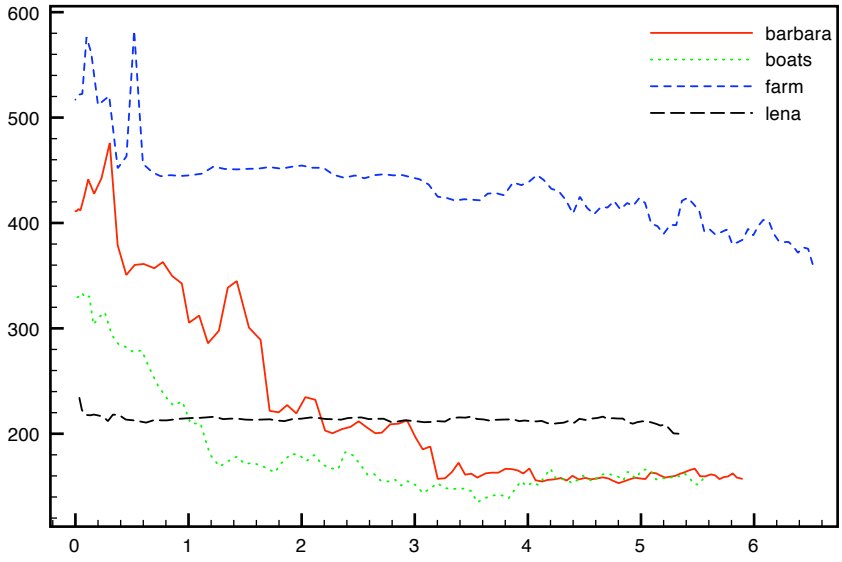
Au vu de ces résultats, nous avons décidé de ne pas ajuster la pas de quantification pour les zones constituées de contours orientés dans différentes directions. La pas de quantification est donc donné uniquement par $\lambda$.

#### Deuxième niveau de décomposition

Pour le deuxième niveau de décomposition, nous pouvons constater (figure 4.15) que les images sont très dégradées lorsque l'entropie baisse. Comme pour les zones multi-orientées du premier niveau de décomposition, nous choisissons de ne pas ajuster le pas de quantification, puisqu'une quantification grossière affecte énormément la qualité de l'image.

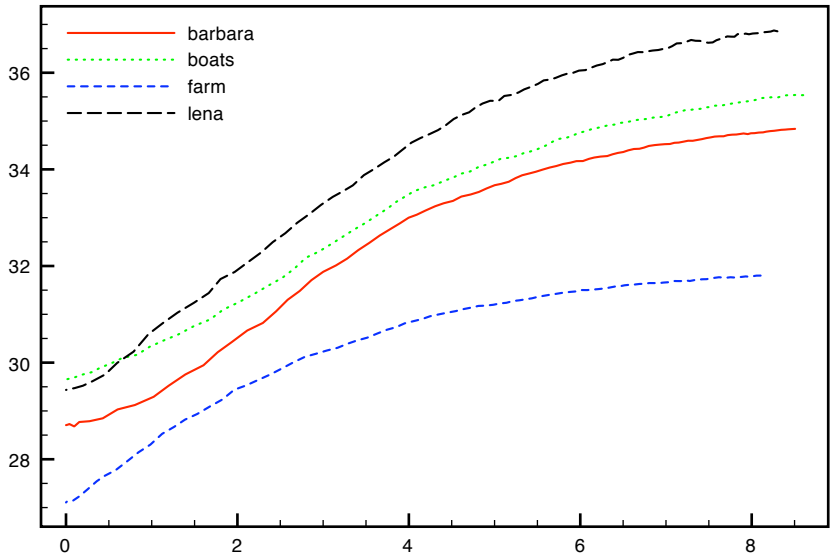



(a) PSNR

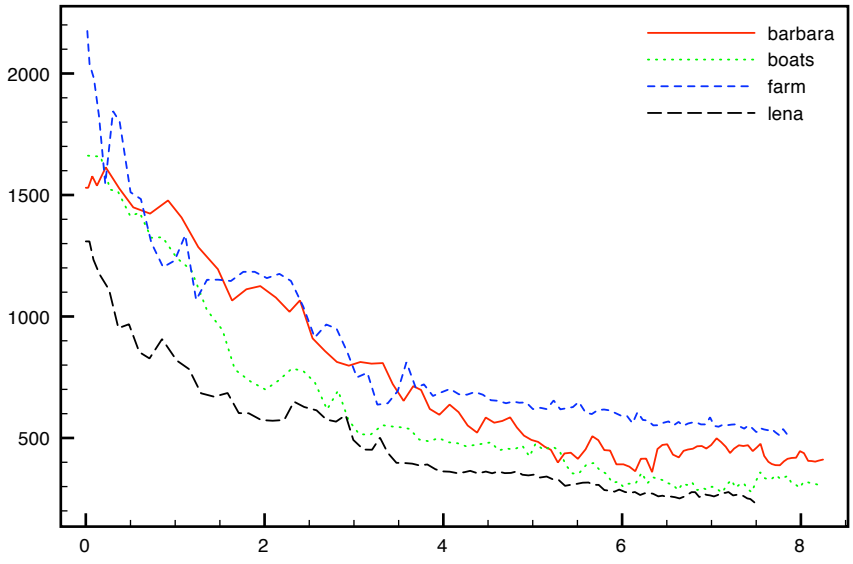



(b) Komparator

FIG. 4.14 – Résultat des courbes entropie-distorsion en prenant en compte les zones multi-orientées du premier niveau de décomposition, avec $\lambda = 0.1$.




(a) PSNR




(b) Komparator

FIG. 4.15 – Résultat des courbes entropie-distorsion en prenant en compte les zones multi-orientées du deuxième niveau de décomposition, avec $\lambda = 0.1$.

## 4.4 Résultats psycho-visuels

Les résultats précédents obtenus avec Komparator ne nous satisfont pas pleinement. Plusieurs paramètres n'ont pu être déterminés précisément. Par la suite, nous avons décidé de procéder à des tests subjectifs, grâce au matériel mis à disposition au laboratoire IVC. Nous disposons d'un écran CRT de marque Samsung de 20", avec une résolution native de $1024 \times 768$ pixels. La fonction de luminance pour chaque niveau de gris est donnée sur la courbe 4.16. La luminance maximum est $92,1 cd/m^2$. Pour extraire le gamma de ces informations, nous procédons à un ajustement de courbe par la fonction 4.3

$$y = a + b \cdot \left(\frac{x}{255}\right)^{\gamma}, \tag{4.3}$$

avec laquelle nous trouvons $\gamma = 1.85$.

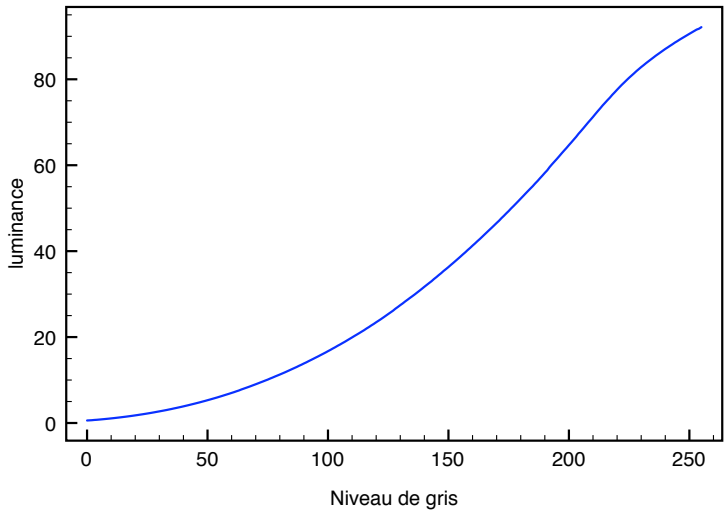



Fig. 4.16 – Fonction luminance de l'écran CRT utilisé pour les tests subjectifs.

## 4.5 Résultats de l'algorithme SPIHT proposé

### 4.5.1 Réalisation d'un codeur SPIHT standard

Dans un premier temps, nous avons implanter l'algorithme SPIHT standard, que nous avons comparer avec celui fourni par la librairie QccPack. Pour cela, nous réalisons un codage après une transformée en ondelette orientée et une quantification scalaire uniforme. Les résultats en terme de débit et de temps d'exécution sont fournis au tableau 4.2 avec un ensemble d'images testé. Le débit correspond à la totalité du flux binaire généré pour coder toute l'information du signal quantifié. Les images reconstruites sont donc rigoureusement identiques pour les deux solutions testées.

Nous pouvons observer que le débit pour coder toute l'information est sensiblement inférieur pour notre implémentation. Cette constatation est due au fait que

| **Image** | **QccPack** | | | **SPIHT réalisé** | | |
|---|---|---|---|---|---|---|
| | **débit (bpp)** | **codage (s)** | **décodage (s)** | **débit (bpp)** | **codage (s)** | **décodage (s)** |
| baboon | 2.71 | 5.30 | 5.03 | 2.24 | 0.74 | 0.30 |
| barbara | 1.88 | 3.70 | 3.08 | 1.50 | 0.63 | 0.19 |
| boats | 1.48 | 2.93 | 2.21 | 1.21 | 0.69 | 0.14 |
| farm | 1.80 | 3.72 | 3.24 | 1.44 | 0.63 | 0.19 |
| lena | 1.39 | 2.89 | 2.13 | 1.12 | 0.68 | 0.14 |
| **MOYENNE** | 1.85 | 3.7 | 3, 14 | 1.5 | 0, 674 | 0.19 |
| **GAIN MOYEN** | | | | 18.9% | 81.8% | 93.9% |

Tab. 4.2 – Comparaison du SPIHT standard réalisé avec QccPack.

QccPack doit probablement ajouter un nombre considérable d'information dans le flux binaire. L'autre observation que nous faisons est que notre implantation est beaucoup plus rapide en durée d'exécution. Ce constat est surprenant, mais s'explique par la complexité que représente la librairie QccPack, qui est destinée à une multitude d'autres possibilités. Désormais, nous avons un algorithme SPIHT standard qui va nous servir de base pour les modifications que nous allons y apporter et ainsi implanter notre propre proposition de codeur.

### 4.5.2 Réalisation de l'algorithme SPIHT proposé

#### Résultats pour un débit identique

Les figures 4.17 et 4.18 illustrent la comparaison entre l'algorithme SPIHT standard et l'algorithme SPIHT proposé. Ici, le masque ne se compose que des zones uniformes. Nous prenons en compte les trois premiers niveaux de décomposition, ce qui nous donne les quatre points de la courbe débit-distorsion de notre algorithme (sans prendre aucun niveau de décomposition pour construire le masque, en en prenant un, deux puis trois). La distorsion est calculée par le PSNR, qui indique mal la qualité subjective de l'image. L'idée est juste d'avoir un aperçu des améliorations par rapport à une approche classique. Le pas de quantification utilisé est constant pour tous les coefficients.

Nous constatons que pour des débits similaires, le SPIHT que nous proposons donne de meilleurs résultats au niveau de PSNR.

#### Résultats pour une qualité identique

Nous allons maintenant nous intéresser aux débits nécessaire pour coder la même image (par "même" image, nous entendons qualité visuelle similaire). En effet, les résultats précédents nous ont démontrés que les zones uniformes des deux premiers niveaux de décomposition ne sont pas nécessaire pour reconstruire une image fidèle à l'originale. De ce fait, nous utilisons l'algorithme SPIHT standard et l'algorithme SPIHT proposé qui fait abstraction des indices des zones uniformes des deux premiers niveaux de décomposition. Nous confrontons les résultats pour une collection d'images sur le tableau 4.3.

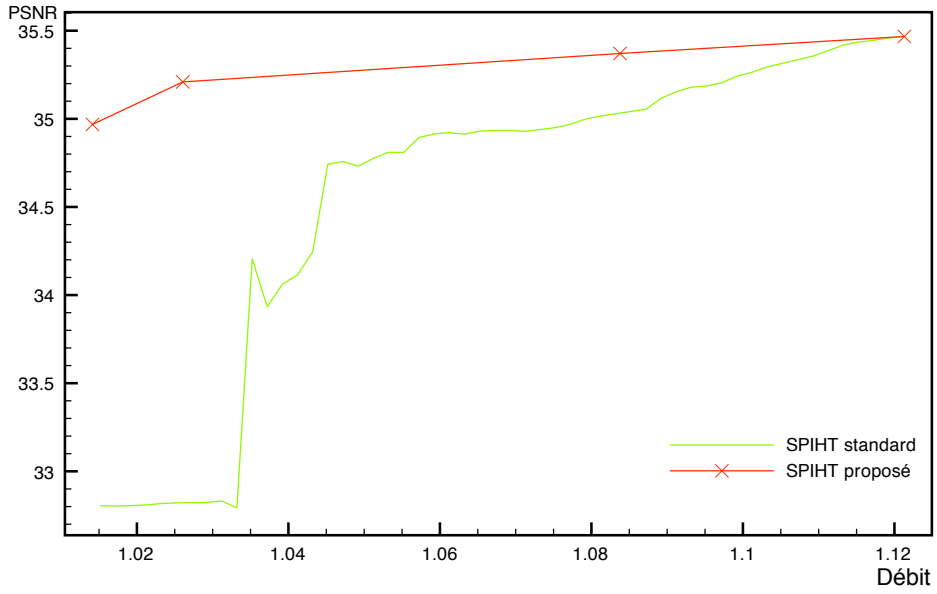



FIG. 4.17 – Comparaison entre SPIHT standard et SPIHT proposé sur l'image "lena".

| **Image** | **SPIHT standard** | | | **SPIHT proposé** | | |
|---|---|---|---|---|---|---|
| | **débit (bpp)** | **codage (s)** | **décodage (s)** | **débit (bpp)** | **codage (s)** | **décodage (s)** |
| baboon | 2.24 | 0.74 | 0.30 | 2.23 | 0.73 | 0.31 |
| barbara | 1.50 | 0.63 | 0.19 | 1.44 | 0.59 | 0.19 |
| boats | 1.21 | 0.69 | 0.14 | 1.12 | 0.63 | 0.14 |
| farm | 1.44 | 0.63 | 0.19 | 1.35 | 0.59 | 0.18 |
| lena | 1.12 | 0.68 | 0.14 | 1.03 | 0.63 | 0.13 |
| **MOYENNE** | 1.5 | 0,674 | 0.19 | 1,43 | 0.63 | 0.19 |
| **GAIN MOYEN** | | | | 4.7% | 6.5% | 0% |

TAB. 4.3 – Comparaison du SPIHT standard avec la version que nous proposons.

Pour le codage de l'intégralité de l'image, le gain moyen constaté est de l'ordre de 4.7%. De plus, vu que l'opération de codage va manipuler des listes d'éléments plus courtes, la durée d'exécution est réduite, de l'ordre 6.5%.

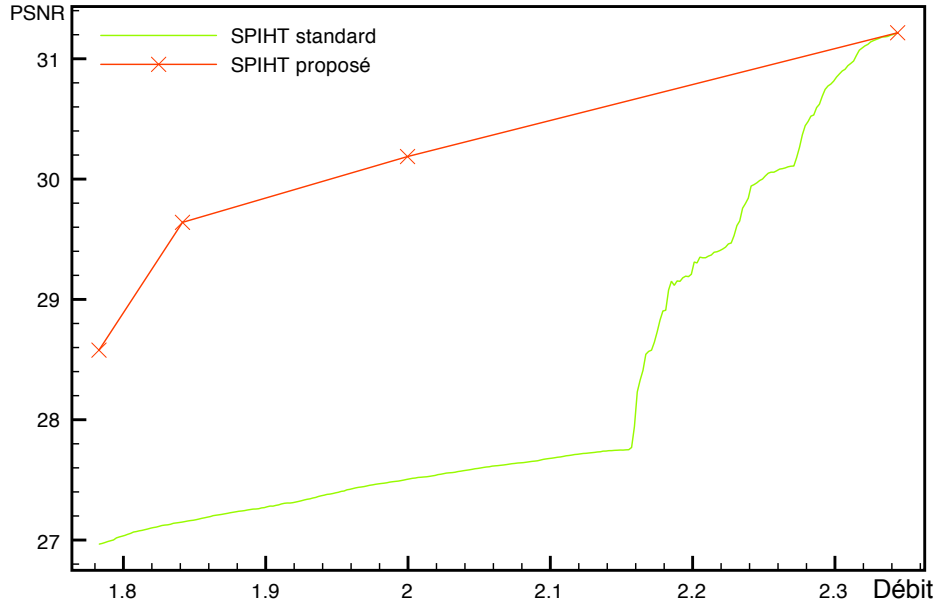



FIG. 4.18 – Comparaison entre SPIHT standard et SPIHT proposé sur l'image "yosemite".

# Chapitre 5

# Conclusion et perspectives

**Sommaire**



## 5.1 Résumé du travail effectué

–>allouer dynamiquement le pas de quantif en fonction du contenu local de l'image

## 5.2 Conclusion

## 5.3 Perspectives de recherche

–>quadtree trop contraignant ? pas adapter à la segmentation de zones quelconques ? –>passage à la couleur à long terme –>beaucoup d'inconnus encore malgré tous nos efforts

# Bibliographie


[Barba et Callet, 2003] Barba, D. et Callet, P. L. (2003). A robust quality metric for color image quality assessment. *Proceedings of the IEEE International Conference on Image Processing*, pages 437–440.

[Canny, 1986] Canny, J. (1986). A computational approach to edge detection. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 8(679-698).

[Gaubatz *et al.*, 2006] Gaubatz, M., Kwan, S., Chern, B., Chandler, D. et Hemami, S. S. (2006). Spatially-adaptive wavelet image compression via structural masking. *ICIP06, International Conference on Image Processing*, pages 1897–1900.

[Howard et Vitter, 1992] Howard, P. G. et Vitter, J. S. (1992). Analysis of arithmetic coding for data compression. *Information Processing and Management*, 26(6):749–763.

[Jeannic *et al.*, 2007] Jeannic, G., Ricordel, V. et Barba, D. (2007). The edge driven oriented wavelet transform : an anisotropic multidirectional representation with oriented lifting scheme. *Visual Communications and Image Processing (IS&T/SPIE Electronic Imaging), Coden PSISDG, doi 10.1117/12.704426*, 6508.

[Saadane, 2000] Saadane, A. (2000). Technologie et normes : Jpeg2000. *Support de cours, École Polytech'Nantes*.

[Said, 2004] Said, A. (2004). *Introducing to Arithmetic Coding - Theory and Practice*. HPL-2004-76. Imaging Systems Laboratory, HP Laboratories Palo Alto, published as a chapter in lossless compression handbook by khalid sayood édition.

[Said et Pearlman, 1996] Said, A. et Pearlman, W. A. (1996). A new fast and efficient image codec based on set partitionning in hierarchical trees. *IEEE Transactions on Circuits and Systems for Video Technology*, 6:243–250.

[Shapiro, 1993] Shapiro, J. M. (1993). Embedded image coding using zerotrees of wavelet coefficients. *IEEE Transactions on Signal Processing, ISSN : 1053-587X, CODEN : ITPRED, doi : 10.1109/78.258085*, 14(12):3445–3462.

[Taubman, 1998] Taubman (1998). Ebcot : Embedded block coding with optimized truncation. *IEEE transactions on image processing ISSN 1057-7149*, 9(7):1158–1170.

[Zeng *et al.*, 2000] Zeng, W., Daly, S. et Lei, S. (2000). Point-wise extended visual masking for jpeg 2000 image compression. *Proceedings of the IEEE International Conference on Image Processing, doi : 10.1109/ICIP.2000.901044*, 1:657–660.

[Zeng *et al.*, 2002] Zeng, W., Daly, S. et Lei, S. (2002). An overview of the visual optimization tools in jpeg 2000. *Signal Processing : Image Communications*, 17(1): 85–104.

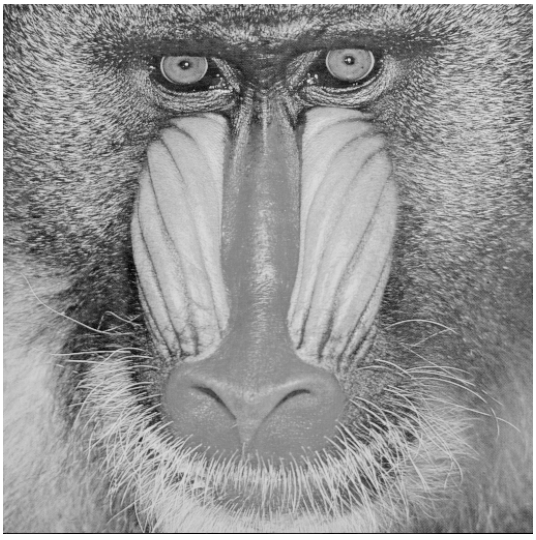
# Table des figures

# Liste des tableaux

# Liste des Algorithmes

# Chapitre 6

# Annexes



(a) baboon

(b) barbara

(c) boats

(d) farm

FIG. 6.1 – Première collection des images citées.

(a) lena

(b) newyork

(c) train

(d) yosemite

FIG. 6.2 – Deuxième collection des images citées.