Transformées orientées par blocs pour le codage vidéo hybride

Antoine Robert

ECOLE DOCTORALE
D'INFORMATIQUE
TELECOMMUNICATIONS
ET ELECTRONIQUE
DE PARIS

Thèse

pour obtenir le grade de Docteur de
l'École Nationale Supérieure de Télécommunications
Spécialité : Signal et Images

Antoine ROBERT

TRANSFORMÉES ORIENTÉES PAR BLOCS POUR LE CODAGE VIDÉO HYBRIDE

Soutenue le 18 février 2008 devant le jury composé de :

Joseph Ronsin	**Président du jury**
Amel Benazza	**Rapporteurs**
James Fowler	
Stéphane Pateux	**Examinateurs**
Béatrice Pesquet-Popescu	**Directrices de thèse**
Isabelle Amonou	

Remerciements

Ce document présente mes travaux de thèse réalisés à France Télécom R&D (site de Rennes) dans le laboratoire Image, Richmedia, nouvelles Interactions et hyperlangageS (IRIS) au sein de l'équipe Compression Vidéo Avancé (CVA).
Ce travail est issu d'une collaboration avec le laboratoire de Traitement du Signal et de l'Image (TSI) de l'École Nationale Supérieure de Télécommunications (ENST) de Paris.

Je voudrais d'abord remercier Vincent Marcatté et Alexandre Nolle pour m'avoir admis au sein du laboratoire IRIS. Je remercie sincèrement Henri Sanson et Ludovic Noblet pour m'avoir accueilli dans l'équipe CVA et pour avoir pris le temps de suivre mes travaux.

J'exprime toute ma gratitude à Béatrice Pesquet-Popescu, professeur de l'ENST Paris, qui a accepté d'être ma directrice de thèse sur ce sujet d'étude. Merci d'avoir guidé et régulièrement évalué la pertinence de mes recherches.

Je remercie tout particulièrement Isabelle Amonou, ingénieur de recherche à France Télécom R&D Rennes, pour m'avoir encadré et dirigé tout au long de cette thèse, pour sa disponibilité et la confiance qu'elle m'a accordé.
Je remercie Stéphane Pateux, ingénieur de recherche à France Télécom R&D Rennes, pour tous les conseils qu'il a pu me donner et pour les discussions scientifiques que nous avons eues.
Je voudrais aussi exprimer mes plus sincères remerciements à tous les membres de l'équipe CVA, pour leur disponibilité, leur gentillesse et tous les conseils judicieux qu'ils ont pu me donner.

Je remercie Amel Benazza, professeur de l'École Supérieure des Communications (SUP'COM) de Tunis, et James Fowler, professeur du département "Electrical & Computer Engineering" de l'université du Mississippi, d'avoir bien voulu accepter la charge de rapporteur.
Je remercie Joseph Ronsin, professeur à l'Institut National des Sciences Appliquées (INSA) de Rennes, qui m'a fait l'honneur d'examiner cette thèse.
Je remercie également Stéphane Pateux, ingénieur de recherche à France Télécom R&D Rennes, d'avoir accepté de juger ce travail.

Enfin, un merci très spécial à mon amie qui m'a encouragé et fortement soutenu tout au long de ces trois années et à notre fils qui nous a donné beaucoup depuis son arrivée.

Table des matières

Table des figures

Liste des tableaux

Glossaire

AFX	*Animation Framework eXtension*
ASO	*Arbitrary Slice Ordering*
AMR	Analyse MultiRésolution
AVC	*Advanced Video Coding*
AWT	*Anisotropic Wavelet Transform*
BinDCT	*Binary Discrete Cosine Transform*
CABAC	*Context-based Adaptive Binary Arithmetic Coding*
CAVLC	*Context-based Adaptive Variable Length Coding*
CBP	*Coded Block Pattern*
CIF	*Common Intermediate Format*
DCT	*Discrete Cosine Transform*
DFB	*Directional Filter Bank*
DWT	*Discrete Wavelet Transform*
EZBC	*Embedded Zero-Block Coding*
FMO	*Flexible Macroblock Ordering*
f.p.s.	frames par seconde (*frames per second*)
FRExt	*Fidelity Range Extensions*
GOP	*Group Of Pictures*
IDR	*Instantaneous Decoding Refresh*
IntDCT	*Integer Discrete Cosine Transform*
IPMP	*Intellectual Property Management and Protection*
ISO	*International Organisation for Standardization*
ITU	*International Telecommunications Union*
JPEG	*Joint Photographic Experts Group*
JVT	*Joint Video Team*
LOT	*Lapped Orthogonal Transform*
LBT	*Lapped Biorthogonal Transform*
LPS	*Least Probable Symbol*
MBAFF	*Macroblock Adaptive Frame/Field*
MCTF	*Motion-Compensated Temporal Filtering*
ME	*Motion Estimation*
MPEG	*Motion Picture Experts Group*
MPS	*Most Probable Symbol*
NAL	*Network Abstraction Layer*

NALU	*Network Abstraction Layer Unit*
PDFB	*Pyramidal Directional Filter Bank*
PSNR	*Peak Signal-to-Noise Ratio*
QCIF	*Quarter Common Intermediate Format*
QP	*Quantification Parameter*
RDO	*Rate Distortion Optimization*
RGB	*Red Green Blue*
SAD	*Sum of Absolute Differences*
SA-DCT	*Shape-Adaptive Discrete Cosine Transform*
SEI	*Supplemental Enhancement Information*
SD	*Standard Definition*
SVC	*Scalable Video Coding*
TF	Transformée de Fourier
TOC	Transformée en Ondelettes Continues
TOD	Transformée en Ondelettes Discrètes
VCEG	*Video Coding Experts Group*
VGA	*Video Graphics Array*
VLC	*Variable Length Coding*
WHT	*Walsh-Hadamard Transform*
YCrCb	Luminance et Chrominances rouge et bleue

Première partie

Introduction générale

Contexte et enjeux

Aujourd'hui, la vidéo numérique permet d'offrir aux utilisateurs un très grand nombre de services tels que la vidéo sur réseaux (DSL, terrestre, mobile...), les DVD, le broadcasting, la vidéo à la demande, etc. Ces applications ont nécessité le développement d'outils de codage efficaces et rapides afin de compresser au maximum les flux vidéo tout en gardant une qualité visuelle optimale.

A l'heure actuelle, l'état de l'art en codage vidéo est représenté par la norme H.264 MPEG-4/AVC ("Advanced Video Coding") qui offre des niveaux de performance supérieurs aux autres schémas classiques proposés.
En effet, bien que ce codeur soit toujours fortement basé sur une architecture de codage hybride et prédictif, certains outils permettent d'améliorer l'efficacité de codage : la compensation en mouvement sur des blocs de taille variable avec une précision au quart de pixel et des images de références multiples, l'amélioration des modes "skipped" et "direct", une prédiction spatiale directionnelle pour le codage Intra, un filtre réducteur d'effets de blocs dans la boucle... En outre, après ce codeur a aussi été amélioré en incluant, par exemple, une taille de transformée 4×4, un codage hiérarchique des blocs ou le codage arithmétique adaptatif contextuel.

Cependant, il existe par ailleurs des techniques de codage qui peuvent être plus efficaces pour certains types de contenus. Ainsi, l'utilisation de transformations en ondelettes, traditionnelles ou de seconde génération, ou bien encore de techniques orientées ou à recouvrement, permet d'obtenir des gains significatifs. Toutefois, ces approches manquent souvent de généricité, et peuvent devenir très largement inefficaces (voire impossibles à mettre en œuvre) sur des contenus non adaptés.

Ce travail a pour but de proposer un nouveau schéma efficace de codage hybride cherchant à mettre à profit les bénéfices apportés par différentes approches de codage. On cherchera ainsi à identifier des zones particulières d'un contenu vidéo et à proposer la technique de codage la plus adaptée pour chacune de ces zones. Pour ce faire, une étude sur les différents schémas de codage vidéo existants sera menée où l'on s'attachera à identifier les types de contenus bien adaptés et les performances attendues. Puis par la suite, on s'intéressera au problème d'hybrider de façon efficace des techniques de compression différentes.

Problématique

Cette étude s'inscrit dans le domaine de la compression vidéo portée par la norme H.264 MPEG-4/AVC et consiste en l'hybridation de cette norme avec plusieurs méthodes existantes sélectionnées en fonction du contexte.

Dans un premier temps, il nous a semblé nécessaire de faire quelques rappels sur la compression vidéo. Pour cela, le chapitre 1 introduit le codage vidéo en général avec la problématique de la compression et les données d'entrée avant de proposer un schéma général. Ensuite, une

présentation des codeurs vidéo hybrides est réalisée selon leurs quatres principaux modules (prédiction, transformation, quantification et codage entropique). L'exemple de H.264/AVC permet alors d'illustrer ces codeurs hybrides. Enfin, les autres types de codeurs vidéo, comme les codeurs basés ondelettes, sont introduits succinctement.

Un module important du codage vidéo est la transformation qui peut être selon les codeurs une transformée en ondelettes, une DCT, une DCT en étages lifting, ou une transformée à recouvrement. Le chapitre 2 décrit ces différentes transformées en illustrant leurs propriétés et en évaluant leurs performances.

Cependant, aucune de ces transformées ne permet de bien représenter les contours des objets de manière efficace pour la compression (elles sont redondantes ou elles ne décorrèlent pas les informations). Afin d'améliorer ce point, plusieurs méthodes ont été proposées, certaines basées sur les ondelettes et d'autres sur la DCT. Le chapitre 3 détaille les ondelettes dites de seconde génération telles que les *curvelets*, les *contourlets*, les bandelettes et les *directionlets*, et les DCT exploitant l'orientation comme la SA-DCT, la BinDCT orientée et la DCT directionnelle.

Aucune des transformées, présentées dans ces chapitres 2 et 3, ne permet d'obtenir de très bon résultats sur toutes les images et de manière relativement simple. Il nous a donc semblé intéressant de rechercher de nouvelles méthodes qui permettent de traiter efficacement les contours. Comme le codeur de l'état de l'art est H.264/AVC, il nous a paru être le plus approprié pour définir ces nouvelles méthodes.

Contributions

Le codeur hybride H.264/AVC prédit toutes les images qu'il traite (spatialement ou temporellement). Les images résultantes sont alors résiduelles, mais possèdent toujours des contours. Afin d'exploiter cette observation, nous avons recherché à adapter la transformation de ce codeur.

L'état de l'art le plus ancien sur cette adaptation aux contours est porté par les ondelettes de seconde génération. Nous avons donc cherché à voir la DCT H.264/AVC comme une transformée en ondelettes en définissant, dans la section 4.1, un schéma de celle-ci sous forme lifting permettant ainsi de lui appliquer des outils de seconde génération.
Les transformées à recouvrement affichent de très bons résultats en codage d'images naturelles. Nous avons alors essayé, dans la section 4.2, d'en introduire une version en pré- et post-traitements dans le codeur H.264/AVC.

Les méthodes précédentes cherchent à adapter la transformée en fonction du signal traité, mais le problème peut être vu différemment en cherchant plutôt à adapter le signal à la transformée qui est utilisée. Dans ce but, le chapitre 5 présente une méthode d'orientation par pré- et post-traitements des images résiduelles H.264/AVC qui en améliore la compression. Ces traitements réalisent des pseudo-rotations locales par cisaillements permettant des redressements locaux vers l'horizontale ou la verticale. Ces pseudo-rotations s'appliquent à l'échelle du macrobloc ou de ses partitions et sont définies par l'orientation ou la forme de la partition traitée.
De plus, une caractéristique de la DCT H.264/AVC utilisée nous a permis en section 5.5 d'adapter

le parcours des coefficients quantifiés issus de cette méthode d'orientation par pré- et post-traitements. Cette adaptation est alors fonction du redressement appliqué à la partition ou de sa forme et permet, d'après les résultats, de diminuer le débit global de la méthode d'orientation proposée.

Deuxième partie

État de l'art

Chapitre 1

Le codage vidéo

1.1 Introduction au codage vidéo

1.1.1 Problématique du codage vidéo

Une vidéo numérique est une séquence d'images numériques permettant de représenter une grande quantité d'informations, informations qu'il est nécessaire de stocker et/ou de transmettre.

ex : film de 90 minutes en format TV (576 lignes de 720 points à 25 images par seconde et 16 bits de couleurs)

720*576 * 25 * 16 * 90*60 = 896 Gbits

Par cet exemple, on voit que le volume représenté par les informations est beaucoup trop important pour le stockage et/ou la transmission d'où la nécessité de compresser les données.
Il faut définir des normes de codage vidéo pour que ces données compressées puissent être décompressées par tout le monde en vue d'une visualisation.

1.1.2 Les données d'entrée

Caractéristiques des données d'entrée

Quel que soit le codeur vidéo utilisé pour réaliser le codage, une bonne connaissance des paramètres d'entrée est impérative. En effet, le codage va prendre en compte les propriétés des données d'entrée liées à l'acquisition de la séquence vidéo.
Ces propriétés sont leur taille, leur fréquence, leur mode et leur couleur.

La taille des images d'entrée peut être quelconque ou standard. Cette taille permet de définir le ratio de l'image (4/3, 16/9). Quelques tailles standardisées sont :

- QCIF = 176 × 144 pixels.
- CIF = 352 × 288 pixels (CIF : Common Intermediate Format).
- 4CIF = 704 × 576 pixels.
- VGA = 640 × 480 pixels.
- TV 4/3 = 768 × 576 pixels.

La fréquence des images d'entrée correspond au nombre d'images que la séquence contient dans une seconde. Elle peut être de 25 - 30 - 50 - 60 images/seconde (ou frames/seconde f.p.s.).

Ces images peuvent être de deux modes, progressif ou entrelacé (cf fig. 1.1).

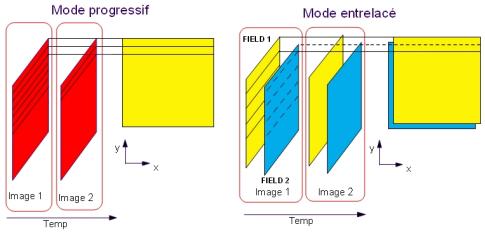


FIG. 1.1 – Images en mode progressif et entrelacé

En mode entrelacé, les images (frames) sont composées de deux champs (fields), un pour les lignes paires et un pour les lignes impaires.
Dans le cas progressif, chaque image n'est composée que d'un champ contenant toutes les lignes.

Changement d'espace colorimétrique

Les images d'entrée peuvent être en noir et blanc, en couleur RGB (Rouge Vert Bleu) sur 8 bits, 16 bits,... Dans le cas des images couleur, une conversion de système colorimétrique est généralement effectuée par :

$$\begin{aligned} Y &= 0.299R + 0.587G + 0.114B \\ Cr &= R - Y \\ Cb &= B - Y \end{aligned} \tag{1.1}$$

Cette conversion a pour but de passer dans le domaine luminance Y - chrominances Cb, Cr.

L'œil humain étant moins sensible aux variations de chrominances que de luminances, il est possible de sous-échantillonner les informations de chrominances. Différents modes sont alors possibles :

- 4 : 4 : 4 pas de sous-échantillonnage des chrominances.
- 4 : 2 : 2 les chrominances ont le même nombre de lignes que la luminance, mais deux fois moins de colonnes.
- 4 : 2 : 0 les chrominances ont deux fois moins de lignes et de colonnes que la luminance.

1.1.3 Schéma de codage vidéo

Le but du codage vidéo est de compresser les données d'une séquence vidéo composée d'une succession d'images naturelles d'une certaine taille et à une certaine fréquence.

Selon ce principe, tout codeur vidéo peut être représenté selon le schéma général suivant :

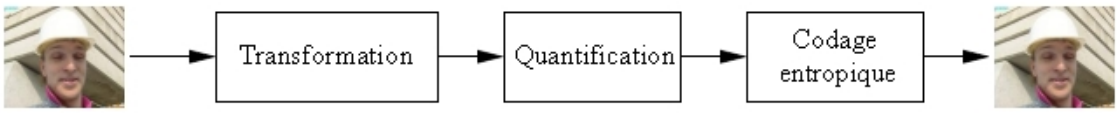


FIG. 1.2 – Schéma général d'un codeur vidéo

Un tel codeur permet de compresser la séquence vidéo comme une suite d'images fixes sans exploiter le mouvement (par exemple MotionJPEG [ISO94a]).

Les principales étapes de ce schéma de codage sont la transformation, la quantification et le codage entropique.

La transformation : cette étape correspond à une transformation mathématique qui permet de décorréler les informations des images. Il s'agit généralement d'une transformation fréquentielle. Cette étape est réversible quand elle est basée sur des fonctions mathématiques bijectives (ce qui est généralement le cas).

La quantification : c'est une étape non réversible, c'est-à-dire qui entraîne des pertes. Cependant cette étape permet d'obtenir le débit souhaité puisqu'elle correspond à une réduction de l'étalement des amplitudes des coefficients transformés.

Le codage entropique : cette étape permet de réaliser le codage à proprement parler des informations. Il consiste à associer à chaque symbole, valeur possible des coefficients transformés et quantifiés, un mot de code dont la longueur dépend de la fréquence d'apparition de ce symbole sur l'ensemble des coefficients.

Afin d'améliorer les performances de ce schéma de codage de base, plusieurs types de codeur ont été mis au point en ajoutant notamment une compensation en mouvement ou une boucle fermée. Le plus utilisé de ces schémas est le schéma hybride présenté dans la section suivante 1.2 qui servira de base à notre travail, cependant, il en existe d'autres types qui seront présentés ultérieurement dans la section 1.3.

1.2 Les principes du codage vidéo hybride

Le schéma de codage vidéo de type hybride est le plus courant puisqu'il est utilisé dans toutes les normes de codage vidéo (MPEG-x, H.26x). Il est appelé hybride parce que c'est un schéma en boucle fermée qui utilise les informations déjà codées/décodées pour réaliser le codage de la partie courante à l'aide de différents modules.

De plus, ce schéma est toujours de la même forme (cf fig. 1.3), seuls les modules qui le composent diffèrent d'un codeur à l'autre.

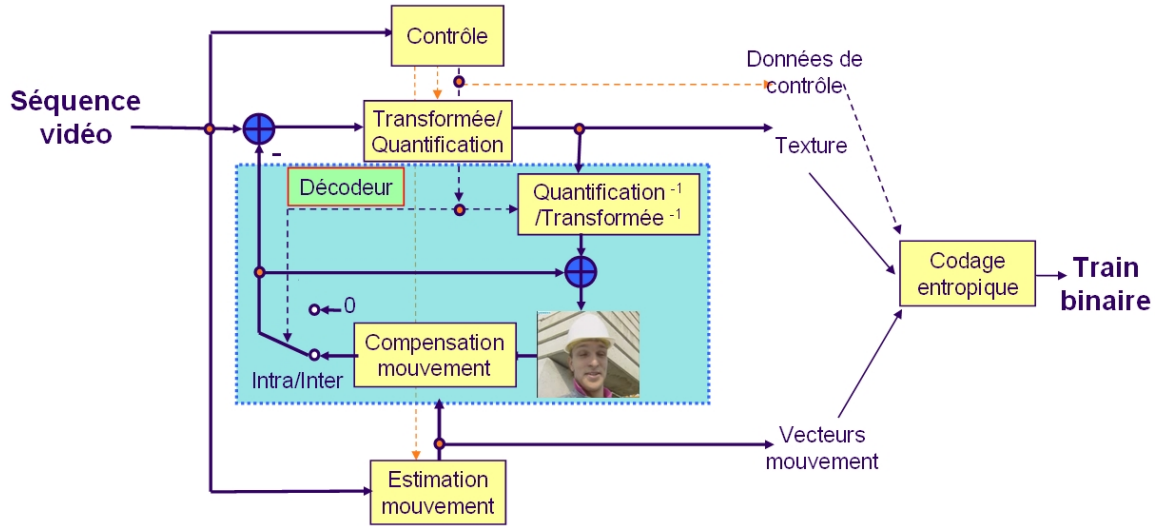


FIG. 1.3 – Schéma général d'un codeur vidéo

Les quatre modules importants de ce schéma sont : l'estimation et la compensation en mouvement, la transformation, la quantification et le codage entropique.

Nous allons, par la suite, présenter ces quatre modules plus en détails, en nous basant sur la norme MPEG-2 [ISO94b].

1.2.1 L'estimation et la compensation en mouvement

Le principe de la compression est d'exploiter les redondances présentes naturellement dans le contenu. En compression vidéo, l'estimation et la compensation en mouvement permettent de réduire la redondance temporelle en utilisant des données passées et/ou futures pour prédire l'image courante.

La première opération réalisée sur la séquence vidéo à coder est de découper chacune des images la composant en macroblocs 16 pixels × 16 pixels (sauf pour H.264/AVC). Ces macroblocs sont alors l'unité de compensation en mouvement.

Il existe trois sortes d'images compensées : les images I dites Intra, les images P dites Inter prédites et les images B dites Inter prédites bidirectionnelles.

Les images Intra

Pour les images I, il n'y a ni estimation ni compensation en mouvement. En effet, ces images permettent un accès aléatoire dans la séquence et sont donc codées directement sans compensation. Ces images sont transformées, quantifiées et codées par codage entropique comme une image fixe JPEG. Elles sont donc décodables indépendamment des autres images assurant ainsi l'accès aléatoire, des points de reprise, une limitation de la propagation d'erreurs,...

Les images Inter prédites

Pour les images P, on suppose que tous les pixels de l'image courante sont prédictibles par translation de ceux d'une image précédemment codée/décodée.

Pour cela et pour chacun des macroblocs de l'image courante, on recherche dans l'image codée/décodée précédemment (I ou P) le macrobloc qui le caractérise le mieux en minimisant une fonction de coût comme le SAD (*sum of absolute differences*). On extrait alors le vecteur permettant de translater le macrobloc candidat vers le macrobloc courant (cf fig. 1.4). Deux paramètres sont donc à définir pour réaliser cette estimation de mouvement : une fenêtre de recherche limitant le nombre de macroblocs candidats doit être initialement explicitée pour diminuer la complexité de cette estimation, et une précision pixellique ou sous-pixellique doit être spécifiée pour permettre de caractériser la précision de cette estimation.

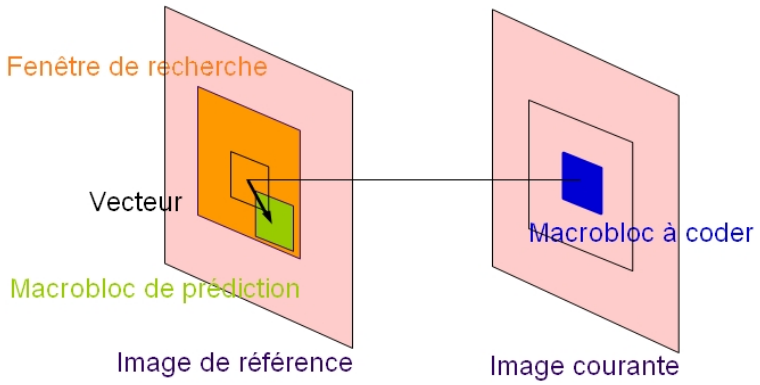


FIG. 1.4 – Estimation de mouvement pour les images prédites

Après cette estimation de mouvement, tous les macroblocs de l'image courante peuvent être représentés par ceux de l'image précédente (I ou P) de référence. Une image est générée, c'est l'image de référence compensée en mouvement à l'aide des vecteurs de translation déterminés précédemment.

L'image P est la différence entre l'image courante à prédire et l'image de référence (I ou P) compensée en mouvement (cf fig. 1.5).

Les images Inter bidirectionnelles

Pour les images B, on suppose, comme pour les images P, que tous les pixels de l'image courante sont prédictibles par translation de ceux d'images de référence.

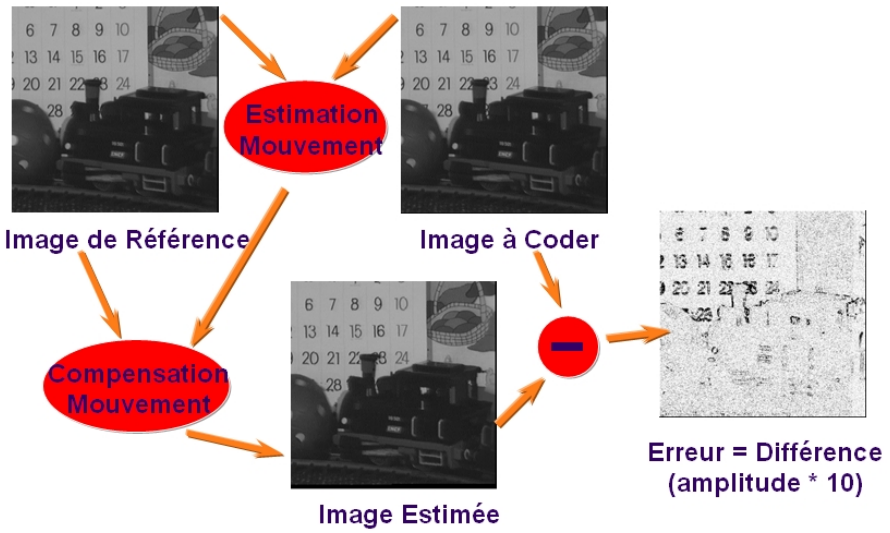


FIG. 1.5 – Compensation en mouvement pour les images prédites

Pour les images P, les images I et P passées sont utilisées comme images de référence. Il s'agit d'une compensation "forward predictive" qui se réfère à une image antérieure. Pour les images B, les images I et P passées ou futures peuvent être utilisées pour l'estimation du mouvement. Il s'agit ici d'une compensation "bidirectionally predictive" qui se réfère à des images antérieures et futures (dans l'ordre d'affichage sinon ce sont des images déjà codées/décodées).

La même méthode d'estimation et de compensation en mouvement que pour les images P est utilisée pour les images B, mais les références sont multiples : une image passée et une image future. Deux vecteurs de mouvement sont alors utilisés, on parle d'interpolation bidirectionnelle (cf fig. 1.6).

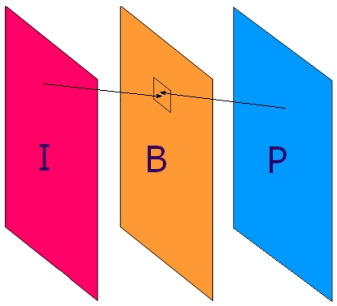


FIG. 1.6 – Compensation en mouvement pour les images bidirectionnelles

L'organisation des données

Comme on l'a vu précédemment, il existe deux types d'images, les Intra et les Inter. Les images Intra ne sont pas compensées en mouvement contrairement aux images Inter. Cette compensation en mouvement des images Inter peut être de deux natures : P prédite à partir d'une image

passée, ou B interpolée à partir d'une image passée et d'une future. Leurs ordres de codage et de transmission ne vont donc pas être les mêmes que l'ordre d'affichage.

Les images I sont codées indépendamment des autres images, elles sont donc codées d'abord. Les images P utilisent une seule image de référence antérieure, alors que les images B utilisent deux images de référence, une passée et une future. Pour ces images Inter, les images de référence sont choisies parmi les images I et les images P déjà codées/décodées. Les images P sont donc codées après les images I et avant les images B (cf fig. 1.7)

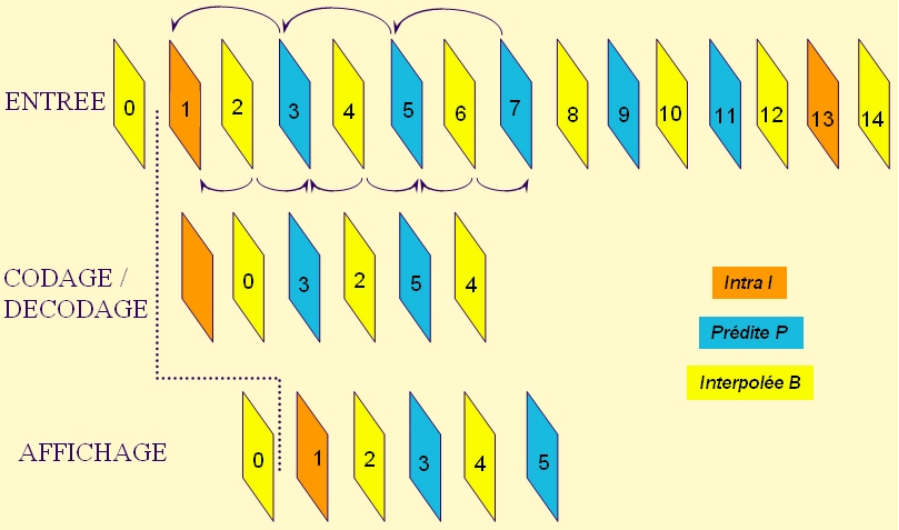


FIG. 1.7 – Ordre de codage des images et leur ordre d'affichage

1.2.2 La transformation

La transformation a pour but d'exploiter les redondances spatiales contenues naturellement dans les images. Pour cela, on utilise une transformation fréquentielle qui permet de concentrer l'information dans les basses fréquences (comme la transformée de Fourier, par exemple).

Après l'estimation et la compensation en mouvement (cf fig. 1.3), l'image prédite est soustraite à l'image courante et ce uniquement pour les images Inter, les images Intra I étant codées directement sans compensation en mouvement. La transformation s'applique donc aux images I et aux images résiduelles Inter.

L'unité de compensation en mouvement qui était le macrobloc 16×16 n'est plus l'unité de codage pour la transformation. On redécoupe donc chacune des images (I ou résiduelles) en blocs 8 pixels × 8 pixels, ces blocs 8×8 sont l'unité de codage pour la suite.

La transformation fréquentielle utilisée est la DCT (Discrete Cosine Transform ou Transfor-

mée en Cosinus Discrets) qui est définie par :

$$C_{m,n} = \alpha(m)\beta(n) \sum_{i=0}^{7} \sum_{j=0}^{7} B_{i,j} \cos\left(\frac{\pi(2i+1)m}{16}\right) \cos\left(\frac{\pi(2j+1)n}{16}\right) \tag{1.2}$$

$$\text{avec} \quad 0 \leq m,n \leq 7 \quad \text{et} \quad \alpha(m)\,(\text{resp.}\beta(n)) = \begin{cases} 1/\sqrt{32} & \text{si m=0 (resp. si n=0)} \\ 1/4 & \text{sinon} \end{cases}$$

$B_{i,j}$ désigne le pixel (i,j) du bloc 8×8 original, et $C_{m,n}$ représente le coefficient du bloc DCT 8×8 transformé.

Le premier élément de la transformée DCT est la valeur moyenne du bloc 8×8 original, alors que les derniers coefficients représentent les hautes fréquences liées aux détails fins du bloc (cf fig. 1.8).

100	100	100	100	100	100	100	100
100	100	100	100	100	100	100	80
100	100	100	100	100	100	80	80
100	100	100	100	100	80	80	80
100	100	100	100	80	80	80	80
100	100	100	80	80	80	80	80
100	100	100	80	80	80	80	80
100	80	80	80	80	80	80	80

Avant DCT

730	46	0	5	0	1	0	0
46	-10	-21	0	-4	0	-1	0
0	-21	10	12	0	2	0	-1
5	0	12	-10	-8	0	-1	0
0	-4	0	-8	10	5	0	1
1	0	2	0	5	-10	-3	0
0	-1	0	-1	0	-3	10	2
0	0	1	0	1	0	2	-10

Après DCT

FIG. 1.8 – Exemple de transformation DCT sur un bloc (arrondi au plus proche entier)

Cette transformation est inversible, on peut retrouver l'image originale à partir des coefficients DCT en utilisant la transformée inverse définie par :

$$B_{i,j} = \sum_{m=0}^{7} \sum_{n=0}^{7} C_{m,n}\, \alpha(m) \cos\left(\frac{\pi(2i+1)m}{16}\right) \beta(n) \cos\left(\frac{\pi(2j+1)n}{16}\right) \tag{1.3}$$

$$\text{avec} \quad 0 \leq i,j \leq 7$$

Bien qu'une reconstruction exacte puisse être théoriquement réalisée, il n'est en pratique pas possible d'utiliser une précision infinie pour les calculs. Les erreurs d'arrondis de la transformée DCT peuvent faire diverger le codeur et le décodeur.

Jusqu'ici, le schéma est totalement inversible, c'est-à-dire que l'on peut retrouver l'information originale sans perte (en négligeant les erreurs liées aux arrondis de la transformation DCT). On parle de codage réversible sans perte.

1.2.3 La quantification scalaire

L'étape de quantification présentée ici introduit des pertes dans le schéma de codage.

L'œil humain est plus sensible aux basses fréquences qu'aux hautes fréquences, il est donc possible de diminuer l'information contenue dans les hautes fréquences avec une pénalité minimale. De plus, de faibles variations entre les pixels de l'image traitée ne sont pas visibles, on peut donc les supprimer pour ne garder que les fortes variations. Ce sont ces opérations que réalise la quantification.

La quantification diminue la quantité d'information contenue dans les blocs DCT en augmentant le nombre de valeurs à 0. Elle réalise donc une transformation irréversible qui entraîne des pertes de précision pour la reconstruction des blocs.

Elle consiste à diviser chacun des coefficients DCT d'un bloc par un pas de quantification scalaire Q et de n'en conserver que la partie entière :

$$C^Q_{m,n} = \left\lfloor \frac{C_{m,n}}{Q} \right\rfloor \qquad \text{avec} \quad 0 \leq m, n \leq 7 \tag{1.4}$$

La valeur du pas de quantification peut être différente pour chaque macrobloc d'une image, et elle est différente pour les images Intra et Inter. On peut aussi utiliser une matrice de quantification qui associe à chaque coefficient son pas de quantification. Cela permet de donner plus d'importance aux basses fréquences qu'aux hautes fréquences.

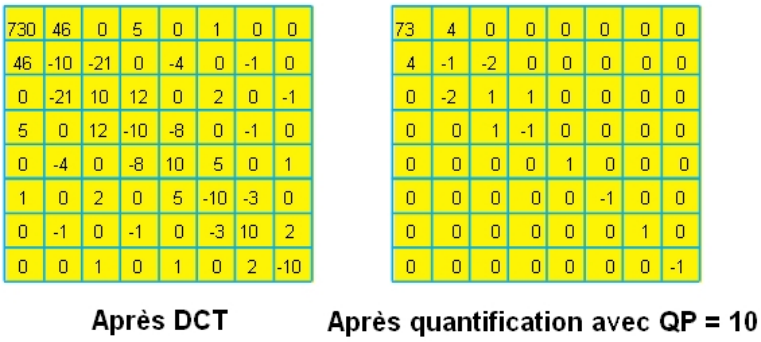


FIG. 1.9 – Exemple de quantification sur un bloc

On voit sur cet exemple que le nombre de coefficients non nuls a diminué, la quantification a bien réalisé la tâche souhaitée qui était de diminuer la quantité d'information à coder.

La quantification permet aussi de réguler le débit de sortie du codeur. En effet, plus on prendra un pas de quantification grand, plus on diminuera la quantité d'information à coder et donc le débit, mais cela influe sur la qualité de la vidéo. Il y a donc un compromis à trouver entre la qualité et le débit souhaité.

1.2.4 Le codage entropique

Une fois les blocs DCT quantifiés, il ne reste plus qu'à les coder avec un codage entropique afin de diminuer le débit de transmission. En effet, un tel codage exploite les propriétés statistiques des coefficients quantifiés en utilisant des mots courts pour représenter les événements les plus probables et des mots plus longs pour les occurrences rares.

Pour faciliter ce codage, les coefficients DCT quantifiés sont réorganisés. Les coefficients sont d'abord réordonnancés dans un vecteur unidimensionnel en les balayant en zigzag (cf fig. 1.10). Cet ordonnancement place le coefficient DC en premier dans le vecteur, et les coefficients restants AC sont arrangés des basses vers les hautes fréquences.

Ce vecteur réarrangé est ensuite décrit en événements $(last, run, level)$ où run décrit la distance entre deux coefficients non nuls du vecteur, $level$ est la valeur du coefficient non nul qui suit la plage de zéros, et $last$ indique si le coefficient est le dernier du bloc (cf fig. 1.10). Ceci permet de ne transmettre que les valeurs non nulles des coefficients DCT quantifiés.

Ces événements sont ensuite codés en utilisant un codage à longueur variable (VLC) qui associe des mots de code courts pour les valeurs courantes et des mots de code longs pour les valeurs à faible occurrence (cf fig. 1.10). Ces mots de code sont issus de tables connues du codeur et transmises au décodeur.

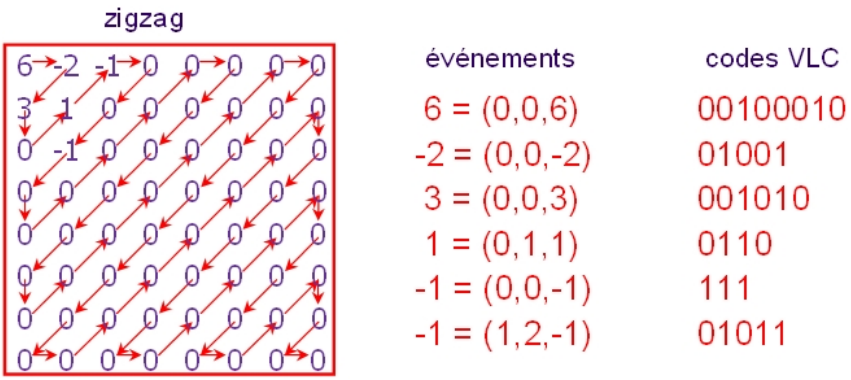


FIG. 1.10 – Exemple de codage entropique d'un bloc

Les autres informations telles que le type de prédiction (Intra ou Inter) et le pas de quantification peuvent également être codées au moyen de codes VLC.

1.2.5 Le train binaire

Lorsque tous les blocs des images (Intra ou Inter) ont été transformés, quantifiés et codés entropiquement, il faut les mettre en forme pour la transmission et/ou le stockage.

La normalisation ne définit pas le codeur mais le train binaire, c'est-à-dire l'organisation des données et leurs caractéristiques. Les données codées héritent leurs caractéristiques des techniques utilisées pour mettre en œuvre les quatre modules du codage présentés précédemment, c'est-à-dire les méthodes d'estimation-compensation en mouvement (précision, taille de fenêtre...), de transformation, de quantification et de codage entropique. L'organisation des données correspond à la structure des données pour la transmission et/ou le stockage.

Dans la plupart des normes, le train binaire est composé de plusieurs parties (cf fig. 1.11) comme la séquence, les images, les macroblocs, les blocs,...

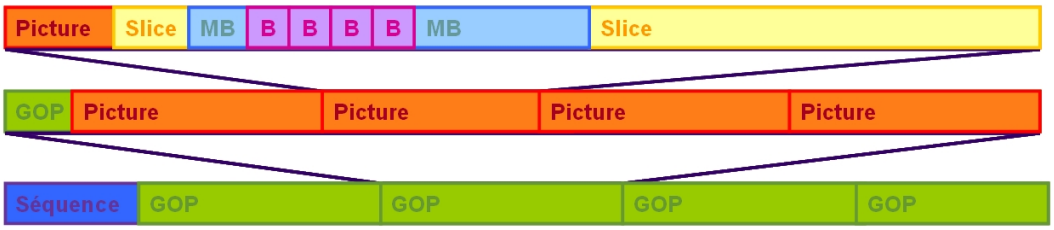


FIG. 1.11 – La structure du train binaire

La séquence avec son en-tête permet de définir le contexte et d'initialiser un décodeur. Cette séquence est divisée en GOP (Group Of Pictures) qui est l'unité d'accès aléatoire dans la séquence, chacun des GOP commencera donc avec une image I pour assurer cet accès aléatoire et le point de reprise. Chaque GOP contient un ensemble d'images qui est l'unité d'affichage, images qui sont décomposées en slices représentant des groupes de macroblocs et servant à la synchronisation. Ces slices sont eux composés de macroblocs 16×16 qui est l'unité d'estimation-compensation en mouvement, eux-mêmes formés de blocs 8×8, unité de codage DCT.

1.2.6 Un exemple : la norme H.264 MPEG-4/AVC

H.264 MPEG-4/AVC [JVT05] [JVT04] est une norme de codage vidéo mise au point très récemment par le JVT (Joint Video Team), équipe conjointe entre l'ISO (MPEG-x) et l'ITU (H.26x) deux grands acteurs de la normalisation video (cf Annexe A.1).

Cette norme vidéo suit le même schéma de codage que présenté précédemment (cf fig. 1.3). Cependant, elle apporte par rapport aux normes antérieures beaucoup de fonctionnalités supplémentaires aux différents modules du schéma, lui permettant de les surpasser (cf Annexe A.3).

Les principales modifications du schéma de codage affectent, comme illustré sur le schéma figure 1.12, les principaux modules du codage :

La prédiction :

- les images Intra subissent une prédiction spatiale [Ric03b] (cf Annexe A.3.2).
- l'estimation-compensation en mouvement est réalisée sur des blocs de tailles variables et avec une précision au quart de pixels [Ric03a] [Wie03] (cf Annexe A.3.3).

La transformation :

- elle s'applique sur des blocs de tailles 4×4 [Ric03d] ou 8×8 avec l'Amendement FREXt [JVT04] de cette norme (cf Annexe A.3.4).
- c'est une approximation entière de la DCT calculée matriciellement.

Le codage entropique peut être :

- un codage à longueur variable adaptatif au contexte CAVLC (Context-based Adaptive Variable Length Coding) [Ric02c] [WSBL03] [TR03] (cf Annexe A.3.6).
- un codage arithmétique binaire adaptatif au contexte CABAC (Context-based Adaptive Binary Arithmetic Coding) [Ric02a] [MSW03] [MSB+02] [MW03] (cf Annexe A.3.6).

Autres parties :

- les images de référence pour la prédiction Inter peuvent être multiples.
- un filtre a été ajouté dans la boucle pour diminuer les effets de blocs [Ric03c].

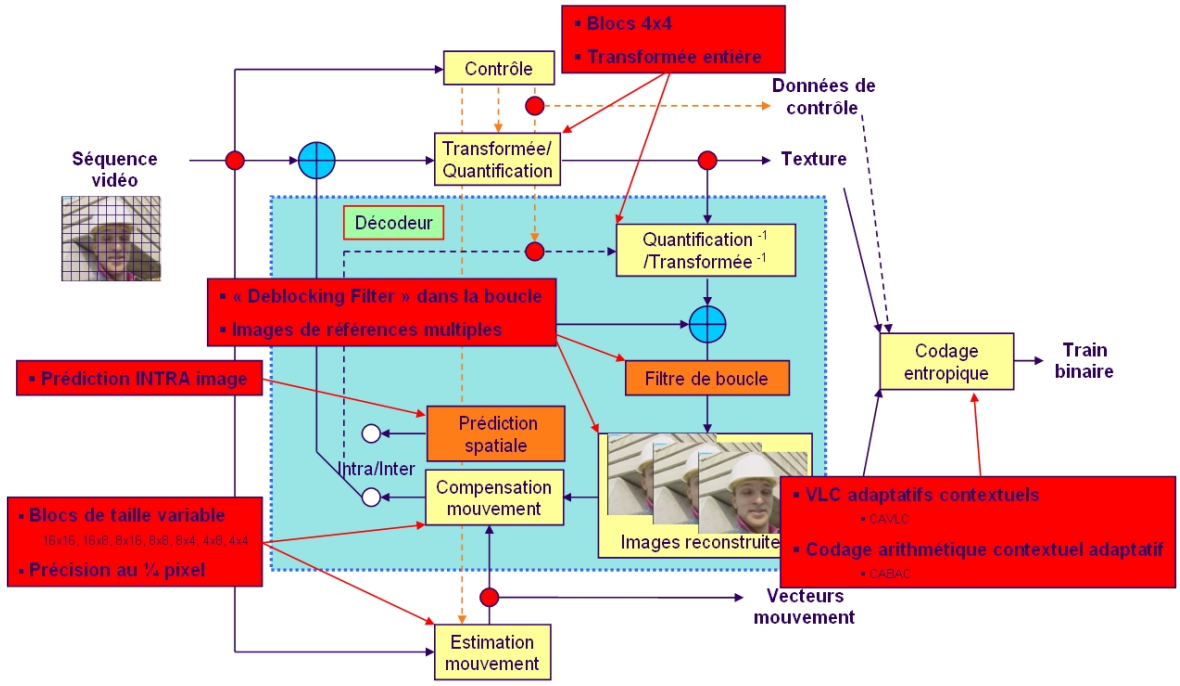


FIG. 1.12 – Schéma général du codeur vidéo H.264/AVC

Parmi ces fonctionnalités, seules certaines vont surtout nous intéresser dans ce travail : les prédictions Intra et Inter (cf Annexe A.3.2 et A.3.3), et essentiellement la transformation qui sera présenté plus en détail dans les chapitres suivants 2 et 4.1, et Annexe A.3.4.

1.3 Les autres types de codeurs vidéo

Nous avons présenté jusqu'ici les principes du codage vidéo hybride que nous utilisons dans notre travail (cf section 1.2 et Annexes A), mais il existe d'autres types de schémas de codage qui méritent d'être mentionnés, notamment des schémas en boucle ouverte basés ondelettes, par analyse-synthèse,...

1.3.1 Les codeurs en boucle ouverte

Les schémas de codage vidéo en boucle ouverte basés ondelettes utilisent généralement un filtrage temporel compensé en mouvement (MCTF "Motion-Compensated Temporal Filtering") couplé à une transformée en ondelettes 2D (cf section 2.2.5). On les appelle schémas "t+2D" puisqu'ils réalisent d'abord une décomposition temporelle avant d'effectuer la décomposition 2D spatiale classique.

Ils peuvent être représenté par :

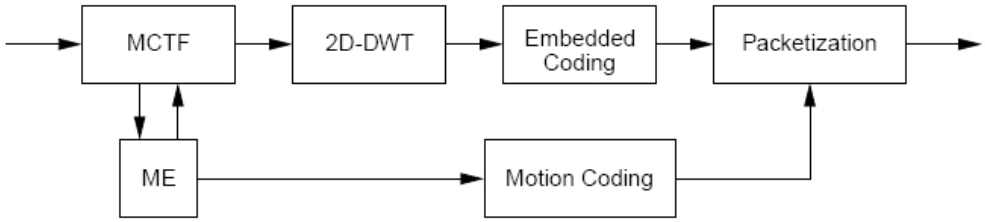


FIG. 1.13 – Schéma général du codeur en boucle ouverte avec ondelettes et MCTF

Cet MCTF correspond à une décomposition en ondelettes temporelles de la séquence vidéo avec l'aide d'une estimation de mouvement (ME). Cette décomposition temporelle peut être représentée selon :

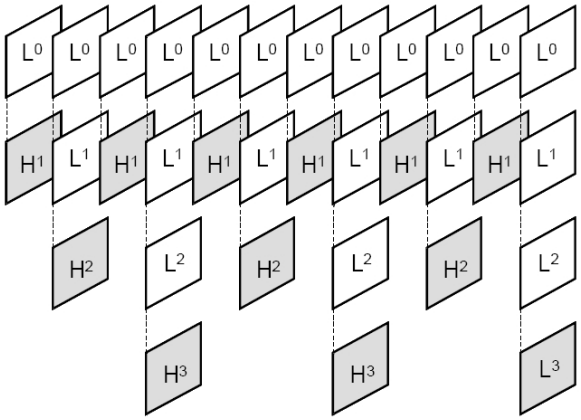


FIG. 1.14 – Décomposition temporelle de 12 images sur 3 niveaux (1/12, 1/4, 1/2) [SMW04]

Chacune des sous-bandes temporelles obtenues après le filtrage MCTF subit alors une transformation en ondelettes spatiale 2D (2D-DWT) que l'on étudiera par la suite dans la section 2.2. Cette transformation a pour but de décorréler les informations des sous-bandes temporelles en une imagette d'approximation (les basses fréquences) et plusieurs imagettes de détails (les hautes fréquences) selon :

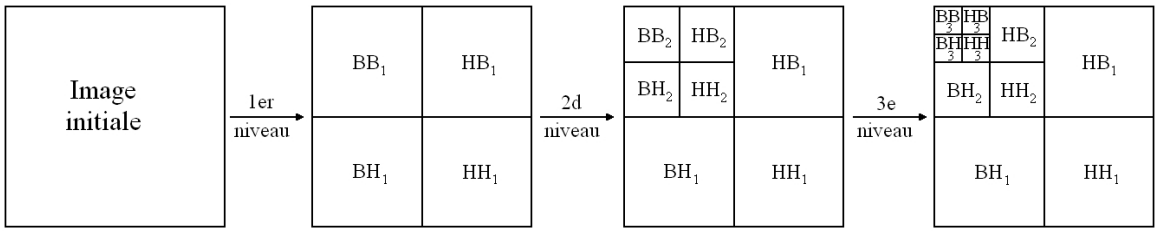


FIG. 1.15 – Décomposition spatiale en ondelettes d'une sous-bande temporelle sur 3 niveaux [RSC01]

Remarque : Ce type de schéma a alors naturellement de par sa construction les propriétés de scalabilité ou d'échelonnabilité temporelle et spatiale.

Les sous-bandes spatio-temporelles obtenues sont ensuite codées entropiquement, par exemple avec un codage emboîté par bloc de zéro (EZBC "Embedded Zero-Block Coding") [HW00]. Les données sont enfin généralement paquetisées avec les informations de mouvement codées entropiquement.

1.3.2 Les codeurs par analyse-synthèse

Les schémas par analyse-synthèse utilisent souvent, comme les schémas en boucle ouverte, des transformations en ondelettes [Cam04].

Dans ce type de schéma, l'étape d'analyse correspond à estimer le mouvement des objets de la séquence vidéo à l'aide de maillages 2D déformables et à décorréler les informations de mouvement et de texture, les textures sont redressées selon le mouvement estimé.

L'utilisation de maillages 2D est une alternative intéressante à l'estimation de mouvement par blocs (cf section 1.2.1) puisqu'ils permettent de fournir un champ dense, continu et inversible du mouvement, comme illustré sur la figure 1.16.

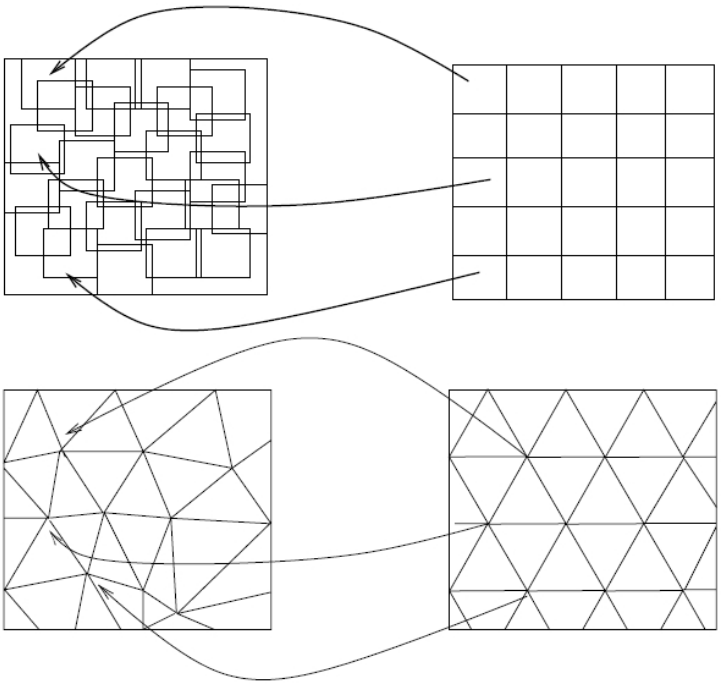


FIG. 1.16 – Comparaison entre l'estimation de mouvement par blocs et par maillages [Cam04]

Après l'analyse, les informations de mouvement et de texture redressée sont codées séparément et indépendamment à l'aide d'un codage efficace par ondelettes. Ce codage correspond à un schéma "t+2D" comme pour les codeurs en boucle ouverte. Un première transformation en ondelettes est appliquée temporellement le long de la trajectoire de mouvement. Et, une seconde est appliquée spatialement sur chacune des sous-bandes temporelles.

Remarque : La décomposition "t+2D" confère aussi à ce type de schéma des propriétés naturelles de scalabilité ou d'échelonnabilité temporelle et spatiale.

A la synthèse, la texture est plaquée sur le maillage qui est déformé selon les informations de mouvement afin de reconstruire les objets ou la séquence vidéo.

1.4 Conclusion

Ce premier chapitre a caractérisé généralement la compression vidéo avant de présenter plus en détails les codeurs basés ondelettes et les codeurs de type hybride qui vont surtout nous intéressé ici tel que H.264 MPEG-4/AVC.

Les schémas de type hybride tirent ce nom de l'utilisation d'une boucle fermée de décodage permettant de réaliser la ou les étapes de prédiction (temporelle et/ou spatiale).
Ces schémas se basent sur quatres principaux modules afin de réaliser la compression des données vidéo d'entrée :

- Une prédiction temporelle pour les images Inter, il n'y a pas de prédiction pour les images Intra, sauf dans le cas de la norme H.264/AVC où cette prédiction est spatiale. Cette prédiction temporelle Inter est réalisée par une estimation et une compensation en mouvement de l'image de référence (ou des images de référence pour la norme H.264/AVC).
- Une transformation fréquentielle qui permet de décorréler spatialement les informations en concentrant les énergies dans les basses fréquences. La transformation classique utilisée par ces codeurs hybrides est la transformée en cosinus discrets DCT. Elle s'applique généralement à des blocs de taille 8×8 et est calculée en nombres flottants qui nécessitent des arrondis entraînant une divergence entre le codeur et le décodeur. Dans le cas de la norme H.264/AVC, la transformation est différente, c'est une approximation de la DCT flottante calculée matriciellement en nombres entiers sur des blocs de taille 4×4 et/ou 8×8.
- Une quantification qui est la seule étape non réversible du schéma. Elle consiste à réduire l'amplitude des coefficients issus de la transformation DCT en divisant chacun d'eux par le pas de quantification choisit. Ce pas de quantification permet alors de réguler le débit de sortie en annulant plus ou moins ces coefficients DCT, c'est-à-dire en diminuant plus ou moins la quantité d'informations liée à ces coefficients DCT qui est à coder.
- Un codage entropique qui permet de représenter efficacement, soit en s'approchant au maximum de la borne entropique, les informations (c.à.d. les coefficients DCT quantifiés) par des symboles binaires. Ces symboles sont courts pour les informations de fortes occurrences et longs pour les faibles occurrences. Ils peuvent être à longueurs fixes ou variables ou être calculés arithmétiquement. De plus, dans tous ces cas, ils peuvent être adaptés en fonction du contexte afin d'être plus performants.

Les autres schémas présentés dans ce chapitre sont basés sur des approches en ondelettes.
Les schémas en boucle ouverte utilisent un filtrage temporel MCTF qui permet de réaliser une transformation en ondelettes de la séquence vidéo suivant l'axe temporel. Cette transformation correspond à effectuer une décomposition temporelle multirésolution de la séquence vidéo en s'appuyant sur son mouvement estimé.
Les schémas par analyse-synthèse utilisent le même type de filtrage temporel, mais appliqué à un maillage déformé par le mouvement détecté ainsi qu'aux textures associées.
Ces deux schémas réalisent ensuite une décomposition en ondelettes spatiale de chacune des sous-bandes de la décomposition temporelle. Les coefficients obtenus sont alors généralement codés à l'aide d'arbres de zéros (les coefficients nuls étant redondants dans les sous-bandes plus précises).

Ces schémas ondelettes sont tout aussi performants et intéressants que les schémas de type hybride, mais ils n'ont jamais été retenus pour des solutions normalisées qui vont plus nous intéresser dans ces travaux.

Chapitre 2

La transformation dans le codage vidéo

2.1 Présentation de l'étape de transformation

Le chapitre précédent (1) a présenté les quatres modules du codage vidéo, et a montré que la transformation tient une place importante dans ce codage. Dans la plupart des normes vidéo, il s'agit d'une transformée en cosinus discret (ou Discrete Cosine Transform), la DCT.

Dans la section 1.2, on a vu que cette transformation DCT [ANR74] [RY90] [Str99] s'applique directement sur des blocs 8×8 des images naturelles. Elle est définie par la formule (1.2) qui est calculée en nombres flottants.

Un des inconvénients de cette transformée DCT est qu'elle soit calculée en flottants. En effet, ces calculs effectués à partir de valeurs numériques donc entières (c.à.d. les valeurs des pixels des images) donnent des coefficients non entiers. Ces coefficients sont ensuite quantifiés ce qui a pour action de les arrondir, arrondis non réversible au décodage. De plus, au décodage, la formule de reconstruction (1.3) est elle aussi calculée en flottants sur des coefficients déquantifiés donc entiers. Les valeurs reconstruites obtenues sont alors arrondies afin de pouvoir définir des valeurs entières de pixels.
Ces arrondis effectués au codage et au décodage entraînent généralement une dérive entre le codeur et le décodeur et empêchent ainsi la reconstruction parfaite.

Cette version de la transformée DCT est la plus ancienne [ANR74] et est utilisée dans de nombreuses normes de codage d'images et vidéo telles que JPEG [ISO94a], MPEG-x [ISO93] [ISO94b] [ISO00a], H.26x [ITU90] [ITU94] [ITU95]. Les deux seules exceptions sont :

- comme on a pu le voir dans la section 1.2.6, H.264 MPEG-4/AVC [JVT05] [JVT04] qui utilise une approximation de la transformée DCT. Cette dernière est calculée matriciellement sur des blocs de taille 4×4 (ou 8×8 avec l'Amendement FREXt [JVT04] de cette norme). Elle sera présentée plus en détail dans la chapitre 4.1 et en Annexe A.3.
- JPEG2000 [ISO00b] qui utilise un autre type de transformation que la DCT : la transformée en ondelettes.

En effet, cette étape de transformation dans un codeur vidéo a toujours été normalisée avec une transformée DCT, mais beaucoup d'autres types de transformations sont possibles.
Ce chapitre présente et permet d'évaluer les performances en compression de plusieurs de ces transformations telles que :

- les transformées en ondelettes continues, discrètes, sous forme lifting,... (cf 2.2)
- des transformées associant les ondelettes et la DCT : les DCT en lifting comme la BinDCT, l'IntDCT (cf 2.3)
- les transformées à recouvrement LOT, LBT,... (cf 2.4)

2.2 Les ondelettes de première génération

Les ondelettes sont issues d'investigations menées sur le traitement numérique de signaux sismiques par le géophysicien J. Morlet au début des années 80. Il fut ensuite assisté par le physicien A. Grossmann pour résoudre des problèmes de physique théorique [GM84].
Depuis de nombreux travaux ont été menés en image dans le domaine des ondelettes, et notamment par Y. Meyer [Mey90], S. Mallat [Mal00a], I. Daubechies [Dau90] et W. Sweldens [JS94].

Les ondelettes sont apparues par manque de représentations temps-fréquence efficaces. En effet, la représentation alors classique est la transformée de Fourier qui ne permet pas d'analyser les signaux conjointement en temps et en fréquences puisque cette transformée masque l'évolution temporelle des signaux. Cependant, comme nous le montrerons, les transformées en ondelettes pallient ce problème.

Il existe au moins deux manières de définir les ondelettes. La première est la définition de la transformée en ondelettes continues que l'on discrétise. Et la seconde est l'analyse multirésolution. Ces deux approches bien que différentes restent cependant équivalentes comme a pu le montrer Mallat [Mal00a].

2.2.1 La transformée en ondelettes continues

Les ondelettes continues

Une ondelette continue ψ est une fonction réelle de $\mathbb{R}$ dans $\mathbb{R}$ dont la transformée de Fourier vérifie certaines conditions de régularité et de localisation.
Soit ψ une fonction de $L^2(\mathbb{R})$ et sa transformée de Fourier $\mathbf{\Psi}(\nu) = TF(\psi(t))$, si cette transformée vérifie la condition d'admissibilité :

$$C_\psi = \int_0^\infty \frac{|\mathbf{\Psi}(\nu)|^2}{\nu} d\nu < \infty \tag{2.1}$$

et si ψ a au moins un moment nul, c'est-à-dire $\psi \in L^1(\mathbb{R})$ ou encore :

$$\int_{\mathbb{R}} \psi(t) dt = 0 \tag{2.2}$$

alors la fonction ψ est appelée *ondelette mère*.

De plus, on dira que l'ondelette mère ψ possède un moment nul d'ordre k si :

$$\int_{\mathbb{R}} t^k \, \psi(t) dt = 0 \tag{2.3}$$

La transformée en ondelettes continues (TOC) associée à cette ondelette mère ψ est alors définie par :

$$\forall f \in L^2(\mathbb{R}), \forall a \in \mathbb{R}_+^*, \forall b \in \mathbb{R}, \qquad W_f(a,b) = \frac{1}{\sqrt{a}} \int_{\mathbb{R}} f(t) \, \psi\Big(\frac{t-b}{a}\Big) \, dt \tag{2.4}$$

Famille d'ondelettes continues

On peut à partir de l'ondelette mère ψ construire une famille d'ondelettes par dilatation et translation de ψ. Le paramètre a est appelé paramètre d'échelle et permet de dilater l'ondelette mère ψ, alors que le paramètre b permet de la translater.
Soit l'espace $\Gamma = \mathbb{R}_+^* \times \mathbb{R}$, la famille d'ondelettes $\psi_{a,b}(t)$ issue de l'ondelette mère $\psi(t)$ est définie par :

$$\forall (a,b) \in \Gamma, \qquad \psi_{a,b}(t) = \frac{1}{\sqrt{a}}\,\psi\Big(\frac{t-b}{a}\Big) \tag{2.5}$$

La transformée en ondelettes continues (TOC) peut alors être définie par :

$$W_f(a,b) = < \psi_{a,b}, f > \tag{2.6}$$

Reconstruction

La transformée en ondelettes continues TOC est un opérateur linéaire injectif continu. Ceci permet une reconstruction selon la formule suivante :

$$\forall f \in L^2(\mathbb{R}) \text{ et continue sur } \mathbb{R}, \qquad f(t) \;=\; \frac{1}{C_\psi} \int_{\mathbb{R}_+} \frac{da}{a^2} \int_{\mathbb{R}} W_f(a,b)\,\psi_{a,b}(t)\,db \tag{2.7}$$

2.2.2 La transformée en ondelettes discrètes

De par la nature numérique des signaux à analyser, on est amené à discrétiser cette transformation en ondelettes (2.4).

Dans le cas continu, l'ondelette mère $\psi(t)$ permet de construire une famille d'ondelettes $\psi_{a,b}(t)$ en faisant varier les paramètres a et b dans $\Gamma = \mathbb{R}_+^* \times \mathbb{R}$.

Pour discrétiser cette transformation, on limite ces paramètres à une grille dyadique. Pour cela, on fixe $a_0 > 1$ et $b_0 > 0$, et on choisit $a_m = a_0^{-m}$ et $b_n = n.b_0$ avec $(m,n) \in \mathbb{Z}^2$. Le réseau $\{(a_m, b_n) | (m,n) \in \mathbb{Z}^2\}$ définit alors une grille dyadique.

La famille d'ondelettes discrètes $\psi_{j,k}(t)$ issue de l'ondelette mère $\psi(t)$ s'écrit alors :

$$\begin{aligned} \forall (j,k) \in \mathbb{Z}^2, \qquad \psi_{j,k}(t) &= \frac{1}{\sqrt{a_j}}\,\psi\Big(\frac{t-b_k}{a_j}\Big) \\ &= a_0^{j/2}\,\psi\big(a^j(t-k.b_0)\big) \end{aligned} \tag{2.8}$$

Et la transformation en ondelettes discrètes correspondante :

$$W_f(j,k) = < \psi_{j,k}, f > = a_0^{j/2} \int_{\mathbb{R}} f(t)\,\psi\big(a^j(t-k.b_0)\big)\,dt \tag{2.9}$$

2.2.3 L'analyse multirésolution

L'analyse multirésolution (AMR) consiste à décomposer un signal sur une gamme très étendue d'échelles, opération que l'on peut comparer à une cartographie. En allant des échelles les plus grossières vers les échelles les plus fines, on accède à des représentations de plus en plus précises du signal donné. L'analyse s'effectue en calculant ce qui diffère d'une échelle à l'autre, c'est-à-dire les détails à une résolution donnée. Ceux-ci permettent, en corrigeant une approximation encore assez grossière, d'accéder à une représentation de meilleure qualité. On appelle cette propriété propre aux ondelettes : la scalabilité ou l'échelonnabilité.

D'après Sweldens [JS94], Mallat [Mal89a] [Mal00a] et Meyer [Mey90], on définit l'analyse multirésolution AMR de $L^2(\mathbb{R})$ comme étant une suite de sous-espaces vectoriels fermés V_j de $L^2(\mathbb{R})$ vérifiant les propriétés suivantes :

1. $(V_j)_{j\in\mathbb{Z}}$ forme une suite d'espaces emboîtés : $V_{j+1} \subset V_j$.
2. $\bigcup_{j\in\mathbb{Z}} V_j$ est dense dans $L^2(\mathbb{R})$.
3. $\bigcap_{j\in\mathbb{Z}} V_j = \{0\}$.
4. $f(t) \in V_j$ si et seulement si sa dilatée $f(t/2) \in V_{j+1}$.
5. il existe $\phi(t) \in L^2(\mathbb{R})$, appelée *fonction d'échelle* ou *ondelette père*, telle que : $\{\phi(t-k)|k \in \mathbb{Z}\}$ forme une base de V_0.

Dans ces conditions, l'analyse multirésolution d'un signal $f(t)$ de $L^2(\mathbb{R})$ consiste à calculer les projections orthogonales successives de $f(t)$ sur les sous-espaces $V_j, j \in \mathbb{Z}$.

On peut remarquer à l'aide de la propriété 4. qu'il faut multiplier le facteur d'échelle par 2 pour passer du sous-espace V_j au sous-espace V_{j+1}. C'est pourquoi j est appelé *niveau de résolution*. Par récursivité, on déduit que :

$$\forall j \in \mathbb{Z}, \quad f(t) \in V_0 \quad \text{si et seulement si} \quad f(t/2^j) \in V_j$$

De plus, la propriété 5. implique que l'espace V_j est invariant par translation d'un multiple de 2^j :

$$\forall k \in \mathbb{Z}, \quad f(t) \in V_j \Rightarrow f(t - k.2^j) \in V_j \tag{2.10}$$

De ces deux remarques, on peut déduire que la famille :

$$\phi_{j,k}(t) = \{2^{-j/2}.\phi(t/2^j - k) \,|\, (j,k) \in \mathbb{Z}^2\} \tag{2.11}$$

forme une base orthonormale de V_j.

2.2.4 Les bancs de filtres et le lifting

Précédemment, nous avons défini l'analyse multirésolution avec les sous-espaces vectoriels V_j. Essayons maintenant de faire un rapprochement entre l'AMR, les ondelettes continues et les bancs de filtres.

La première propriété de ces espaces permet de dire que V_1 est un sous-espace de V_0. Ceci implique que toute fonction de V_1 peut être écrite comme une combinaison linéaire des fonctions $\phi(t-l)$, $l \in \mathbb{Z}$ qui forment une base de V_0. D'après l'équation (2.11), la fonction $2^{-1/2}.\phi(t/2) \in V_1$, et peut donc être décrite à l'aide d'une suite $(h_0[l])_{l\in\mathbb{Z}}$ telle que :

$$\frac{1}{2^{1/2}}\phi\Big(\frac{t}{2}\Big) = \sum_{l\in\mathbb{Z}} h_0[l].\phi(t-l) \tag{2.12}$$

L'orthonormalité de la famille $\{\phi(t-l)\,|\,l\in\mathbb{Z}\}$ nous permet d'obtenir directement les coefficients $h_0[l]$:

$$h_0[l] = \Big\langle \frac{1}{2^{1/2}}\phi\Big(\frac{t}{2}\Big),\, \phi(t-l)\Big\rangle \tag{2.13}$$

L'équation (2.12) est appelée *équation à 2 échelles* puisqu'elle permet de lier la dilatée d'un facteur 2 avec des versions translatées de cette fonction. Avec cette équation et un changement de variable, il est possible d'exprimer l'équation à 2 échelles entre les niveaux de résolution V_j et V_{j+1} :

$$\frac{1}{2^{\frac{j+1}{2}}}\phi\Big(\frac{t}{2^{j+1}} - k\Big) = \sum_{l\in\mathbb{Z}} h_0[l-2k].\frac{1}{2^{j/2}}\phi\Big(\frac{t}{2^j} - l\Big) \tag{2.14}$$

La suite $(h_0[l])_{l\in\mathbb{Z}}$ ainsi définie peut être considérée comme la réponse d'un filtre, appelé *filtre miroir conjugué*. D'après Meyer [Mey90] et Mallat [Mal89b] [Mal89a], la transformée de Fourier de sa réponse impulsionnelle $\mathbf{H_0}(\nu)$ doit vérifier certaines conditions nécessaires et suffisantes pour que h_0 puisse engendrer une fonction d'échelle :

- $|\mathbf{H_0}(0)| = \sqrt{2}$.
- $|\mathbf{H_0}(\nu)|^2 + |\mathbf{H_0}(\nu+1/2)|^2 = 2$

Dans ce cas, la fonction d'échelle $\phi(t)$ engendrée par $h_0[l]$ a pour transformée de Fourier :

$$\mathbf{\Phi}(\nu) = \prod_{p=1}^{+\infty} \frac{\mathbf{H_0}(2^{-p}\nu)}{2^{1/2}} \tag{2.15}$$

On peut définir le supplémentaire orthogonal W_{j+1} de V_{j+1} dans V_j qui est alors l'espace de détails tel que :

$$V_j = V_{j+1} \oplus W_{j+1} \tag{2.16}$$

Il existe sur cet espace W_{j+1} une famille $\psi_{j+1,k}$ qui forme une base orthonormale, comme la famille $\phi_{j,k}$ sur V_j. Toujours d'après Meyer et Mallat, cette famille peut être décomposée sur la base orthonormale de V_j selon (voir (2.14)) :

$$\forall k \in \mathbb{Z},\quad \frac{1}{2^{\frac{j+1}{2}}}\psi\Big(\frac{t}{2^{j+1}} - k\Big) = \sum_{l\in\mathbb{Z}} h_1[l-2k].\frac{1}{2^{j/2}}\phi\Big(\frac{t}{2^j} - l\Big) \tag{2.17}$$

Comme précédemment, la suite $(h_1[l])_{l\in\mathbf{Z}}$ peut être considérée comme la réponse impulsionnelle d'un filtre dont la transformée de Fourier doit vérifier :

- $|\mathbf{H_1}(\nu)|^2 + |\mathbf{H_1}(\nu+1/2)|^2 = 2$
- mais aussi $\mathbf{H_0}(\nu)\mathbf{H_1^*}(\nu) + \mathbf{H_0}(\nu+1/2)\mathbf{H_1^*}(\nu+1/2) = 0$ (orthogonalité de V_{j+1} et W_{j+1})
- et $|\mathbf{H_0}(\nu)|^2 + |\mathbf{H_1}(\nu)|^2 = 2$ soit $\mathbf{H_1}(0) = 0$.

Nous avons ainsi défini deux filtres $h_0[l]$ (passe-haut) et $h_1[l]$ (passe-bas) comme étant les filtres miroirs conjugués de la fonction d'échelle $\phi(t)$ et de l'ondelette mère $\psi(t)$ respectivement.

On appelle $a_j[k]$ la projection orthogonale du signal $f(t)$ sur l'espace d'approximations V_j et $d_j[k]$ sa projection orthogonale sur l'espace de détails W_j. On déduit facilement que :

$$a_j[k] = \sum_{l\in\mathbb{Z}} a_{j-1}[l].h_0^*[l-2k] \tag{2.18}$$

$$d_j[k] = \sum_{l\in\mathbb{Z}} a_{j-1}[l].h_1^*[l-2k] \tag{2.19}$$

Il s'agit ici de calculer :

$$\tilde{a}_j[k] = \sum_{l\in\mathbb{Z}} a_{j-1}[l].h_0^*[l-k] \tag{2.20}$$

$$\tilde{d}_j[k] = \sum_{l\in\mathbb{Z}} a_{j-1}[l].h_1^*[l-k] \tag{2.21}$$

où $(\tilde{h}[l]) = (h^*[-l])$ et d'effectuer un sous-échantillonnage d'un facteur 2 pour passer du niveau $j-1$ au niveau de résolution j, opérations que l'on peut représenter suivant :

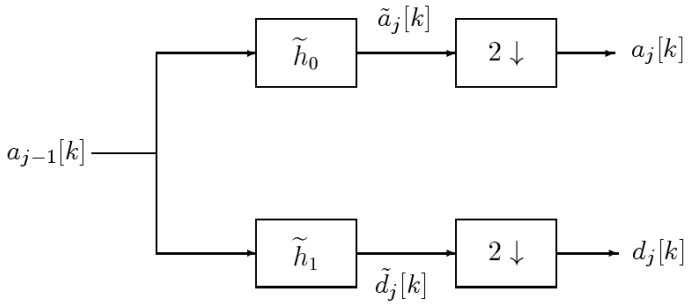


FIG. 2.1 – Banc de filtres d'analyse

La reconstruction est effectuée à l'aide d'un banc de filtres de synthèse qui a pour but de réaliser les opérations duales de celles de l'analyse. Une interpolation d'un facteur 2 est la duale du sous-échantillonnage et consiste à intercaler un 0 entre deux échantillons successifs :

$$\breve{a}_j[k] = \begin{cases} a_j[k/2] & \text{si } k \text{ est pair} \\ 0 & \text{sinon} \end{cases} \tag{2.22}$$

Le filtrage est alors réalisé avec les filtres $(h_0[l])$ et $(h_1[l])$ et la formule de reconstruction pour passer du niveau j au niveau de résolution $j-1$ est donnée par :

$$a_{j-1}[k] = \sum_{l\in\mathbb{Z}} h_0[k-2l].a_j[l] + \sum_{l\in\mathbb{Z}} h_1[k-2l].d_j[l] \tag{2.23}$$

Ce banc de filtres de synthèse peut alors être représenté par :

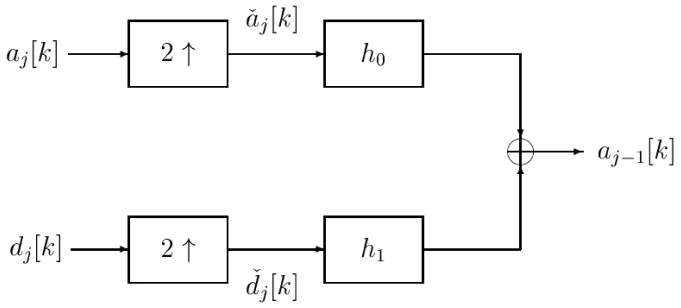


FIG. 2.2 – Banc de filtres de synthèse

On peut alors effectuer ces opérations d'analyse et de synthèse sur plusieurs niveaux à partir des échantillons discrétisés de la fonction $f(t)$ à analyser. En d'autres termes, on part avec $a_0[k] = f[k]$ et on calcule successivement les $a_j[k]$ et $d_j[k]$ sur le nombre de niveaux j souhaités en cascadant les bancs de filtres d'analyse.

Cette décomposition multi-échelle peut être représentée par :

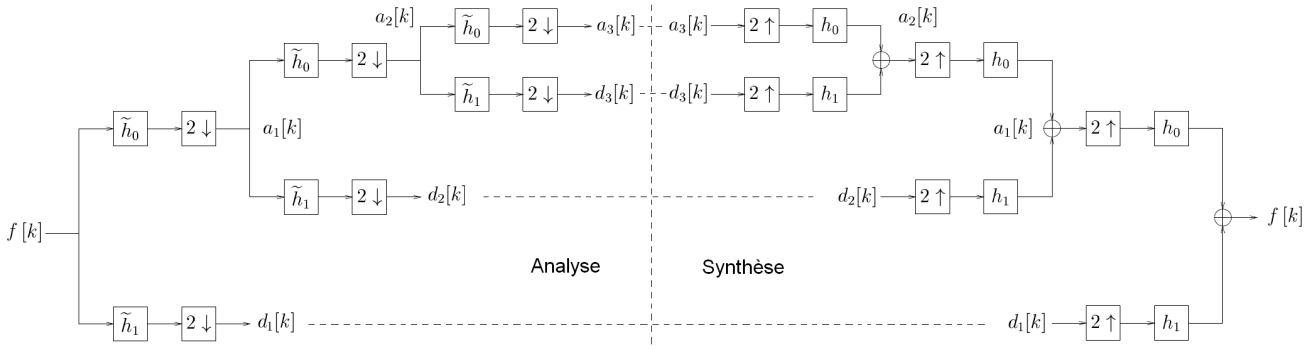


FIG. 2.3 – Décomposition en ondelettes sur 3 niveaux de résolution

Fonctions d'échelles et ondelettes

On suppose que l'on dispose d'une analyse multirésolution de $L^2(\mathbb{R})$ avec sa fonction d'échelle $\phi(t)$ et son filtre miroir conjugué $(h_0[l])_{l\in\mathbb{Z}}$.

A cette AMR correspond une ondelette mère $\psi(t)$ dont la transformée de Fourier $\mathbf{\Psi}(\nu)$ doit vérifier :

$$\mathbf{\Psi}(\nu) = \frac{1}{\sqrt{2}}.\mathbf{H_1}\left(\frac{\nu}{2}\right).\mathbf{\Phi}\left(\frac{\nu}{2}\right) \tag{2.24}$$

$$\text{soit,} \qquad \psi(t) = \sum_{l\in\mathbb{Z}} h_1[l].\sqrt{2}.\phi(2t-l)$$

$$\text{avec} \qquad \mathbf{H_1}(\nu) = e^{-i\nu}.\mathbf{H_0^*}(\nu+1/2) \tag{2.25}$$

$$\text{soit,} \qquad h_1[l] = (-1)^l.h_0^*[1-l]$$

Dans ce cas et pour toute échelle j, $\{\psi_{j,k}\}_{k\in\mathbb{Z}}$ est une base orthonormée de W_j. D'où $\{\psi_{j,k}\}_{(j,k)\in\mathbb{Z}^2}$ forme une base orthonormée de $L^2(\mathbb{R})$ et ainsi ψ est une ondelette mère orthogonale qui génère une famille d'ondelettes de la forme :

$$\psi_{j,k}(t) = \frac{1}{\sqrt{2^j}}.\psi\Big(\frac{t}{2^j} - k\Big) \quad \text{(soit avec } a_j = 2^j \text{ et } b_{j,k} = k.2^j\text{)} \tag{2.26}$$

Les ondelettes biorthogonales

Dans ce qui a été présenté, les filtres définis sont orthogonaux et donc à phase non linéaire. Cependant cette propriété de phase linéaire peut être importante et notamment en traitement d'images. On préférera alors utiliser des filtres biorthogonaux et des ondelettes biorthogonales [CDF92].

Pour construire ces ondelettes biorthogonales, il faut relaxer la condition d'orthonormalité de la base. Pour cela, on utilise deux bases d'ondelettes non orthonormales de $L^2(\mathbb{R})$, une pour l'analyse $\{\psi_1\}$ et une pour la synthèse $\{\psi_2\}$.

Cependant, ces deux bases, définies comme précédemment, $\{2^{-j/2}\psi_1(t/2^j - k)\,|\,(j,k)\in\mathbb{Z}^2\}$ et $\{2^{-j'/2}\psi_2(t/2^{j'} - k')\,|\,(j',k')\in\mathbb{Z}^2\}$ vérifient la propriété de biorthogonalité :

$$\Big\langle \frac{1}{2^{j/2}}\psi_1\Big(\frac{t}{2^j} - k\Big),\, \frac{1}{2^{j'/2}}\psi_2^*\Big(\frac{t}{2^{j'}} - k'\Big)\Big\rangle = \begin{cases} 1 & \text{si } j = j' \text{ et } k = k' \\ 0 & \text{sinon} \end{cases} \tag{2.27}$$

L'analyse restera donc identique à celle présentée en version orthogonale avec les mêmes définitions de l'ondelette ψ_1, de la fonction d'échelle ϕ_1 et des filtres miroirs conjugués $(\tilde{h}_0)$ et $(\tilde{h}_1)$. Et, ce sont les filtres de synthèse (h_0) et (h_1) qui sont modifiés tels que :

$$\begin{aligned} \mathbf{H_1}(\nu) &= e^{-i\nu}.\widetilde{\mathbf{H}}_\mathbf{0}^*(\nu + 1/2) \quad \text{soit}, h_1[l] = (-1)^l.\tilde{h}_0[1-l] \\ \widetilde{\mathbf{H}}_\mathbf{1}(\nu) &= e^{-i\nu}.\mathbf{H}_\mathbf{0}^*(\nu + 1/2) \quad \text{soit}, \tilde{h}_1[l] = (-1)^l.h_0[1-l] \end{aligned} \tag{2.28}$$

On relaxe ainsi les conditions propres à l'orthogonalité des filtres, c'est-à-dire : $(\tilde{h_i}[l]) = (h_i^*[-l])$, $i = 1, 2$.

La condition de biorthogonalité (2.27) devient alors pour les filtres miroirs conjugués :

$$\widetilde{\mathbf{H}}_\mathbf{0}^*(\nu).\mathbf{H_0}(\nu) + \widetilde{\mathbf{H}}_\mathbf{0}^*(\nu + 1/2).\mathbf{H_0}(\nu + 1/2) = 1 \tag{2.29}$$

Dans le cas orthogonal, on avait $(\tilde{h_i}[l]) = (h_i^*[-l])$ et la condition de biorthogonalité (2.29) était la condition d'orthogonalité : $|\mathbf{H_0}(\nu)|^2 + |\mathbf{H_0}(\nu + 1/2)|^2 = 2$.

La conception des ondelettes biorthogonales est donc plus aisée que celle d'ondelettes orthogonales. En effet, dans le cas biorthogonal, on n'impose pas que les filtres d'analyse (respectivement de synthèse) soient orthogonaux entre eux.

Le lifting

W. Sweldens est le premier à avoir introduit le lifting [SS96] [Swe96] [Swe97b] [Swe97a], mais depuis beaucoup d'autres travaux ont été menés sur ce procédé très performant [BVDE92] [DS98] [CDSY98].

Cette technique est algébrique et a pour but de faciliter la réalisation des filtres miroirs en les factorisant en filtres simples. Grace à cette méthode, on ne fait plus du tout appel à la transformée de Fourier, et on n'utilise plus les notions de translation et dilatation d'une ondelette mère.

Ce procédé est basé sur la propriété suivante :

Soit un système de filtres biorthogonaux $\{h_0, \tilde{h}_0, h_1, \tilde{h}_1\}$ *(défini comme précédemment), le système* $\{h, \tilde{h}, g, \tilde{g}\}$ *défini par :*

$$\begin{aligned} \mathbf{H}(\nu) = \mathbf{H_0}(\nu) \qquad & \text{et} \qquad \widetilde{\mathbf{H}}(\nu) = \widetilde{\mathbf{H}}_\mathbf{0}(\nu) + \widetilde{\mathbf{H}}_\mathbf{1}(\nu).\mathbf{S}(\nu) \\ \mathbf{G}(\nu) = \mathbf{H_1}(\nu) - \mathbf{H_0}(\nu).\mathbf{S}^T(\nu) \qquad & \text{et} \qquad \widetilde{\mathbf{G}}(\nu) = \widetilde{\mathbf{H}}_\mathbf{1}(\nu) \end{aligned} \tag{2.30}$$

avec $\mathbf{S}(\nu)$ *un polynôme trigonométrique, est aussi un système de filtres biorthogonaux.*

Cette modification des filtres entraîne aussi une modification des fonctions d'échelle ϕ_1, ϕ_2 et des ondelettes ψ_1, ψ_2 selon :

$$\begin{aligned} \psi(t) &= \psi_1(t) - \sum_k s_k.\phi_1(t-k) \\ \tilde{\phi}(t) &= 2\sum_k \tilde{h}_0[k].\phi_2(2t-k) + \sum_k s_{-k}\psi_2(t-k) \\ \tilde{\psi}(t) &= 2\sum_k \tilde{h}_1[k].\phi_2(2t-k) \end{aligned} \tag{2.31}$$

La condition de biorthogonalité (2.29) peut être écrite sous forme matricielle selon :

$$\begin{bmatrix} \widetilde{\mathbf{H}}_\mathbf{0} \\ \widetilde{\mathbf{H}}_\mathbf{1} \end{bmatrix} . \begin{bmatrix} \mathbf{H}_\mathbf{0}^* & \mathbf{H}_\mathbf{1}^* \end{bmatrix} = I \tag{2.32}$$

Donc pour les nouveaux filtres, on a :

$$\begin{bmatrix} \widetilde{\mathbf{H}} \\ \widetilde{\mathbf{G}} \end{bmatrix} . \begin{bmatrix} \mathbf{H}^* & \mathbf{G}^* \end{bmatrix} = \begin{bmatrix} 1 & \mathbf{S} \\ 0 & 1 \end{bmatrix} . \begin{bmatrix} 1 & -\mathbf{S} \\ 0 & 1 \end{bmatrix} = I \tag{2.33}$$

et ils vérifient bien la condition de biorthogonalité.

Comme on a pu le voir précédemment, la reconstruction parfaite sera obtenue si et seulement si les filtres vérifient certaines conditions que l'on peut mettre sous forme matricielle à l'aide de *la matrice de modulation* M :

$$M = \begin{bmatrix} \mathbf{H_0} & \widetilde{\mathbf{H}}_\mathbf{0} \\ \mathbf{H_1} & \widetilde{\mathbf{H}}_\mathbf{1} \end{bmatrix}$$

La condition de reconstruction parfaite sera alors :

$$\widetilde{M}^T.M = 2I \tag{2.34}$$

Et il en est de même pour les nouveaux filtres :

$$M = \begin{bmatrix} \mathbf{H} & \widetilde{\mathbf{H}} \\ \mathbf{G} & \widetilde{\mathbf{G}} \end{bmatrix}$$

Une autre représentation que la matrice de modulation est possible, il s'agit de *la matrice polyphase* qui est définie telle que :

$$P = \begin{bmatrix} \mathbf{H_e} & \mathbf{G_e} \\ \mathbf{H_o} & \mathbf{G_o} \end{bmatrix}$$

où h_e (resp. g_e) contient les coefficients pairs et h_o (resp. g_o) les coefficients impairs du filtre h (resp. g).
Et dans ce cas, la condition de reconstruction parfaite sera donnée par :

$$P.\widetilde{P}^T = I \tag{2.35}$$

Comme on a pu le voir, il est possible, à partir des filtres miroirs conjugués, de construire de nouveaux filtres biorthogonaux (2.30).

Dans le cas direct, on a :

$$\mathbf{G}^{new} = \mathbf{H_1} + \mathbf{H_0}.\mathbf{S} \quad \text{et} \quad \widetilde{\mathbf{H}}^{new} = \widetilde{\mathbf{H}}_\mathbf{0} - \widetilde{\mathbf{H}}_\mathbf{1}.\mathbf{S}^T \tag{2.36}$$

Auquel cas, la matrice polyphase devient :

$$P^{new} = P.\begin{bmatrix} 1 & \mathbf{S} \\ 0 & 1 \end{bmatrix}$$

Et dans le cas dual, on peut écrire :

$$\mathbf{H}^{new} = \mathbf{H_0} + \mathbf{H_1}.\mathbf{T} \quad \text{et} \quad \widetilde{\mathbf{G}}^{new} = \widetilde{\mathbf{H}}_\mathbf{1} - \widetilde{\mathbf{H}}_\mathbf{0}.\mathbf{T}^T \tag{2.37}$$

Auquel cas, la matrice polyphase duale devient :

$$\widetilde{P}^{new} = \widetilde{P}.\begin{bmatrix} 1 & 0 \\ \mathbf{T} & 1 \end{bmatrix}$$

On a alors à l'aide de la matrice polyphase P^{new} une décomposition directe utilisant un étage lifting appelé *prédiction* de la forme :

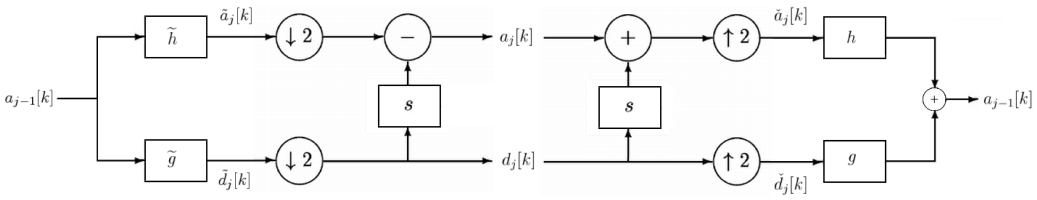


FIG. 2.4 – Décomposition avec la matrice polyphase P^{new} utilisant un étage de prédiction

Dans le cas dual, la matrice polyphase $\widetilde{P}^{new}$ utilise un étage lifting dual appelé *mise à jour* :

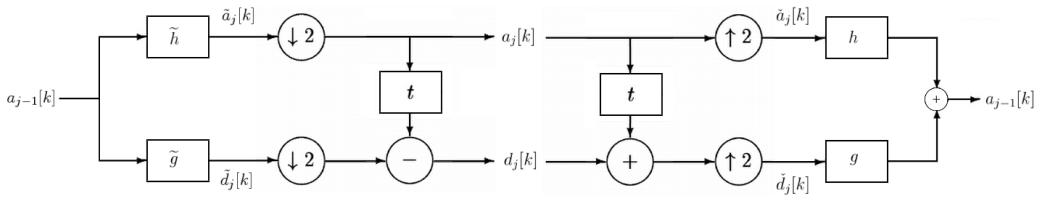


FIG. 2.5 – Décomposition duale avec la matrice polyphase $\widetilde{P}^{new}$ utilisant un étage de mise à jour

Grâce à un algorithme euclidien, il est démontrable [DS98] que l'on peut continuer cette factorisation de la matrice polyphase. Toute matrice polyphase vérifiant (2.35) est décomposable en une série de produits de matrices simples telle que :

$$P^{new} = \prod_{i=1}^{m} \begin{bmatrix} 1 & \mathbf{S_i} \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 \\ \mathbf{T_i} & 1 \end{bmatrix} \cdot \begin{bmatrix} K & 0 \\ 0 & 1/K \end{bmatrix} \tag{2.38}$$

Et pour la matrice polyphase duale :

$$\widetilde{P}^{new} = \prod_{i=1}^{m} \begin{bmatrix} 1 & 0 \\ -\mathbf{S_i} & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & -\mathbf{T_i} \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1/K & 0 \\ 0 & K \end{bmatrix} \tag{2.39}$$

En d'autres termes, tout banc de filtres biorthogonaux d'ondelettes peut être décomposé en une séparation pair/impair suivi de m étages de prédiction (les impairs servent à prédire les pairs) et de mise à jour (l'erreur de prédiction), et enfin d'une mise à l'échelle.

La décomposition s'effectuera alors par :

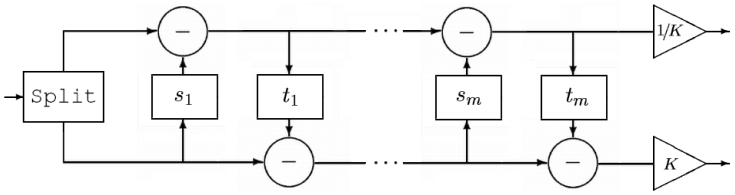


FIG. 2.6 – Décomposition en étages lifting

Et la reconstruction obtenue directement par parcours inverse :

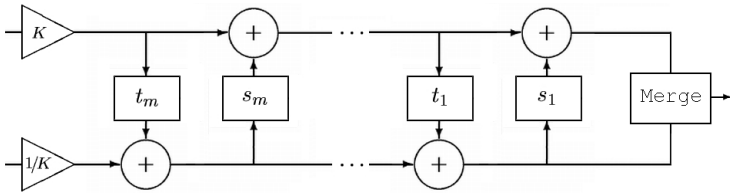


FIG. 2.7 – Reconstruction en étages lifting

2.2.5 Les ondelettes en deux dimensions

Jusqu'ici nous n'avons considéré que des signaux et des transformées en ondelettes à une seule dimension. Mais, il est possible de définir des bases d'ondelettes pour des signaux à deux dimensions tels que les images.
Pour cela, il existe plusieurs méthodes : celles avec des ondelettes séparables et celles avec des ondelettes non séparables.

Dans la plupart des cas, on utilise une analyse multirésolution séparable, ou des ondelettes séparables. On considère alors que l'analyse bidimensionnelle est le produit de deux analyses unidimensionnelles comme définies précédemment.

L'analyse d'une image sera alors réalisée en effectuant d'abord l'analyse sur les lignes, puis sur les colonnes.
Le signal à analyser $f(x, y)$ est d'abord filtré passe-bas sur les lignes avec $\tilde{h}$, et passe-haut sur les lignes avec $\tilde{g}$, puis sous-échantillonné, on obtient ainsi deux signaux : $s_B(x, y)$ pour les basses fréquences et $s_H(x, y)$ pour les hautes fréquences selon :

$$s_B(x, y) = \sum_k \tilde{h}[k].f(2x - k, y) \quad (\text{resp. } \tilde{g} \text{ pour } s_H(x, y))$$

On réitère ensuite les mêmes opérations sur les colonnes, c'est-à-dire qu'on filtre passe-bas et passe-haut avant de sous-échantillonner chacune des approximations $s_B(x, y)$ et $s_H(x, y)$. On obtient ainsi une imagette d'approximation $s_{BB}(x, y)$ ainsi que trois imagettes de détails $s_{BH}(x, y)$, $s_{HB}(x, y)$ et $s_{HH}(x, y)$ selon :

$$\begin{aligned} s_{BB}(x, y) &= \textstyle\sum_k \tilde{h}[k].s_B(x, 2y - k) \quad (\text{resp. } \tilde{g} \text{ pour } s_{BH}(x, y)) \\ s_{HB}(x, y) &= \textstyle\sum_k \tilde{h}[k].s_H(x, 2y - k) \quad (\text{resp. } \tilde{g} \text{ pour } s_{HH}(x, y)) \end{aligned}$$

Ceci permet d'analyser une image sur un niveau. Si on veut plus de niveaux de décomposition, on recommence ce traitement sur l'imagette d'approximation $s_{BB}(x, y)$.

On a alors une décomposition sur plusieurs niveaux telle que :

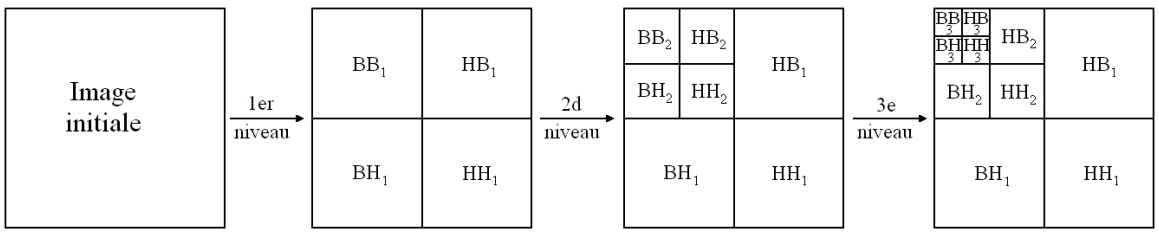


FIG. 2.8 – Décomposition en ondelettes d'une image sur 3 niveaux

Par exemple, pour l'image Lena décomposée sur 3 niveaux, on obtient :

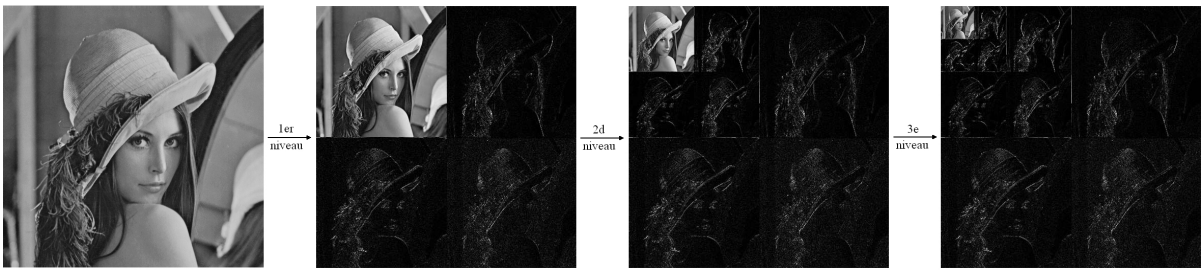


FIG. 2.9 – Décomposition en ondelettes de l'image Lena sur 3 niveaux avec un filtre 9/7 (type JPEG2000) [RSC01]

Pour la reconstruction, on réalise d'abord un sur-échantillonnage avant d'effectuer les mêmes opérations que précédemment avec les filtres de synthèses h et g. On améliore la qualité de la sous-bande basse fréquence $s_{BB}(x, y)$ avec les détails des autres sous-bandes $s_{BH}(x, y)$, $s_{HB}(x, y)$ et $s_{HH}(x, y)$.

Dans le cas d'une décomposition sur plusieurs niveaux, la sous-bande basse fréquence BB_i est amélioré à l'aide des autres sous-bandes de ce niveau i (BH_i, HB_i et HH_i) pour donner la sous-bande basse fréquence du niveau inférieur BB_{i-1}. On réitère ces opérations jusqu'au niveau 0 pour retrouver l'image de qualité initiale.

Par exemple, pour l'image Lena décomposée sur 3 niveaux (fig. 2.9), on a :

(a) Avec 0 niveau (image BB_3)

(b) Avec 1 niveau

(c) Avec 2 niveaux

(d) Avec tous les niveaux

FIG. 2.10 – Reconstruction de l'image Lena décomposée sur 3 niveaux [RSC01]

Cependant, on peut être amené à réaliser une base d'ondelettes non séparables qui sera alors de dimension 2 : $\psi(x, y)$. Ces ondelettes non séparables sont beaucoup plus complexes et difficiles à mettre en oeuvre, mais elles permettent de représenter les contours un peu plus efficacement que les ondelettes séparables, et d'effectuer ainsi une meilleure décorrélation des données. Elles peuvent être utilisées pour des applications de débruitage, d'indexation, de tatouage,...

2.3 La DCT en lifting

2.3.1 Introduction

Comme on a pu le voir dans le chapitre sur le codage vidéo et dans la section 2.1, il est nécessaire, lors du codage, d'employer une transformation fréquentielle afin d'exploiter les redondances spatiales contenues dans les images. C'est généralement la transformation DCT (Discrete Cosine Transform).

Dans toutes les normes des familles MPEG et H.26x sauf H.264 MPEG-4/AVC, la DCT utilisée est une DCT flottante. Elle s'applique à des blocs 8×8 et est définie par :

$$C_{m,n} = \alpha(m)\beta(n)\sum_{i=0}^{7}\sum_{j=0}^{7} B_{i,j}\cos\left(\frac{\pi(2i+1)m}{16}\right)\cos\left(\frac{\pi(2j+1)n}{16}\right) \qquad (2.40)$$
$$\text{avec} \quad 0 \leq m,n \leq 7$$

où $B_{i,j}$ désigne le pixel (i,j) du bloc 8×8 original, $C_{m,n}$ représente le coefficient du bloc DCT 8×8 associé, et :

$$\alpha(m)\,(\text{respectivement}\beta(n)) = \begin{cases} \frac{1}{\sqrt{32}} & \text{si m=0 (respectivement si n=0)} \\ \frac{1}{4} & \text{sinon} \end{cases}$$

Pour la norme H.264 MPEG-4/AVC, la DCT utilisée est une DCT entière. C'est une approximation entière de la DCT flottante qui permet de s'affranchir de la complexité opératoire de la DCT flottante. Elle s'applique à des blocs 4×4 et s'exprime sous forme matricielle :

$$Y = H\,.\,X\,.\,H^T \qquad \text{avec} \quad H = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix} \qquad (2.41)$$

où Y est le bloc de coefficients DCT obtenu par la transformation H du bloc d'entrée X.

La DCT flottante pose un certain nombre de problèmes notamment liés à l'utilisation d'une précision infinie lors des calculs. Cela impose l'utilisation d'arrondis impliquant généralement une divergence entre le codeur et le décodeur. De nombreux travaux ont donc été réalisés afin d'éliminer cette divergence et de diminuer la complexité opératoire de cette transformation.

Ces problèmes sont en partie résolus dans H.264/AVC avec la DCT entière. Mais, celle-ci n'est qu'une approximation de la DCT flottante, elle ne permet pas d'accéder aux mêmes coefficients. Des travaux ont donc été menés pour mieux approcher la DCT flottante ou pour trouver une alternative à la DCT.

2.3.2 La DCT binaire (BinDCT)

Trac D. Tran a réalisé de nombreux travaux afin de diminuer la complexité de la DCT 8×8 flottante utilisée par JPEG et H.263+ [Tra99] [LT00] [Tra00a] [LT01].

Tran se base sur la factorisation de la matrice DCT 8×8 flottante de Chen [CSF77]. Cette factorisation est issue de deux propriétés :

- Toute matrice orthogonale $N \times N$ peut être factorisée en $N(N-1)/2$ rotations.
- Toute matrice inversible $N \times N$ peut être complètement caractérisée par $N(N-1)$ étages lifting, N facteurs diagonaux et éventuellement une matrice de permutation.

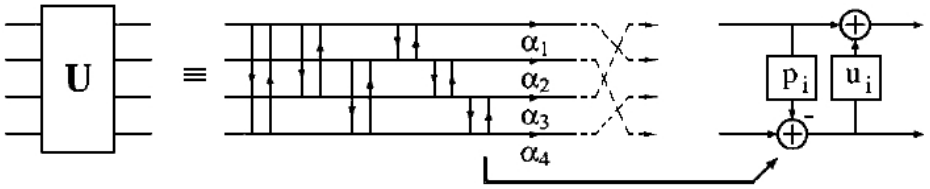


FIG. 2.11 – Décomposition d'une matrice en étages lifting, facteurs diagonaux et permutation [Tra99]

La factorisation de la DCT 8×8 flottante par Chen est alors donnée par son graphe :

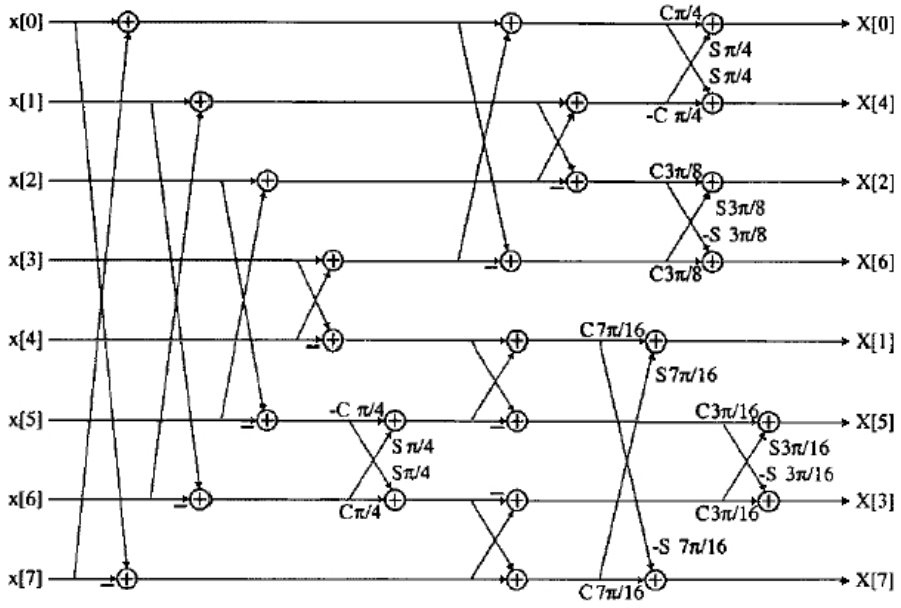


FIG. 2.12 – Graphe de la factorisation de Chen de la DCT 8×8 flottante [LT00]

où, pour les facteurs diagonaux, S désigne le sinus et C le cosinus.

Cette factorisation permet de réaliser la transformation DCT 8×8 flottante en 13 multiplications et 29 additions.

Cependant, toutes ces multiplications se situent au niveau des rotations qui peuvent être représentées sous forme lifting [BVDE92]. Il faut pour cela trois étages lifting (cf fig. 2.13) qui s'écrivent sous forme matricielle selon :

$$\begin{bmatrix} cos(\alpha) & -sin(\alpha) \\ sin(\alpha) & cos(\alpha) \end{bmatrix} = \begin{bmatrix} 1 & \frac{cos(\alpha)-1}{sin(\alpha)} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ sin(\alpha) & 1 \end{bmatrix} \begin{bmatrix} 1 & \frac{cos(\alpha)-1}{sin(\alpha)} \\ 0 & 1 \end{bmatrix} \tag{2.42}$$

D'après le graphe de la factorisation de Chen (cf fig. 2.12), on remarque que certaines des rotations sont situées à l'extrémité du graphe. Il est alors possible d'utiliser une seconde décomposition et de faire migrer des facteurs de remise à l'échelle dans l'étape de quantification pour diminuer la complexité de la transformation. Cette seconde décomposition permet de représenter les rotations à l'aide de 2 étages lifting et 2 facteurs de remise à l'échelle. Cette méthode est appelée le "scaled lifting".

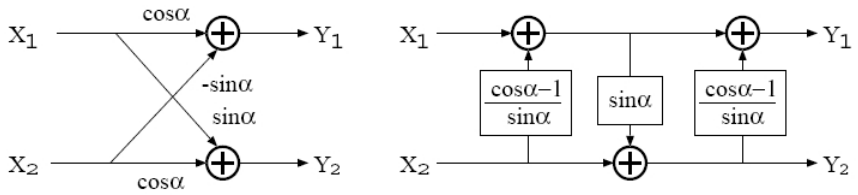


FIG. 2.13 – Une rotation sous la forme de 3 étages lifting [LT00]

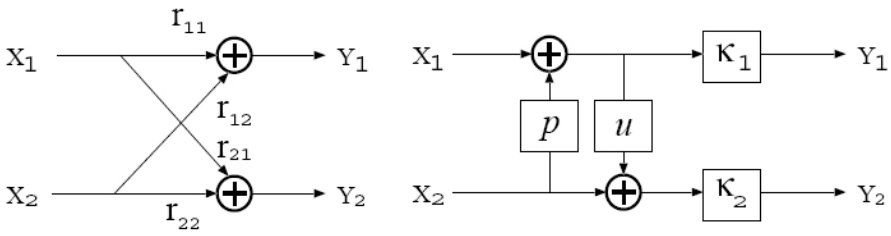


FIG. 2.14 – Une rotation décomposée avec le "scaled lifting" [LT00]

En comparant ces deux décompositions, les paramètres du "scaled lifting" peuvent être déterminés comme étant :

$$\begin{aligned} p &= \frac{r_{12}}{r_{11}} \\ u &= \frac{r_{11} \cdot r_{21}}{r_{11} \cdot r_{22} - r_{21} \cdot r_{12}} \\ \kappa_1 &= r_{11} \\ \kappa_2 &= \frac{r_{11} \cdot r_{22} - r_{21} \cdot r_{12}}{r_{11}} \end{aligned} \tag{2.43}$$

Ces étages lifting traduisent des transformations biorthogonales, leurs inverses auront donc aussi une structure lifting. Les transformations inverses seront simplement obtenues en remplaçant les sommes des transformées directes par des soustractions et réciproquement.

De plus, l'utilisation d'une structure lifting et/ou scaled lifting assure une reconstruction parfaite (sans quantification).

Les paramètres des décompositions lifting peuvent être calculés à partir des formules précédentes (2.42) et (2.43). Cependant, il est préférable d'approcher ces paramètres avec des valeurs dyadiques, c'est-à-dire des valeurs de la forme : $k/2^m$. La division par 2^m sera alors réalisée en effectuant un décalage binaire de m bits.

Dans ce cas, cette transformation n'aura besoin que d'additions et de décalages binaires. Elle n'utilise que des opérations en arithmétique binaire d'où son nom de BinDCT.

La structure générale de la BinDCT peut maintenant être déduite de la factorisation de Chen et des deux décompositions lifting des rotations selon :

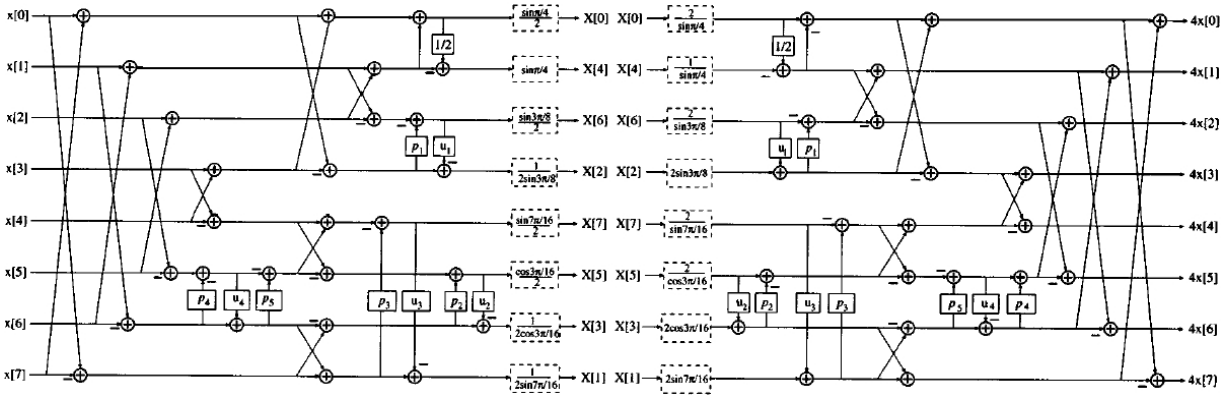


FIG. 2.15 – Structure générale de la transformation BinDCT directe et inverse [LT01]

où les facteurs de remise à l'échelle (encadrés en pointillés) sont absorbés par l'étape de quantification.

Les paramètres des lifting sont donnés en fonction de la configuration de la BinDCT. En effet, il existe différentes configurations Ci pour la BinDCT, chacune ayant une précision différente des approximations p et u. Ces paramètres sont :

	flottante	C1	C2	C3	C4	C5	C6	C7	C8	C9
p1	0.4142135623	13/32	7/16	13/32	7/16	3/8	1/2	1/2	1	0
u1	0.3535533905	11/32	3/8	11/32	3/8	3/8	3/8	1/2	1/2	0
p2	0.6681786379	11/16	5/8	11/16	5/8	7/8	7/8	1	1	0
u2	0.4619397662	15/32	7/16	15/32	7/16	1/2	1/2	1/2	1/2	0
p3	0.1989123673	3/16	3/16	3/16	3/16	3/16	3/16	1/4	0	0
u3	0.1913417161	3/16	3/16	3/16	3/16	3/16	1/4	1/4	0	0
p4	0.4142135623	13/32	13/32	7/16	7/16	7/16	7/16	1/2	0	0
u4	0.7071067811	11/16	11/16	11/16	11/16	11/16	3/4	3/4	1/2	0
p5	0.4142135623	13/32	13/32	3/8	3/8	3/8	3/8	1/2	1/2	0
Shifts	-	23	21	21	19	17	14	9	5	1
Adds	-	42	39	40	37	36	33	28	24	18

TAB. 2.1 – Plusieurs configurations de BinDCT avec leurs paramètres de lifting [LT01]

Avec ces paramètres et ce graphe, il est possible de définir la transformation BinDCT souhaitée pour approcher la transformation DCT 8×8 flottante. Mais cette BinDCT reste une approximation, elle ne fournit pas les mêmes coefficients que la DCT flottante.

Par exemple, il est possible de calculer les coefficients de la matrice de transformation correspondant à la configuration C7 :

Matrice de transformation BinDCT-C7							
1	1	1	1	1	1	1	1
15/16	101/128	35/64	1/4	-1/4	-35/64	-101/128	-15/16
3/4	1/2	-1/2	-3/4	-3/4	-1/2	1/2	3/4
1/2	3/32	-11/16	-1/2	1/2	11/16	-3/32	-1/2
1/2	-1/2	-1/2	1/2	1/2	-1/2	-1/2	1/2
1	-23/16	-1/8	1	-1	1/8	23/16	-1
1/2	-1	1	-1/2	-1/2	1	-1	1/2
1/4	-21/32	13/16	-1	1	-13/16	21/32	-1/4

TAB. 2.2 – Coefficients de la matrice de transformation BinDCT-C7 [LT01]

Cette matrice est différente de celle de la DCT 8×8 flottante, mais elle a la même dynamique suivant les lignes, lui permettant ainsi d'approcher les coefficients transformés.

Pour vérifier les performances de la BinDCT, Tran l'a intégré dans un codeur et un décodeur vidéo de type H.263+. Il a aussi été nécessaire d'en modifier la quantification pour y inclure les facteurs de remise à l'échelle.

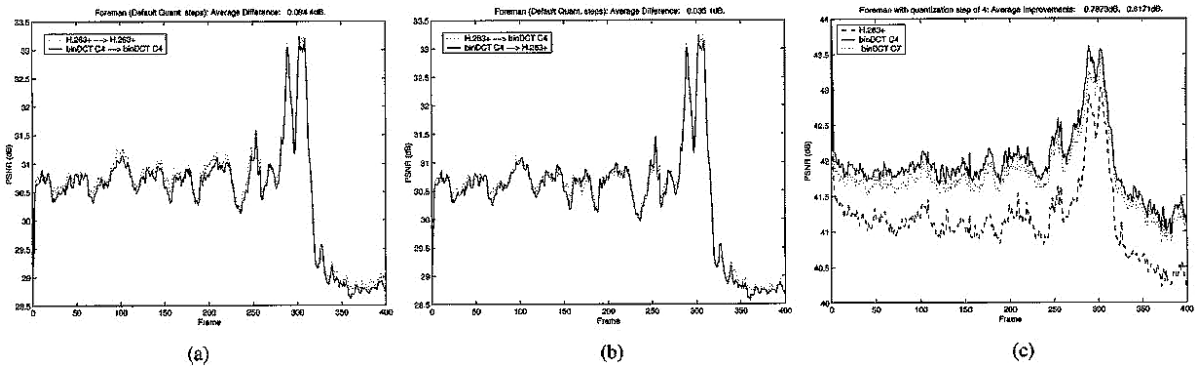


FIG. 2.16 – Comparaison entre H.263+ et la BinDCT adaptée à H.263+ [LT01]

On peut voir sur la première courbe (cf fig. 2.16 a)) que la BinDCT appliquée à H.263+ a les mêmes performances (moins de 0.1 dB de moins) que le H.263+ original (avec la DCT flottante).

Par contre, si on fixe le pas de quantification pour les deux méthodes (cf fig. 2.16 c) avec $Q_{step} = 4$), alors la BinDCT appliqué à H.263+ donne de meilleures résultats (+0.79 dB pour C4 et +0.61 dB pour C7) que la référence H.263+. Cependant, dans cette configuration, la taille du fichier compressé sera augmentée de 2.5%.

De plus, la seconde courbe (cf fig. 2.16 b)) montre que la BinDCT appliqué au codeur donne un flux décodable par H.263+. Et, qu'un flux H.263+ peut être décodé par la BinDCT appliquée à un décodeur H.263+. Ceci permet de dire que la BinDCT est compatible avec H.263+ et sa DCT flottante classique.

On peut donc conclure que la BinDCT-C4 appliqué à H.263+ est similaire en terme de performances à la référence H.263+.

Tran a aussi mesuré la rapidité d'exécution de la BinDCT par rapport à la DCT flottante classique de JPEG en utilisant un PC Pentium-III 550 MHz sous Linux.

Algorithmes	temps (x10^{-6}s)
DCT flottante	119.05
BinDCT-C1	2.45
BinDCT-C4	2.09
BinDCT-C7	2.06

TAB. 2.3 – Temps d'exécution de différentes DCT sur un bloc 8×8 [LT01]

La BinDCT est donc une approximation rapide (cf Tab.2.3), efficace (cf fig. 2.16) et peu complexe (uniquement en arithmétique binaire) de la DCT 8×8 flottante de JPEG et H.263+.

2.3.3 La DCT entière (IntDCT)

De nombreux travaux ont été réalisés afin de définir une transformation DCT entière. Plusieurs techniques ont été testées, chacune utilisant une factorisation différente de la matrice DCT.

Certaines méthodes utilisent une factorisation directe [CSF77], c'est-à-dire que la matrice DCT est factorisée en produit de matrices creuses selon DP_1LUP_2 où D est une matrice diagonale représentant les facteurs de remise à l'échelle, P_1 et P_2 sont des matrices de permutation, et L et U des matrices triangulaires inférieures et supérieures respectivement. Dans ce cas, le coefficient DCT entier est donné par : $c = P_1[L[UP_2x]]$ où [.] représente l'arrondi au plus proche entier.

D'autres méthodes utilisent une factorisation indirecte [CON00], qui consiste à factoriser la matrice DCT en utilisant la transformation de Walsh-Hadamard (WHT). Dans ce cas, la matrice DCT est décomposée en $\sqrt{1/N}BTBH_W$ où B est une matrice réalisant une inversion de bits, H_W est la transformée de Walsh-Hadamard, et T une matrice diagonale.

Les autres techniques, présentées ici [CXL01] [Abh02] [Abh03] [COT+02] [ZCBK01] [WHS01] [HM03] [PT03b] [PT03a], utilisent cette factorisation indirecte, mais couplée à une technique récursive, c'est-à-dire que les coefficients DCT sont calculés à partir de ceux d'une taille plus petite (la DCT N×N est calculée à l'aide des DCT N/2×N/2). Cet ajout de récursivité a pour intérêt de limiter la complexité et les temps d'exécution des calculs : par exemple, on commence par calculer des DCT 2×2 pour accéder aux DCT 4×4 nécessaires au calcul de la DCT 8×8 généralement recherchée.

Pour les méthodes présentées ici, il est important de repartir de la transformée DCT unidimensionnelle d'un signal x et de longueur N qui s'écrit de la manière suivante :

$$X(k) = \epsilon_k \sqrt{\frac{2}{N}} \sum_{n=0}^{N-1} x(n) \cos\left((2n+1)\frac{\pi k}{2N}\right) \qquad \text{où} \quad \epsilon_k = \begin{cases} \frac{1}{\sqrt{2}} & \text{si k=0} \\ 1 & \text{sinon} \end{cases} \tag{2.44}$$

Cette équation (2.44) peut être réécrite sous la forme :

$$X = \alpha_N \, D_N \, x \tag{2.45}$$

où x et X sont respectivement le signal et sa transformée, D_N est la matrice DCT N×N, et α_N est une matrice diagonale contenant des constantes de normalisation.

Afin de pouvoir réaliser une factorisation récursive, il est nécessaire de réarranger la matrice D_N. En effet, en effectuant des permutations sur les lignes et les colonnes, il est possible d'isoler les effets sur les échantillons d'indices pairs de ceux sur les indices impairs. Les lignes paires sont rassemblées dans la partie supérieure de la matrice et les lignes impaires dans la partie inférieure. Les colonnes sont aussi permutées pour que la première et la seconde moitié soient identiques pour les lignes paires et de signe opposé pour les lignes impaires. Ce réarrangement D'_N de la matrice D_N nécessite que le signal x soit aussi réarrangé selon :

$$\left\{ \begin{array}{ll} \tilde{x}(n) & = x(2n) \\ \tilde{x}(N+\frac{n}{2}) & = x(N-(2n+1)) \end{array} \right. \quad \text{pour} \quad n = 0, \ldots, N/2-1 \tag{2.46}$$

Cette modification du signal x permet d'écrire la transformation DCT (2.44) sous forme de deux transformations, une pour les coefficients pairs et une pour les coefficients impairs. Et en notant que les facteurs de normalisation sont donnés par :

$$\alpha_N(k,k) = \sqrt{\frac{2}{N}}\,\epsilon_k$$

On obtient alors pour les transformations :

$$X(2k) = \sum_{n=0}^{\frac{N}{2}-1} \underbrace{\cos\left((4n+1)\frac{\pi k}{2\frac{N}{2}}\right)}_{DCT_{\frac{N}{2}}} \frac{1}{\sqrt{2}} \left[\tilde{x}(n) + \tilde{x}(\frac{N}{2}+n)\right] \tag{2.47}$$

$$X(2k+1) = \sum_{n=0}^{\frac{N}{2}-1} \underbrace{\cos\left((4n+1)\frac{\pi(2k+1)}{2N}\right)}_{E_{\frac{N}{2}}} \frac{1}{\sqrt{2}} \left[\tilde{x}(n) - \tilde{x}(\frac{N}{2}+n)\right] \tag{2.48}$$

Et sous forme matricielle :

$$\begin{array}{rcl} X & = & D'_N \, \tilde{x} \\ \left[\frac{X_{2k}}{X_{2k+1}}\right] & = & \left[\frac{D'_{\frac{N}{2}} U_1}{E_{\frac{N}{2}} U_2}\right] \\ \text{où} \left[\frac{U_1}{U_2}\right] & = & \underbrace{\frac{1}{\sqrt{2}} \left[\begin{array}{c|c} I_{\frac{N}{2}} & I_{\frac{N}{2}} \\ \hline I_{\frac{N}{2}} & -I_{\frac{N}{2}} \end{array}\right]}_{A_N = I_{\frac{N}{2}} \otimes WH_2} \left[\frac{\tilde{x}_1}{\tilde{x}_2}\right] \end{array} \tag{2.49}$$

où $I_{\frac{N}{2}}$ est la matrice identité de taille $\frac{N}{2}$, et WH_2 est la transformée de Walsh-Hadamard de taille 2.

Réalisation des coefficients d'indice pair

D'après les équations (2.47) et (2.49), les coefficients d'indice pair peuvent être calculés récursivement. La DCT est calculée récursivement à l'aide d'une $DCT_{\frac{N}{2}}$ de taille moitié plus petite à chaque pas. A la fin de la récursion, on a $N = 2$ et la matrice A_2 qui est identique à celle de WH_2. De plus, à chaque pas de la récursion, $\tilde{x}$ est transformé en U_1 et U_2 selon (2.49) qui est la même transformation que WH_2.

Cette transformation de Walsh-Hadamard WH_2 peut se factoriser en trois étapes lifting. En utilisant des arrondis au plus proche entiers pour ces étages lifting, il est possible de définir une transformation entière réversible. Ces étages lifting sont :

$$WH_2 = \underbrace{\begin{bmatrix} 1 & 1-\sqrt{2} \\ 0 & 1 \end{bmatrix}}_{U_2} \underbrace{\begin{bmatrix} 1 & 0 \\ \frac{1}{\sqrt{2}} & -1 \end{bmatrix}}_{P_1} \underbrace{\begin{bmatrix} 1 & \sqrt{2}-1 \\ 0 & 1 \end{bmatrix}}_{U_1} \tag{2.50}$$

Réalisation des coefficients d'indice impair

Pour la réalisation des coefficients d'indice impair, il faut d'après (2.49) multiplier U_2 par la matrice de cosinus $E_{\frac{N}{2}}$. Il est montrable [Abh02] [Abh03] que les éléments C_N^{nk} de cette matrice peuvent être écrits sous la forme :

$$C_N^{nk} = \underbrace{(\sqrt{2}\alpha_N \times E_{\frac{N}{2}} \times S_N \times WH_{\frac{N}{2}})}_{O_N^{nk}} \times WH_{\frac{N}{2}} \tag{2.51}$$

où S_N est une matrice de signe.

On a la relation : $WH_N = WH_{\frac{N}{2}} \otimes WH_2$ qui nous permet de dire que les opérations $WH_{\frac{N}{2}}$ dans (2.51) peuvent être réalisées en entier.

Il ne reste plus qu'à définir O_N^{nk} en étages lifting pour que la détermination des coefficients d'indice impair puisse être réalisée en entier.

Les lignes et les colonnes de O_N^{nk} sont permutées en utilisant des matrices de permutation $P1$ et $P2$ de telle sorte que les éléments de O_N^{nk} soient partitionnés en matrices de rotations 2×2. Cette décomposition, d'après [Abh02] [Abh03], s'écrit :

$$\begin{aligned} O_N^{nk} = \quad & P2_N \times \left[diag(R_{\phi_N^n} \otimes I_{\frac{N}{N}})_{n=0,\ldots,\frac{N}{4}-1} \right] \times \ldots \\ & \ldots \times \left[diag(R_{\phi_{16}^n} \otimes I_{\frac{N}{16}})_{n=0,\ldots,3} \right] \\ & \times \left[diag(R_{\phi_8^n} \otimes I_{\frac{N}{8}})_{n=0,1} \right] \times \left[R_{\phi_4^0} \otimes I_{\frac{N}{4}} \right] \\ & \times S_N \times P_N \end{aligned} \tag{2.52}$$

Les permutations de colonnes effectuées par $P1_N$ et de lignes par $P2_N$ peuvent être vues comme les permutations réalisées initialement sur la matrice DCT.

L'angle de base pour cette décomposition est : $\phi_4^0 = \frac{\pi}{8}$, les autres angles étant obtenus par parcours d'un arbre binaire selon ϕ_{2N}^n sur la branche supérieure et $\phi_{2N}^n + \frac{\pi}{4}$ sur la branche inférieure. On aura donc comme angles suivants : $\phi_8^0 = \frac{\pi}{16}$ et $\phi_8^1 = \frac{5\pi}{16}$.

Les matrices de rotations R_ϕ peuvent alors être factorisées en étages lifting selon (2.42).

Structure générale de la IntDCT

Cette transformation peut se représenter sous forme d'un schéma général retraçant les différentes étapes. La transformation inverse sera obtenue par inversion des étapes de récursion et de lifting en changeant leurs signes et leur ordre.

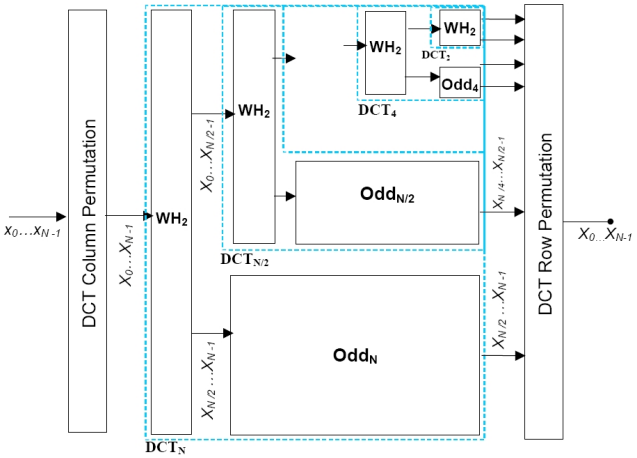


FIG. 2.17 – Structure générale de la transformation IntDCT directe [Abh03]

Pour donner un exemple, on présente ici la transformation IntDCT de taille 8 avec son graphe flots de données :

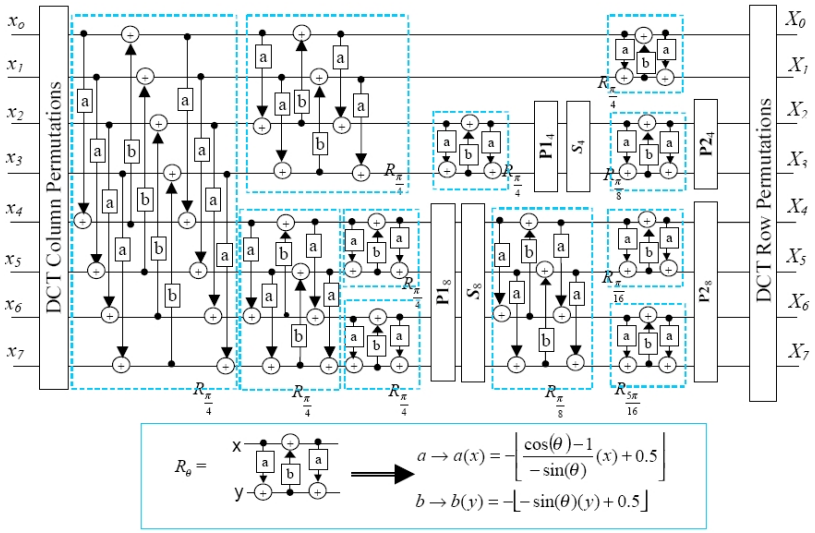


FIG. 2.18 – Graphe flots de données de la IntDCT 8 [Abh03]

Cette IntDCT a été testée comme la BinDCT, dans le cadre du codage d'images fixes et de séquences vidéo. Il en résulte de même que la IntDCT a des performances comparables à la DCT flottante 8×8 de H.263+ et de JPEG. De plus, il a été montré qu'elle est aussi performante que la transformée en ondelettes entières (IWT [Abh03] sur 5 niveaux).

2.4 Les transformées à recouvrement

2.4.1 Introduction

Les transformées à recouvrement ("lapped transforms") ne sont pas d'autres méthodes pour approcher de la DCT flottante 8×8, mais de nouvelles transformées, fournissant ainsi des alternatives à la DCT.

Ces transformées ont déjà été largement étudiées et il en existe plusieurs versions qui ont été notamment introduites par Henrique S. Malvar :

- en 1989 la "Lapped Orthogonal Transform" LOT [Mal89c] pour le codage d'images.
- en 1990 la "Modulated Lapped Transform" MLT [Mal90] pour le codage audio.
- en 1998 [Mal98] [Mal00b] la "Lapped Biorthogonal Transform" LBT et la "Hierarchical Lapped Biorthogonal Transform" HLBT pour le codage d'images et les "Modulated Lapped Biorthogonal Transform" MLBT et "Nonuniform Lapped Biorthogonal Transform" NMLBT pour le codage audio.

Cependant, ces différentes versions de lapped transforms ont aussi été étudiées par d'autres auteurs tels que Trac Tran et T. Nguyen [ITN02] [Tra00b] avec la GenLOT (Generalized linear-phase Lapped Orthogonal Transform) et GLBT (Generalized Lapped Biorthogonal Transform).

Ces transformées existent depuis le début des années 90 (avec la LOT) sans pour autant qu'elles aient une place parmi les transformées courantes telles que la DCT. Ceci est dû au fait que ces transformées augmentent peu les performances de codage, ne justifiant pas l'augmentation importante de complexité. Mais cette remarque était vraie pour la LOT et l'est beaucoup moins pour la LBT. C'est pourquoi récemment beaucoup de travaux ont été réalisés sur cette LBT pour la rendre entière et rapide (c'est-à-dire peu complexe) [ZCC01] [CZL02] [Tra00b].

Nous allons ici présenter la LBT qui est la plus récente et la plus prometteuse de ces transformées en se basant sur la version orthogonale (LOT).

2.4.2 Construction d'une transformée à recouvrement orthogonale LOT

Pour la DCT et pour coder MN échantillons, chacun des M blocs de tailles N est transformé et codé indépendamment. En notation matricielle, cela s'écrit : $y = T^T x$ et pour l'inverse : $x = Ty$ avec x le signal d'entrée, y la sortie et T^T la transposée de la matrice de transformation qui est diagonale en blocs telle que :

$$T = \begin{bmatrix} D & & 0 \\ & D & \\ & & \ddots \\ 0 & & D \end{bmatrix}$$

où D est une matrice d'ordre N définissant la transformation d'un bloc.

La particularité des transformées à recouvrement est qu'elles utilisent plus de N échantillons pour calculer les N coefficients transformés. En effet, elles utilisent L échantillons avec $N < L \leq 2N$. Ces transformées se servent donc de L-N échantillons du voisinage par débordement pour

calculer les N coefficients du bloc courant. Ces transformations peuvent s'écrire sous la même forme que précédemment, mais avec une matrice T différente donnée par :

$$T = \begin{bmatrix} P_1 & & & & 0 \\ & P_0 & & & \\ & & \ddots & & \\ & & & P_0 & \\ 0 & & & & P_2 \end{bmatrix}$$

où P_0 est une matrice L × N, P_1 et P_2 ne sont que des variantes de P_0 afin de gérer les extrémités de l'image.

Pour que cette transformation soit inversible, il faut que la matrice T soit orthogonale. Pour cela, il faut que les colonnes de P_0 soient orthogonales (2.53 (i)) et que les fonctions de recouvrements des blocs voisins le soient aussi (2.53 (ii)) :

$$\begin{cases} (i) & P_0^T P_0 = I \\ (ii) & P_0^T W P_0 = P_0^T W^T P_0 = 0 \end{cases} \tag{2.53}$$

où I est la matrice identité et W est un opérateur de décalage d'ordre N défini par :

$$W = \begin{bmatrix} 0 & I \\ 0 & 0 \end{bmatrix}$$

Pour que la LOT soit optimale, il faut que P_0 soit aussi optimale. Pour cela, cette matrice doit maximiser la compression des données, soit diagonaliser la matrice d'autocorrélation de la sortie selon :

$$R_{yy} = P_0^T R_{xx} P_0 \tag{2.54}$$

Cependant, on peut être amené à utiliser une matrice P particulière, mais qui vérifie tout de même les conditions d'inversibilité (2.53). Dans ce cas, il faut trouver une matrice unitaire Z qui satisfasse : $P_0 = PZ$ et on aura toujours les conditions d'inversibilité selon :

$$\begin{cases} (i) & P_0^T P_0 = Z^T P^T P Z = Z^T Z = I \\ (ii) & P_0^T W P_0 = Z^T P^T W P Z = 0 \end{cases} \tag{2.55}$$

Et l'optimalité avec :

$$R_{yy} = Z^T P^T R_{xx} P Z \tag{2.56}$$

Ce cas permet d'être un peu plus flexible sur le choix de P qui peut être définie à partir de la DCT selon :

$$P = \frac{1}{2} \begin{bmatrix} D_e - D_o & D_e - D_o \\ J(D_e - D_o) & -J(D_e - D_o) \end{bmatrix} \tag{2.57}$$

où D_e et D_o sont les matrices N × N/2 contenant les fonctions DCT pour les échantillons pairs et impairs respectivement, et J est la contre-identité (les 1 sont sur l'antidiagonale).

Dans ce cas, on peut montrer [Mal89c] que Z peut être approchée par :

$$Z = \begin{bmatrix} I & 0 \\ 0 & \tilde{Z} \end{bmatrix} \tag{2.58}$$

où $\tilde{Z}$ est une matrice d'ordre N/2 qui peut elle aussi être approchée par une cascade de N/2-1 rotations selon :

$$\tilde{Z} = T_1 T_2 \dots T_{N/2-1} \quad \text{avec} \quad T_i = \begin{bmatrix} I & 0 & 0 \\ 0 & Y(\alpha) & 0 \\ 0 & 0 & I \end{bmatrix} \tag{2.59}$$

où $Y(\alpha)$ est une rotation comme défini en (2.42) et décomposable en étages lifting selon les figures 2.13 ou 2.14.

Donc, en se basant sur les définitions de $P_0 = PZ$, de P en (2.57), et de Z en (2.58) et (2.59), on peut écrire P_0 sous la forme :

$$P_0 = \frac{1}{2} \begin{bmatrix} D_e & D_o & 0 & 0 \\ 0 & 0 & D_e & D_o \end{bmatrix} \begin{bmatrix} I & I & 0 & 0 \\ I & -I & 0 & 0 \\ 0 & 0 & I & I \\ 0 & 0 & I & -I \end{bmatrix} \begin{bmatrix} 0 & 0 \\ I & I \\ I & -I \\ 0 & 0 \end{bmatrix} \begin{bmatrix} I & 0 \\ 0 & \tilde{Z} \end{bmatrix} \tag{2.60}$$

2.4.3 La transformée à recouvrement biorthogonale LBT

Sur le même schéma que la LOT

Comme on a pu le voir précédemment, la transformation orthogonale nécessite la définition d'une matrice $P_0 = PZ$ orthogonale pour les transformations directe et inverse. Dans le cas biorthogonal, on utilise des matrices différentes pour les transformations directe et inverse.

Pour cela, on utilise P_f pour la transformation directe (forward) et P_b pour la transformation inverse (backward). Chacune de ces matrices prises indépendamment de l'autre ne satisfait pas les conditions d'inversibilité (2.53), mais la biorthogonalité leur permet, si on les considère comme une paire, de satisfaire ces conditions :

$$\begin{cases} (i) & P_f^T P_b = I \\ (ii) & P_f^T W P_b = 0 \end{cases} \tag{2.61}$$

Cependant, l'utilisation de ces deux matrices différentes P_f et P_b nécessite de remettre les données à la même échelle. Cette transformation étant biorthogonale, c'est-à-dire asymétrique, cela revient à multiplier dans le cas direct et diviser dans le cas inverse les coefficients impairs par une matrice :

$$V_f = \begin{bmatrix} \sqrt{2} & & & 0 \\ & 1 & & \\ & & 1 & \\ 0 & & & 1 \end{bmatrix} \quad \text{et} \quad V_b = V_f^{-1} = \begin{bmatrix} \frac{1}{\sqrt{2}} & & & 0 \\ & 1 & & \\ & & 1 & \\ 0 & & & 1 \end{bmatrix}$$

Dans le cas où P_f et P_b sont issues de la DCT (comme P pour la LOT (2.57)), ces matrices s'expriment selon :

$$P_f = \frac{1}{2} \begin{bmatrix} D_e - V_f D_o & D_e - V_f D_o \\ J(D_e - V_f D_o) & -J(D_e - V_f D_o) \end{bmatrix} \tag{2.62}$$

$$\text{et} \quad P_b = \frac{1}{2} \begin{bmatrix} D_e - V_b D_o & D_e - V_b D_o \\ J(D_e - V_b D_o) & -J(D_e - V_b D_o) \end{bmatrix} \tag{2.63}$$

Et l'écriture de P_0 dans (2.60) sera modifiée, les matrices V_f ou V_b selon que l'on soit en transformation directe ou inverse, seront ajoutées au milieu des quatre matrices (c.à.d. entre les deux matrices ne contenant que des I et des 0).

Cette LBT (Lapped Biorthogonal Transform) peut alors être représentée sous forme d'un graphe de flots de données où $c = \sqrt{2}$ pour la transformation directe (issu de V_f) et $c = 1/\sqrt{2}$ pour l'inverse (issu de V_b) :

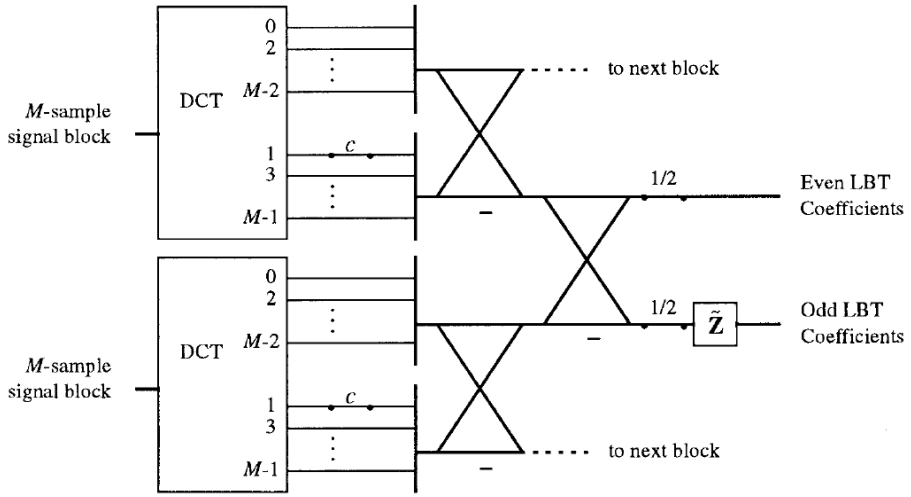


FIG. 2.19 – Graphe flots de données de la LBT [Mal98]

Cette transformée LBT a été testée par rapport à la LOT et à la DCT sur l'image en niveaux de gris "peppers 512×512" par Daniel Veiner [Vei00] de l'Université de Stanford (dans le cadre d'une étude sur les performances de la LOT et de la LBT)

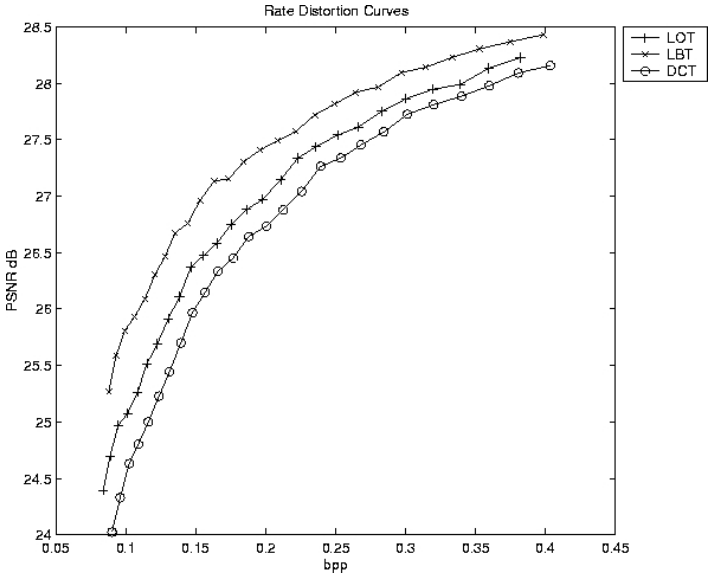


FIG. 2.20 – Courbes débit-distorsion pour l'image "peppers 512×512" [Vei00]

On voit sur ces courbes que la LOT améliore les performances de la DCT d'approximativement 0.5 dB et que la LBT l'améliore de plus de 1.1 dB (en général 1 dB pour de faibles débits).

De plus, il est possible de réaliser cette transformation en étages lifting : la DCT peut être la BinDCT ou la IntDCT, $V_f(0)$ et $V_b(0)$ peuvent être vus comme une rotation décomposable en étages lifting, et $\tilde{Z}$ est une cascade de rotations elles aussi décomposables en étages lifting.

Version en pré- et post-traitements

Un des inconvénients des méthodes de transformées à recouvrement présentées précédemment (LOT et LBT) vient de leurs schémas qui sont du type DCT couplée à un post-traitement, puis un pré-traitement couplé à la DCT inverse (cf fig. 2.21 (a)). Ceci implique que ces méthodes soient complètement dépendantes de la DCT, c'est-à-dire que si l'on veut utiliser une autre transformation (ondelette par exemple), il faut redéfinir les étapes de post- et pré-traitements.

C'est pourquoi Tran et al. ont trouvé une équivalence à ces méthodes (LOT et LBT) dont le schéma est du type pré-traitement couplé à une DCT (ou autre transformation), puis une DCT inverse (ou autre transformation inverse correspondante) couplée à un post-traitement (cf fig. 2.21 (b)). Ce schéma laisse donc une plus grande liberté dans le choix de la transformation, on l'appelle LBT dans le domaine temporel [TT01] [LTT01] [TLT03] [DT03] et plus récemment [SFY06].

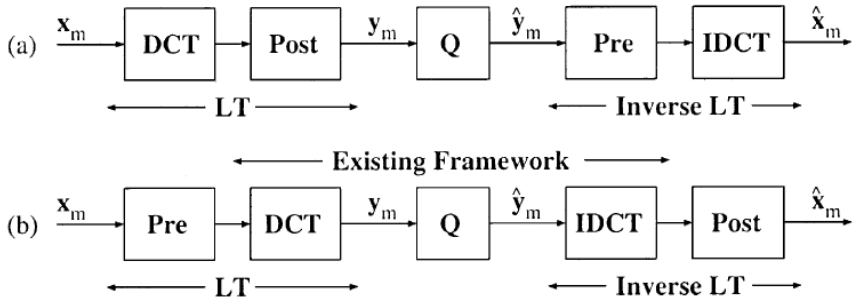


FIG. 2.21 – Schémas de (a) la LBT classique, et (b) la LBT en pré- et post-traitements [TLT03]

Dans l'article [TLT03], Tran et al. montrent l'équivalence entre ces deux schémas en se basant sur la matrice polyphase de la type-II "Fast LOT" [Mal92] :

$$\begin{aligned} E(z) &= \frac{1}{2}\begin{bmatrix} I & 0 \\ 0 & V \end{bmatrix}\begin{bmatrix} I & I \\ I & -I \end{bmatrix}\begin{bmatrix} I & 0 \\ 0 & zI \end{bmatrix}\begin{bmatrix} I & I \\ I & -I \end{bmatrix} C_M^{II} J_M \\ &= G(z) C_M^{II} J_M \end{aligned} \tag{2.64}$$

où $G(z)$ représente le post-traitement appliqué aux coefficients DCT dans lequel la matrice V est arbitraire. Elle peut être $V = DC_{M/2}^{IV} J C_{M/2}^{II^T}$ dans le cas de la type-II "Fast LOT".

Cette matrice polyphase peut, d'après [TLT03], être réécrite sous la forme :

$$\begin{aligned} E(z) &= D_M C_M^{II} \begin{bmatrix} 0 & I \\ zI & 0 \end{bmatrix} \frac{1}{2} \begin{bmatrix} I & J \\ J & -I \end{bmatrix} \begin{bmatrix} I & 0 \\ 0 & V \end{bmatrix} \begin{bmatrix} I & J \\ J & -I \end{bmatrix} \\ &= D_M C_M^{II} \hat{\Lambda}(z) P \end{aligned} \tag{2.65}$$

où D_M est une matrice diagonale alternant les $+1$ et -1, $\hat{\Lambda}(z)$ la matrice $\begin{bmatrix} 0 & I \\ zI & 0 \end{bmatrix}$ et P la matrice représentant le pré-traitement :

$$P = \frac{1}{2} \begin{bmatrix} I & J \\ J & -I \end{bmatrix} \begin{bmatrix} I & 0 \\ 0 & V \end{bmatrix} \begin{bmatrix} I & J \\ J & -I \end{bmatrix} \tag{2.66}$$

où la matrice V est arbitraire, mais peut être $V = JC_{M/2}^{II^T} C_{M/2}^{IV} J$ pour avoir l'équivalence avec la type-II "Fast LOT".

Le schéma correspondant à cette matrice polyphase (2.65) est donné par :

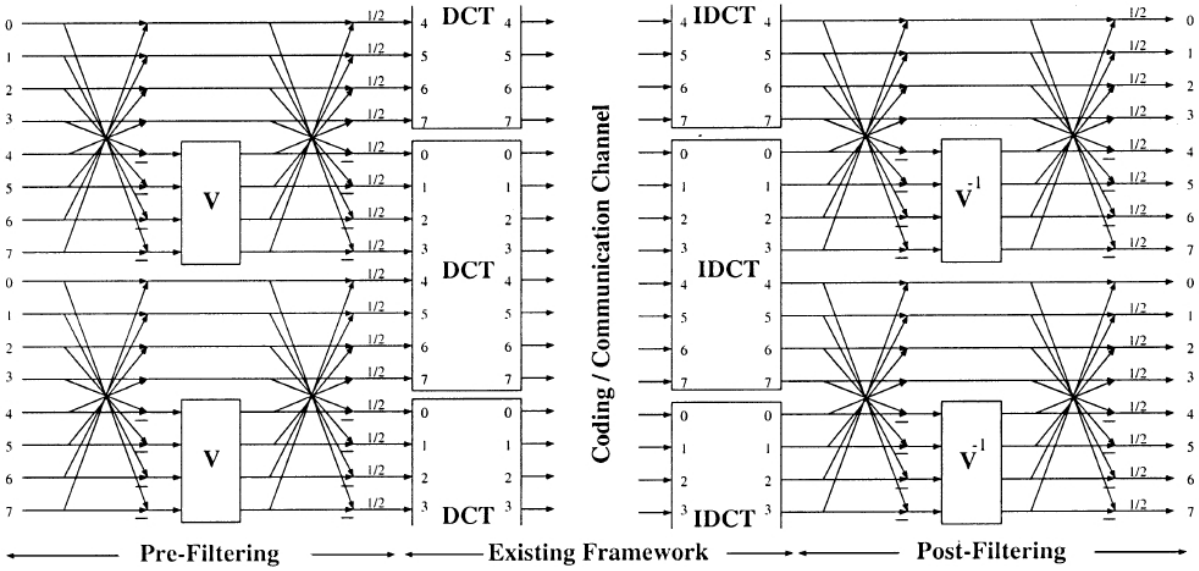


FIG. 2.22 – Schéma de la LBT en pré- et post-traitements [TLT03]

Remarque : La version orthogonale de ce schéma est obtenue en utilisant une matrice V orthogonale, et la version biorthogonale en utilisant une matrice V inversible.

Afin de se rendre compte des effets de cette LBT en pré- et post-traitements, prenons le cas 2×2 ($C_1^{IV} = 1$, $C_1^{II^T} = 1$ et $V = s$) illustré sur la figure 2.23.

En définissant par $\{x_i\}$ les données d'entrée, $\{p_i\}$ la sortie du pré-traitement, $\{\hat{p}_i\}$ l'entrée du post-traitement et $\{\hat{x}_i\}$ les données de sortie, on peut écrire pour le post-traitement :

$$\begin{aligned} \hat{x}_i &= \frac{1}{2}[(\hat{p}_i + \hat{p}_{i+1}) + \frac{1}{s}(\hat{p}_i - \hat{p}_{i+1})] \\ &= \hat{p}_i + \frac{\frac{1}{s} - 1}{2}(\hat{p}_i - \hat{p}_{i+1}) \qquad (2.67) \\ \hat{x}_{i+1} &= \frac{1}{2}[(\hat{p}_i + \hat{p}_{i+1}) - \frac{1}{s}(\hat{p}_i - \hat{p}_{i+1})] \\ &= \hat{p}_{i+1} - \frac{\frac{1}{s} - 1}{2}(\hat{p}_i - \hat{p}_{i+1}) \qquad (2.68) \end{aligned}$$

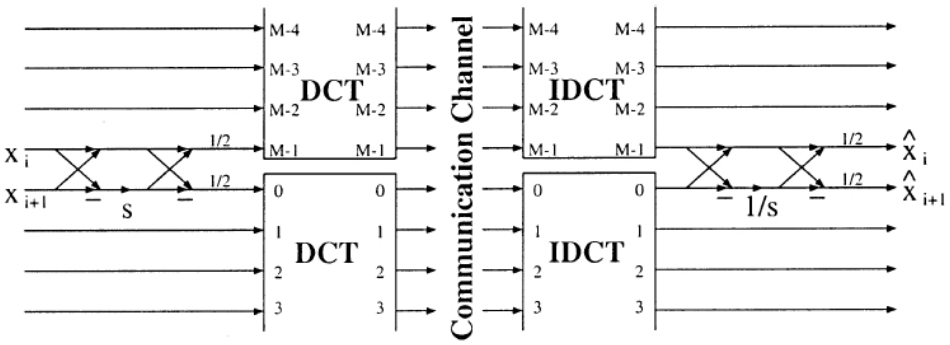


FIG. 2.23 – Schéma de la LBT en pré- et post-traitements sur 2 points [TLT03]

Et pour le pré-traitement :

$$p_i = x_i + \frac{s-1}{2}(x_i - x_{i+1}) \tag{2.69}$$

$$p_{i+1} = x_{i+1} - \frac{s-1}{2}(x_i - x_{i+1}) \tag{2.70}$$

où s doit être supérieur à 1 pour avoir un sens (on pourra prendre par exemple le nombre d'or $\frac{1+\sqrt{5}}{2}$). En effet, le pré-traitement cherche à homogénéiser les données avant la DCT comme illustré sur la figure 2.24, choisir $s < 1$ reviendrait à déplacer les données dans la mauvaise direction.

Cet exemple permet de voir les effets de cette LBT en pré- et post-traitements sur les données avant et après le processus normal de transformation DCT. Ces effets peuvent être représentés par :

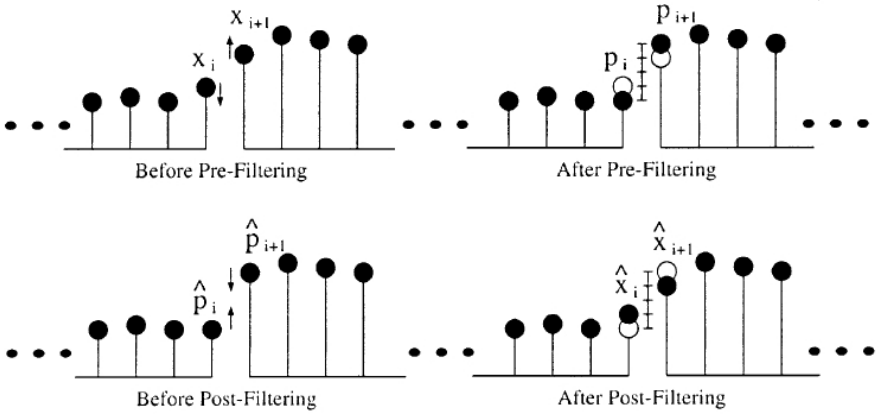


FIG. 2.24 – Effets des pré- et post-traitements sur les données [TLT03]

De plus, il est possible de réaliser une transformation d'entiers à entiers avec une reconstruction parfaite en utilisant des étages lifting dans les pré- et post-traitements du schéma 2.22.

En effet, on peut remplacer chaque "papillon" du pré-traitement P (2.66) par des matrices

non-normalisées de Haar :

$$\begin{aligned} P &= \frac{1}{2}\begin{bmatrix} I & J \\ J & -I \end{bmatrix}\begin{bmatrix} I & 0 \\ 0 & V \end{bmatrix}\begin{bmatrix} I & J \\ J & -I \end{bmatrix} \\ &= \begin{bmatrix} \frac{1}{2}I & J \\ \frac{1}{2}J & -I \end{bmatrix}\begin{bmatrix} I & 0 \\ 0 & V \end{bmatrix}\begin{bmatrix} I & J \\ \frac{1}{2}J & -\frac{1}{2}I \end{bmatrix} \end{aligned} \tag{2.71}$$

Le schéma de la LBT en pré- et post-traitements est alors modifié selon ces étages lifting, il peut être représenté par :

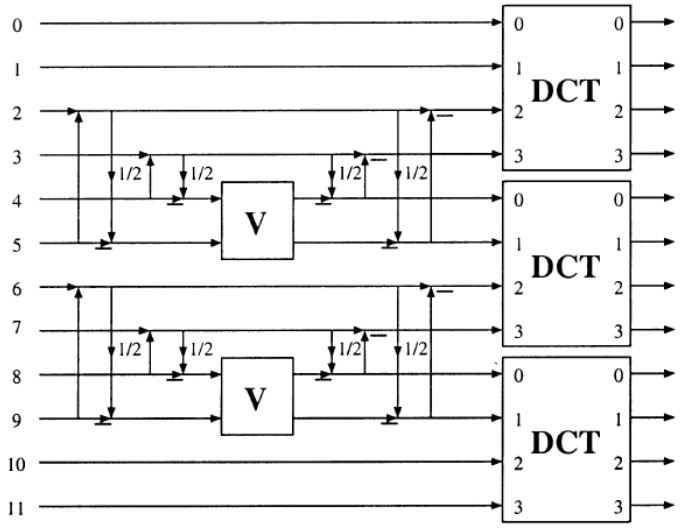


FIG. 2.25 – Schéma de la LBT en pré- et post-traitements sous forme lifting [TLT03]

où V est une matrice de rotation décomposable en étages lifting comme représenté sur les figures 2.13 et 2.14 de la section 2.3.

Cette LBT en pré- et post-traitements a été testée par Tran et al. [DT03] en codage d'images en la comparant à la DCT seule. Quelques résultats sont présentés dans le tableau 2.4 et illustrés sur la figure 2.26.

Compression	DCT	DCT+pre/post	DCT	DCT+pre/post
	Barbara (512×512)		Boat (512×512)	
1 : 8	33.05	**34.11**	36.46	**36.60**
1 : 16	27.78	**29.06**	32.25	**32.67**
1 : 32	24.34	**25.00**	28.13	**28.58**
1 : 64	-	**19.20**	23.66	**23.96**
	Cafe (512×512)		Goldhill (128×128)	
1 : 8	**28.52**	28.50	34.36	**34.64**
1 : 16	24.36	**24.55**	31.45	**31.72**
1 : 32	20.73	**21.00**	28.33	**28.85**
1 : 64	-	**15.59**	24.07	**24.50**

TAB. 2.4 – Résultats de la LBT en pré- et post-traitements comparée à la DCT [DT03]

FIG. 2.26 – Illustration des résultats de la LBT en pré- et post-traitement avec une portion de l'image Café codée à 0.25 bit/pix avec JPEG (originale, DCT seule, DCT+LBT en pré- post-) [DT03]

2.5 Conclusion

Ce chapitre présente plusieurs transformations dont il est possible de faire usage dans un schéma de compression vidéo en remplacement de la transformation DCT classique.

Les ondelettes de première génération sont une alternative scalable ou échelonnable à cette DCT. De plus, elles peuvent être construites de plusieurs manières pour différents types d'applications.
Les ondelettes peuvent être continues ou discrètes. Elles sont alors définies à partir de la dilatation et de la translation d'une ondelette mère, fonction continue possédant au moins un moment nul. Dans le cas continu, la dilatation et la translation sont à valeurs réelles, et dans le cas discret, elles sont limitées à une grille dyadique.
Ces ondelettes peuvent aussi être définies à partir de l'analyse multirésolution qui consiste à décomposer les signaux sur une gamme étendue d'échelles. Cette analyse peut être réalisée par des bancs de filtres ou par un schéma dit lifting. L'analyse par bancs de filtres réalise une succession de filtrages passe-haut et passe-bas des bandes basses fréquences des signaux. Le schéma lifting est équivalent à ces bancs de filtres, mais ne nécessite qu'une succession d'étapes de prédiction et de mise à jour.
Ces différentes constructions des ondelettes de première génération peuvent ensuite être étendues au cas à deux dimensions utiles ici pour décomposer des images ou des vidéos.

La plupart des schémas de codage vidéo normalisé utilise une transformation DCT calculée en nombres flottants aussi bien au codage qu'au décodage. Ceci entraîne une divergence entre le codeur et le décodeur. Il est alors devenu nécessaire de définir une version de cette DCT évitant ces problèmes d'arrondis : les DCT en lifting.
La BinDCT est une approximation de la DCT flottante classique. Elle est réalisée à l'aide d'un schéma lifting (semblable à ceux des ondelettes) qui ne nécessite que des opérations binaires évitant ainsi l'utilisation d'arrondi.
La DCT entière IntDCT est proche de la BinDCT, mais le schéma lifting associé ne nécessite que des opérations entières. Pour cela, l'IntDCT utilise une autre factorisation de la transformée DCT classique que la BinDCT. Cette IntDCT reste donc aussi une approximation de la DCT flottante classique. La factorisation employée permet, de plus, d'utiliser une technique récursive de calculs entiers permettant d'obtenir la DCT de taille N recherchée à partir de calculs de DCT de taille $N/2$.

Une autre possibilité consiste à utiliser des transformées à recouvrement, non pas cette fois-ci pour trouver une alternative à la DCT, mais pour en améliorer les performances. Ces transformées sont alors un complément à la DCT.
Les transformées à recouvrement peuvent être orthogonales, appelées LOT ou biorthogonales, appelées LBT. Ces deux versions se basent sur la DCT flottante classique et sont appliquées dans le domaine transformé. Elles consistent, au codage, à réaliser les transformées DCT classiques, puis à post-traiter les coefficients de blocs voisins. Au décodage, un pré-traitement des données est effectué avant l'étape de transformation DCT inverse.
Plus récemment, une version biorthogonale de ces transformées à recouvrement appliquée dans le domaine temporel a été définie. Cette LBT devient alors indépendante de la transformation fréquentielle appliquée, la seule contrainte étant que celle-ci doit être basée blocs. La LBT tem-

porelle consiste alors, au codage, à pré-traiter les informations entre plusieurs blocs avant de leur appliquer l'étape de transformation. Au décodage, un post-traitement est appliqué aux informations après la transformation inverse.

Ce chapitre montre qu'il existe beaucoup d'autres transformations que la DCT flottante classique pour les schémas de compression vidéo. Ces transformations sont toutes aussi performantes, et elles peuvent être :

- des alternatives à la DCT classique comme les ondelettes de première génération.
- des améliorations de la DCT classique comme les DCT en lifting, la BinDCT et l'IntDCT.
- des compléments à la DCT classique comme les transformées à recouvrement.

Cependant, toutes ces transformations ne permettent pas de représenter efficacement les structures géométriques contenues dans les images telles que les contours. Il est alors intéressant d'étudier les transformations qui ont cette capacité.

Chapitre 3

L'exploitation des orientations dans les transformées

3.1 Introduction

Le chapitre précédent a présenté et évalué différentes transformations nécessaires à un codeur vidéo afin de décorréler spatialement au maximum les informations contenues dans les images. Cela a permis de montrer qu'il existe des alternatives à la transformée DCT utilisée dans les schémas normalisés.

Les transformées en ondelettes de première génération réalisent une décomposition multirésolution des images leur conférant ainsi des propriétés naturelles de scalabilité ou d'échelonnabilité manquantes à la DCT.
Les versions lifting de la DCT comme la BinDCT et l'IntDCT permettent de réaliser le calcul de la DCT très rapidement et d'éviter la divergence entre le codeur et le décodeur en n'utilisant que des opérations réversibles binaires ou entières respectivement.
Les transformées à recouvrement exploitent quant à elles les corrélations qui peuvent exister entre les blocs traités par une DCT afin d'en améliorer les performances. Elles sont plutôt à voir comme un complément à cette DCT que comme une transformation alternative.

Cependant, ces transformations généralement appliquées suivant les lignes puis les colonnes ne sont pas bien adaptées pour représenter efficacement les structures géométriques contenues dans les images telles que les contours.

Il est alors nécessaire de pousser plus amont dans ces transformations afin d'en isoler celles capables d'effectuer une représentation efficace de ces contours.

L'état de l'art sur ces transformations adaptées au contenu est notamment porté par les ondelettes appelées de seconde génération telles que les *ridgelets*, les *curvelets*, les *contourlets*, les bandelettes de première et seconde génération, et les *directionlets*.
Mais, il existe d'autres types de transformations adaptées aux structures géométriques basées sur des approches de type DCT : la DCT adaptée aux contours (SA-DCT "Shape-Adaptive DCT"), la DCT directionnelle et la BinDCT orientée.

Ce chapitre aura pour vocation de présenter et d'évaluer ces deux catégories de transformations adaptées aux structures géométriques des images : les ondelettes de seconde génération et les DCT exploitant les orientations.

3.2 Les ondelettes de seconde génération

La section 2.2 présentait les ondelettes dites de première génération. Ces ondelettes respectent certaines propriétés (P) telles que l'orthogonalité ou la biorthogonalité, la localisation spatiale et fréquentielle, l'adaptablilité à l'analyse multirésolution...

Cependant, ces ondelettes sont relativement contraintes. Les trois principales contraintes (C) identifiées par Sweldens [Swe97a] sont :

1. Les ondelettes de première génération définissent des bases de $\mathbb{R}^n$, elles auraient besoin d'être définies sur un domaine arbitraire de $\mathbb{R}^n$ comme appliquées le long d'une courbe, d'une surface...
2. Les ondelettes de première génération ne fournissent des bases que pour les espaces ayant une mesure invariante par translation (Haar, Lebesgue). Cependant, une analyse le long d'une courbe ou d'une surface et les approximations pondérées nécessitent des bases régies par une mesure pondérée.
3. Les ondelettes de première génération impliquent un échantillonnage régulier des données, mais elles devraient pouvoir s'adapter aux échantillonnages irréguliers.

Les ondelettes vérifiant les premières propriétés (P) et contournant les trois contraintes (C) sont appelées ondelettes de seconde génération.

Ces ondelettes de seconde génération peuvent donc être appliquées le long de courbes, à un échantillonnage irrégulier ou utiliser une mesure pondérée. Mais elles doivent toujours vérifier les propriétés (P), elles peuvent donc être modélisées par un schéma lifting [DS98], schéma qui est devenu rapidement à valeurs entières [CDSY98] depuis l'apparition de ces ondelettes.

Plusieurs de ces ondelettes dites de seconde génération vont être présentées. Celles-ci utilisent toutes avantageusement les structures géométriques contenues dans les images, en s'appliquant le long des contours par exemple.

3.2.1 Les *curvelets* et les *contourlets*

Les ondelettes dites de première génération permettent de représenter efficacement les singularités locales des signaux. Cependant, il en existe beaucoup d'autres telles que les singularités le long de droites ou de courbes, singularités que ces ondelettes ne représentent pas efficacement.

Afin d'éviter ce problème, les *curvelets* [CD00b] et les *contourlets* [DV01b] ont été mises au point. En effet, elles sont des premières ondelettes dites de seconde génération à utiliser des courbes ou des contours comme support.

Ces deux ondelettes sont très similaires. En effet, il s'agit de la même transformation sauf que les *curvelets* sont définies dans le cas continu et les *contourlets* dans le cas discret.

Pour pouvoir définir les *curvelets* et les *contourlets*, nous devons tout d'abord définir une autre ondelette de seconde génération : la transformée *ridgelet* [CD99] sur laquelle ces deux ondelettes sont basées.

Les *ridgelets*

Les *ridgelets* sont des ondelettes de seconde génération qui permettent de représenter efficacement les singularités présentes le long de droites.

Elles ont tout d'abord été définies dans le cas continu par Candès et Donoho [Can98] [CD99], mais les applications pratiques généralement discrètes ont poussé Do et Vetterli à en donner une version discrétisée [DV00] [DV03].

La transformation *ridgelet* continue [Can98] [CD99] est une fonction 2-D de $\mathbb{R}^2$. On a vu dans l'équation (2.5) la définition d'une ondelette 1-D de première génération, de même on peut définir une ondelette de première génération dans le cas 2-D en réalisant un produit de tenseurs entre deux ondelettes 1-D :

$$\psi_{a1,a2,b1,b2}(\mathbf{t}) \;=\; \psi_{a1,b1}(t1)\,\psi_{a2,b2}(t2) \tag{3.1}$$

La transformation en ondelettes associée à cette ondelette 2-D s'écrit alors :

$$W_f^2(a1,a2,b1,b2) \;=\; \int_{\mathbb{R}^2} \psi_{a1,a2,b1,b2}(\mathbf{t})\, f(\mathbf{t})\, d\mathbf{t} \tag{3.2}$$

Cette transformation en ondelettes est locale ((a1,b1), (a2,b2)), cependant la transformation *ridgelet* continue que nous cherchons à caractériser doit avoir pour support une droite que l'on peut définir par :

$$t1\cos(\theta) \;+\; t2\sin(\theta) \;=\; \textit{constante} \tag{3.3}$$

La transformation *ridgelet* continue associée est alors définie par :

$$RI_f^2(a,b,\theta) \;=\; \int_{\mathbb{R}^2} \psi_{a,b,\theta}(\mathbf{t})\, f(\mathbf{t})\, d\mathbf{t} \tag{3.4}$$

où les *ridgelets* 2-D $\psi_{a,b,\theta}(\mathbf{t})$ sont définies à partir d'une fonction 1-D $\psi(t)$:

$$\psi_{a,b,\theta}(\mathbf{t}) \;=\; \frac{1}{\sqrt{a}}\,\psi\Big(\frac{t1\cos(\theta)+t2\sin(\theta)-b}{a}\Big) \tag{3.5}$$

Les *ridgelets* qui ont pour support des droites permettent donc de représenter efficacement les contours rectilignes.

Les *curvelets*

Comme on a pu le voir précédemment, les *ridgelets* représentent efficacement les objets lisses à contours droits, cependant les contours des objets réels d'une image naturelle sont rarement droits, ils sont plus généralement courbes (cf fig. 3.1).

En se basant sur le fait que les contours courbes peuvent être vus comme droits à une échelle suffisamment fine, Candès et Donoho ont envisagé les *curvelets* [CD00a] [CD00b] pour mieux représenter ces contours courbes que les ondelettes de première génération, comme illustré sur la figure 3.1.

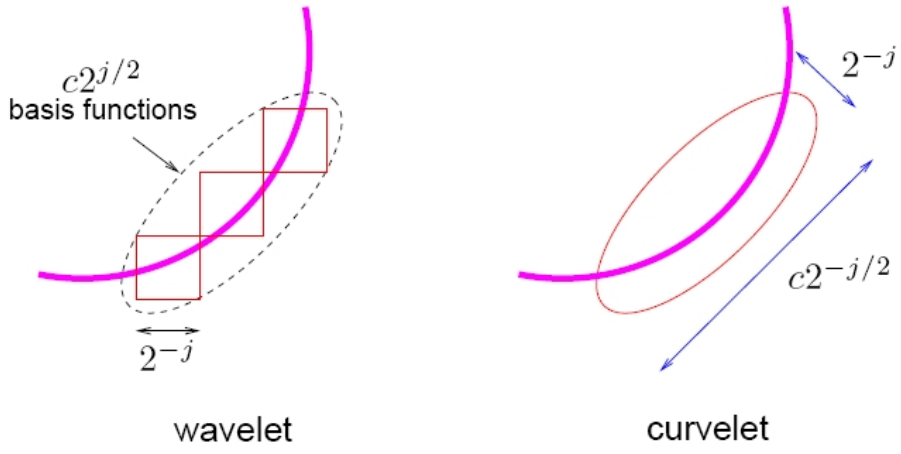


FIG. 3.1 – Exemple d'analyse ondelettes et *curvelets* d'un contour courbe [DV01a]

En suivant cette remarque sur les contours, la décomposition *curvelet* vient alors naturellement et peut être décrite par :

1. Décomposition en sous-bandes de l'image réalisée par filtrage spatial passe-bande.
2. Fenêtrage de chaque sous-bande en blocs dont la taille dépend de l'échelle de la sous-bande.
3. Application de la transformée *ridgelet* sur ces blocs, le nombre d'orientations possibles pour ces *ridgelets* est alors proportionnel à l'échelle (cf fig. 3.2).

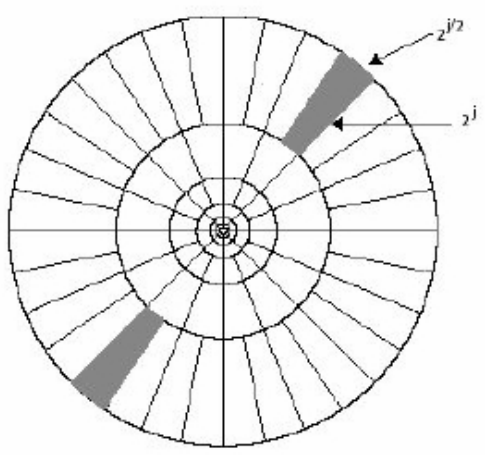


FIG. 3.2 – Les orientations possibles de la décomposition *curvelet* [CD02]

Pour illustrer cette décomposition, un exemple est donné par la suite sur la figure 3.4.

Ces *curvelets* vérifient plusieurs propriétés, telles qu'une structure pyramidale due à la décomposition en sous-bandes, et l'anisotropie, en vérifiant :

$$largeur \approx longueur^2$$

Ceci permet aux *curvelets* de bien s'adapter aux contours, leurs coefficients sont grands quand elles sont alignées avec le contour et très faibles dans le cas contraire, comme représenté sur la figure 3.3.

FIG. 3.3 – Taille des *curvelets*

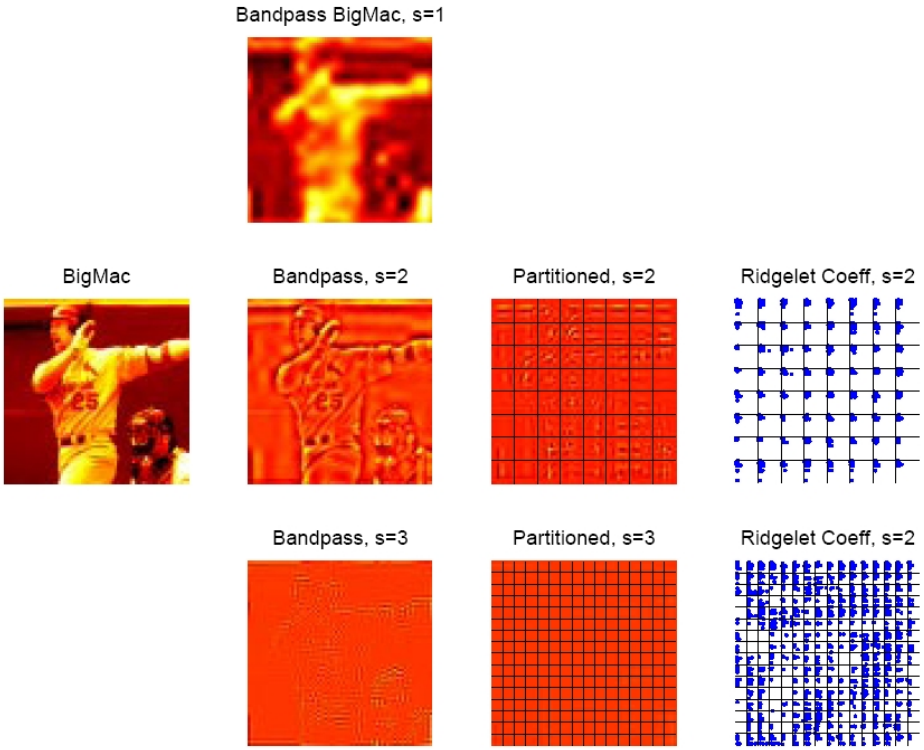


FIG. 3.4 – Exemple de décomposition *curvelet* [CD00a]

Des travaux complémentaires sur les *curvelets* ont été réalisés par la suite pour les rendre plus robustes [CD02], les discrétiser [CDDY06], et pour réaliser du débruitage [SCD02], ou du codage d'images [WZVS06].

Les *contourlets*

Par la suite, Do et Vetterli ont réalisé un rapprochement entre les *curvelets* et les bancs de filtres directionnels [DV01b], afin de rendre cette transformation discrète, et ainsi ont créé les *contourlets* [DV01a] [DV05] [BG05] (cf fig. 3.5).

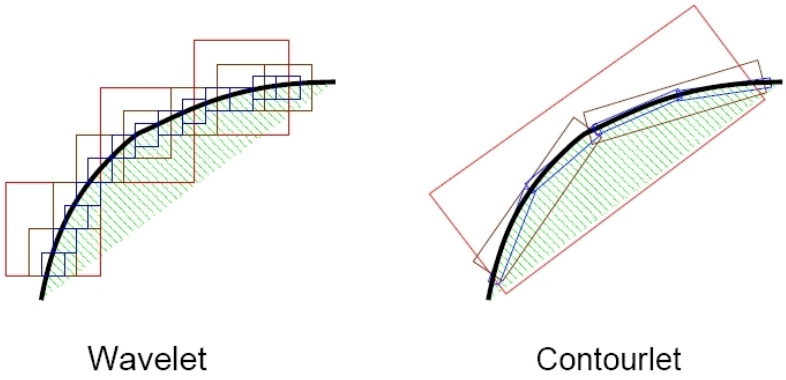


FIG. 3.5 – Exemple d'analyse ondelettes et *contourlets* d'un contour courbe [DV05]

Afin de réaliser l'opération 1, de décomposition en sous-bandes, une pyramide Laplacienne (LP) peut être utilisée. En effet, cette décomposition pyramidale Laplacienne génère à chaque niveau une version sous-échantillonnée et passe-bas de l'original, et la différence entre l'image originale et cette prédiction, la version passe-bande. Cependant, cette pyramide Laplacienne introduit un facteur de redondance de 4/3 mal adapté à la compression.

Les deux autres opérations sont alors réalisées à l'aide d'un banc de filtres directionnels (DFB) [BS92] qui est un ensemble de filtres ayant les mêmes caractéristiques fréquentielles, mais appliquées selon des directions différentes. Ce type de banc de filtres a été conçu pour capturer les hautes fréquences qui représentent la directionnalité des images. Les basses fréquences sont mal représentées par cette décomposition, mais l'association avec une pyramide Laplacienne permet de les isoler dans les images passe-bas. On nomme alors cette décomposition par banc de filtres directionnels pyramidaux (PDFB) les *contourlets* (cf fig. 3.6).

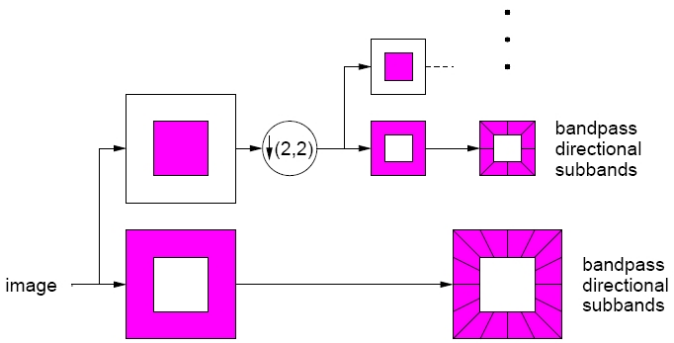


FIG. 3.6 – La décomposition *contourlet* par pyramide Laplacienne LP et banc de filtres directionnels DFB [DV05]

Les différentes directions du banc de filtres utilisé pour cette décomposition dépendent alors de l'échelle (ou de la fréquence) d'analyse, comme illustré sur la figure 3.7 :

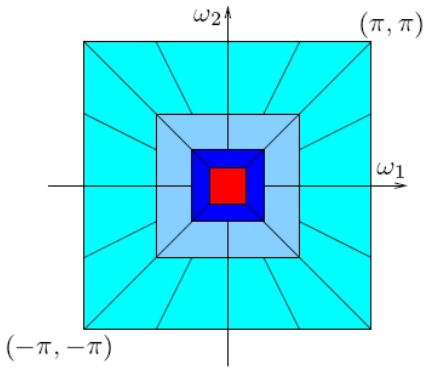


FIG. 3.7 – Les directions *contourlets* en fonction de la fréquence [DV01a]

Il est important de noter que si la décomposition pyramidale Laplacienne et le banc de filtres directionnel permettent tous deux une reconstruction parfaite, alors la transformation *contourlet* discrète est à reconstruction parfaite aussi.

3.2.2 Les bandelettes

Plus récemment, Mallat et al. ont mis au point une nouvelle transformation ondelette dite de seconde génération qui utilise avantageusement les structures géométriques contenues dans les images, les bandelettes [LPM05] [PM05a]. Cette transformation a été définie dans deux versions appelées bandelettes de première et de seconde génération.

Les bandelettes de première génération

Ces premières bandelettes [LPM00] [LPM04] [LPM05] permettent de représenter la géométrie des images en utilisant des flots géométriques au lieu des contours. Un flot géométrique est un champ de vecteurs $\overrightarrow{\tau}$ qui indique la direction où l'image a localement des variations régulières comme illustré sur la figure 3.8.

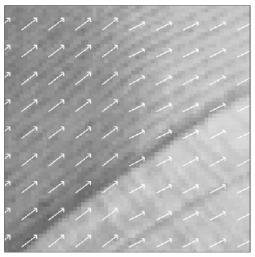


FIG. 3.8 – Flot géométrique dans une région avec le champ de vecteurs associé [LPM05]

Les bandelettes sont construites à partir d'ondelettes bidimensionnelles déformées le long du flot géométrique. Elles définissent alors des bases orthonormées de bandelettes pour chaque région de l'image où ces flots géométriques restent parallèles ou sont non définis. Il y a donc trois types de régions :

- celles où l'image est uniformément régulière, et où il n'y a donc pas de flot, on la décompose alors avec une base d'ondelettes bidimensionnelles classique.
- celles où le flot géométrique est parallèle verticalement $\overrightarrow{\tau}(x_1)$ (c.à.d. à dominante verticale ($> 45^o$), cas du troisième bloc de la figure 3.9).
- et celles où le flot géométrique est parallèle horizontalement $\overrightarrow{\tau}(x_2)$ (c.à.d. à dominante horizontale ($< 45^o$), cas des premier et deuxième blocs de la figure 3.9).

La première étape de la représentation en bandelettes consiste donc à isoler ces régions de flots géométriques parallèles. Cette segmentation est réalisée à l'aide d'une décomposition en arbre quaternaire en carrés dyadiques. Si le bloc (au début toute l'image) contient plusieurs types de flots géométriques, alors on le découpe en quatre parties égales et on recommence le même traitement pour chacun des blocs créés, sinon il définit une nouvelle région Ω_i. Un exemple illustre cette segmentation sur la figure 3.9 :

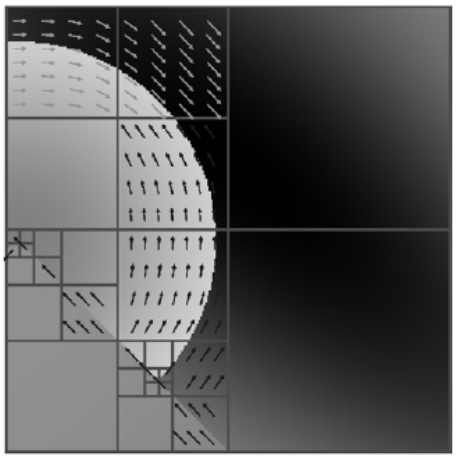


FIG. 3.9 – Exemple de segmentation en carrés dyadiques d'une image avec ses flots géométriques [LPM05]

Par la suite, dans chaque région Ω_i, la transformée bandelettes réalise :

1. un ré-échantillonnage qui calcule les valeurs de l'image le long du flot géométrique. Il est réalisé ici par une interpolation par splines cubiques,
2. une transformation en ondelettes déformées, effectuée par un filtrage en sous-bandes le long du flot. Ce filtrage traverse les frontières des blocs pour éviter les problèmes de discontinuités à ces frontières,
3. une "bandeletisation" qui transforme les coefficients des ondelettes déformées en coefficients bandelettes le long du flot.

Dans l'étape 2, si la région Ω_i est régulière (sans flot géométrique) alors la décomposition est effectuée sur une base d'ondelettes bidimensionnelles classique à l'aide d'un banc de filtres

biorthogonaux 9/7 de Daubechies [CDF92]. Une base d'ondelettes séparables discrète est définie par son ondelette mère ψ et sa fonction d'échelle associée ϕ, elle s'écrit :

$$\left\{ \begin{array}{l} \psi_{j,m_1}[n_1]\phi_{j,m_2}[n_2], \\ \phi_{j,m_1}[n_1]\psi_{j,m_2}[n_2], \\ \psi_{j,m_1}[n_1]\psi_{j,m_2}[n_2] \end{array} \right\}_{j,m_1,m_2} \tag{3.6}$$

Dans le cas contraire, si la région Ω_i possède un flot géométrique $\overrightarrow{\tau}$, alors la décomposition est effectuée sur une base d'ondelettes déformées par une déformation $W(\overrightarrow{\tau})$. On a vu précédemment que ce flot peut être soit parallèle horizontalement soit parallèle verticalement, on peut alors définir ce flot comme étant :

$$\left\{ \begin{array}{ll} \overrightarrow{\tau_i}[n_1, n_2] = \overrightarrow{\tau_i}[n_2] = (c_i[n_2], 1) & \text{pour un flot parallèle horizontalement} \\ \overrightarrow{\tau_i}[n_1, n_2] = \overrightarrow{\tau_i}[n_1] = (1, c_i[n_1]) & \text{pour un flot parallèle verticalement} \end{array} \right. \tag{3.7}$$

où $c_i[n_j]$ représente la courbe formée par le flot géométrique.

Ce flot ainsi décomposé permet de définir la base d'ondelettes déformées nécessaire à cette décomposition comme étant :

$$\left\{ \begin{array}{l} \psi_{j,m_1}[n_1 - c_i[n_2]]\phi_{j,m_2}[n_2], \\ \phi_{j,m_1}[n_1 - c_i[n_2]]\psi_{j,m_2}[n_2], \\ \psi_{j,m_1}[n_1 - c_i[n_2]]\psi_{j,m_2}[n_2] \end{array} \right\}_{j,m_1,m_2} \tag{3.8}$$

dans le cas où le flot géométrique est parallèle horizontalement, et dans le cas parallèle verticalement on a :

$$\left\{ \begin{array}{l} \psi_{j,m_1}[n_1]\phi_{j,m_2}[n_2 - c_i[n_1]], \\ \phi_{j,m_1}[n_1]\psi_{j,m_2}[n_2 - c_i[n_1]], \\ \psi_{j,m_1}[n_1]\psi_{j,m_2}[n_2 - c_i[n_1]] \end{array} \right\}_{j,m_1,m_2} \tag{3.9}$$

Il est intéressant de remarquer que la décomposition d'une image $f[n_1, n_2]$ sur cette base d'ondelettes déformées est équivalente à déformer au préalable avec $W(\overrightarrow{\tau})$ cette image afin de la redresser vers l'horizontale ou la verticale et à la décomposer sur une base d'ondelettes classiques (3.6), comme représenté sur la figure 3.10.

$$\langle f[n_1, n_2], \Psi[n_1 - c_i[n_2], n_2]\rangle = \langle f[n_1 + c_i[n_2], n_2], \Psi[n_1, n_2]\rangle \text{dans le cas parallèle horiz.} \tag{3.10}$$

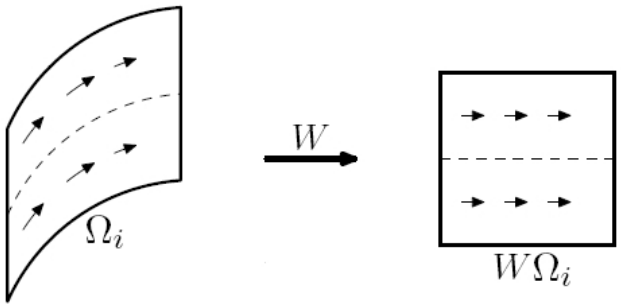


FIG. 3.10 – Exemple de déformation W d'une région Ω_i [LPM04]

Le banc de filtres alors utilisé pour cette décomposition sur la base d'ondelettes (3.6) est un banc 9/7 de Daubechies [CDF92], mais adapté pour filtrer à travers les frontières de régions, comme illustré sur la figure 3.11.

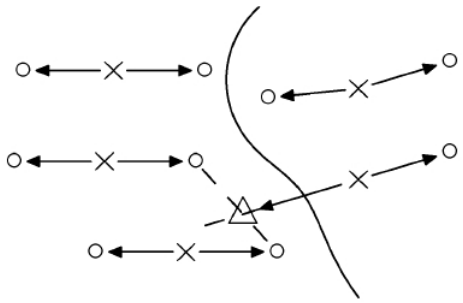


FIG. 3.11 – Le point représenté par un triangle est calculé par interpolation de ses trois voisins et est utilisé par la région de droite pour son filtrage [LPM05]

L'étape 3 est appelée "bandeletisation". Elle exploite la régularité de la fonction le long du flot géométrique, en remplaçant les ondelettes déformées :

$$\{\psi_{j,m_1}[n_1 - c_i[n_2]]\phi_{j,m_2}[n_2]\}_{j,m_1,m_2} \qquad \text{dans le cas parallèle horizontalement} \qquad (3.11)$$

par une famille de fonctions engendrant le même espace :

$$\{\psi_{j,m_1}[n_1 - c_i[n_2]]\psi_{l,m_2}[n_2]\}_{j,l>j,m_1,m_2} \qquad \text{dans le cas parallèle horizontalement} \qquad (3.12)$$

Ces fonctions sont appelées bandelettes, de par leurs supports allongés le long des lignes de flots géométriques (cf fig. 3.12).
Cette opération de "bandeletisation" est réalisée par transformées monodimensionnelles discrètes.

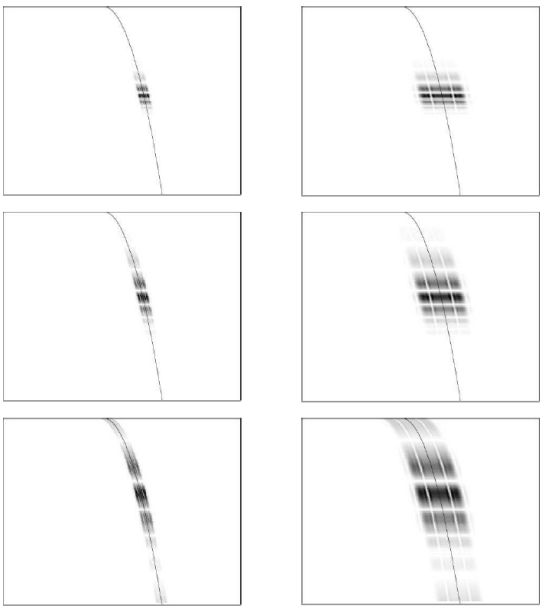


FIG. 3.12 – Exemples de bandelettes le long d'une courbe [LPM01]

La base orthonormée de bandelettes de la région Ω_i est donc définie par :

$$\left\{ \begin{array}{l} \psi_{j,m_1}[n_1 - c_i[n_2]]\psi_{l,m_2}[n_2], \\ \phi_{j,m_1}[n_1 - c_i[n_2]]\psi_{j,m_2}[n_2], \\ \psi_{j,m_1}[n_1 - c_i[n_2]]\psi_{j,m_2}[n_2] \end{array} \right\}_{j,l>j,m_1,m_2} \quad \text{dans le cas parallèle horizontalement} \qquad (3.13)$$

Quelques résultats de ces bandelettes sont donnés sur la figure 3.13.

FIG. 3.13 – Résultats de bandelettes : (a) Lena, (b) détail de Lena, (c) détail de Barbara. Lena est à un PSNR de 33.04 dB en bandelettes contre 32,55 dB en ondelettes. Barbara a un PSNR de 31,22 dB en bandelettes contre 28.50 dB en ondelettes [LPM03]

Les bandelettes de seconde génération

Les bandelettes de première génération décrites précédemment utilisent avantageusement les structures géométriques des images. Cependant, elles ne sont pas directement définies dans le cas discret, et elles ne proposent pas de représentation multirésolution de la géométrie. C'est pourquoi Mallat et al. ont défini les bandelettes de seconde génération [PM05a], [PM05b].

La construction de ces bandelettes est différente de celle des bandelettes de première génération, puisqu'elles permettent une représentation multirésolution de la géométrie. Cette construction correspond à appliquer une transformée géométrique orthogonale aux coefficients d'ondelettes de l'image, au préalable représentée sur une base d'ondelettes classiques.

La transformation bandelette se déroule donc comme indiqué sur la figure 3.14, soit :

1. Une transformation en ondelettes classiques orthogonale ou biorthogonale de l'image f.
2. Dans chaque sous-bande, on effectue une segmentation hiérarchique en carrés dyadiques au sens de la meilleure représentation de la géométrie.
3. Par la suite, dans chaque carré dyadique, on recherche la meilleure géométrie définissant la directionnalité.
4. On réalise une projection 1D orthogonale à la géométrie déterminée définissant ainsi un signal discret 1D f_d.
5. Une transformation 1D discrète en ondelettes du signal 1D f_d donnant les coefficients bandelettes b_k.

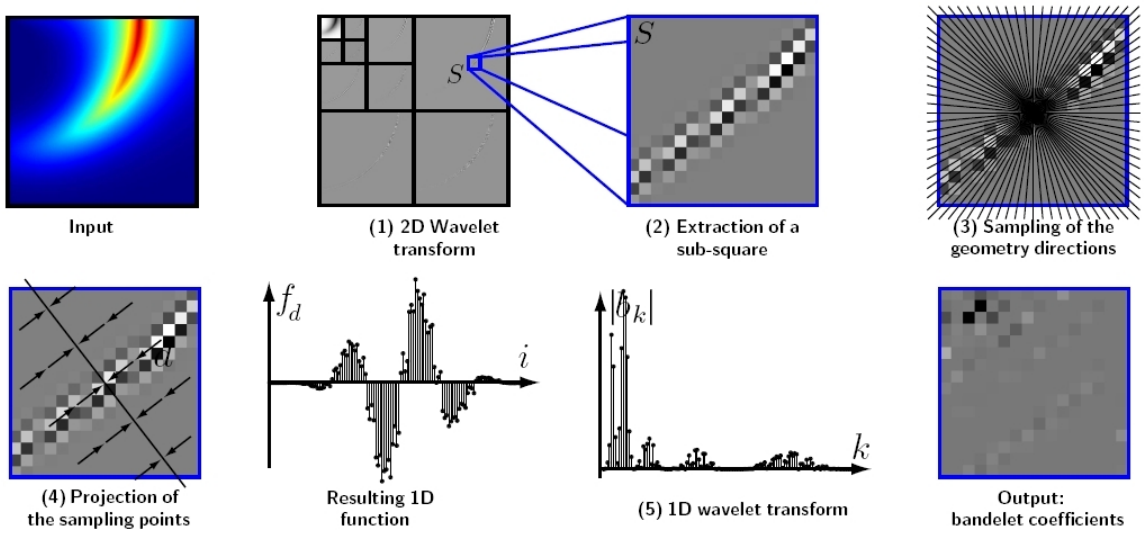


FIG. 3.14 – Description de la transformation bandelette de seconde génération [PM05b]

Les étapes 2 et 3 correspondent aux mêmes étapes que pour les bandelettes de première génération [LPM04].

Les étapes 4 et 5 correspondent à la "bandeletisation".
La première manipulation consiste à projeter les points x de notre région sur un axe d perpendiculaire à la direction de la géométrie déterminée à l'étape précédente. On obtient ainsi les points

$\tilde{x}$. Afin de construire un signal 1D, on réordonne ces points $\tilde{x}$ suivant leur valeur le long de l'axe d, comme illustré sur la figure 3.15. On obtient alors le signal 1D discret f_d défini par :

$$\forall i, \qquad f_d[i] = f(x_i) \tag{3.14}$$

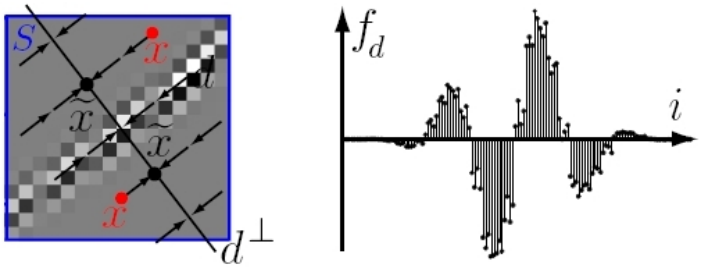


FIG. 3.15 – Exemple de projection orthogonale 1D [PM05b]

La seconde partie de cette "bandeletisation" correspond à une transformation 1D en ondelettes de première génération comme pour l'étape 3 de "bandeletisation" des bandelettes de première génération.

Les coefficients bandelettes b_k sont ainsi générés, et ils dépendent d'une base de bandelettes $\mathcal{B} = \{b_n\}$, où les b_n sont des fonctions bandelettes qui dépendent de l'échelle 2^j de la première transformée, du carré dyadique étudié S de largeur L, et de l'échelle 2^k de la transformée 1D. Ces fonctions ont alors une largeur de $2^j L$ et une longueur de 2^{j+k}. Il est intéressant de noter que ces fonctions se chevauchent, mais pas les carrés dyadiques, ne posant ainsi pas de problème d'effets de blocs.

Quelques résultats de décomposition par ces bandelettes sont donnés sur la figure 3.16.

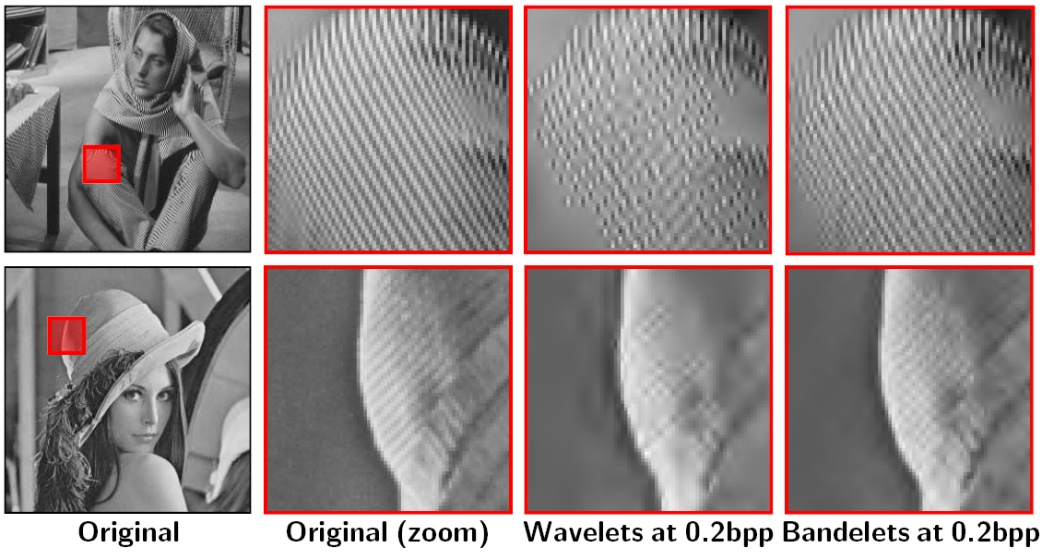


FIG. 3.16 – Comparaison entre les ondelettes et les bandelettes de seconde génération à 0.2bpp [PM05a]

3.2.3 Les *directionlets*

Très récemment, Vetterli et al. ont mis au point une nouvelle transformée exploitant la directionnalité contenue dans les images : les *directionlets* [VBLVD06] [VBLVD05] [VBLVD07] [VBLV07].

Les ondelettes de première génération ne représentent pas efficacement les discontinuités 1D telles que les contours. En effet, ces discontinuités sont très anisotropiques (c.à.d. directionnelles), alors que cette transformation en ondelettes est isotropique puisque verticale et horizontale (c.à.d. non directionnelles). Les *directionlets* sont donc une transformation basée sur des fonctions anisotropiques comme illustré sur la figure 3.17.

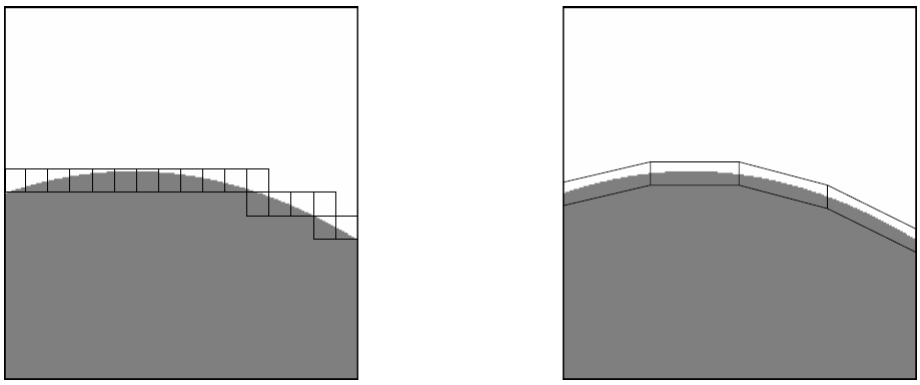


FIG. 3.17 – Représentation des fonctions de base isotropiques des ondelettes 1G et anisotropiques des *directionlets* [VBLVD06]

De plus, les ondelettes de première génération n'utilisent que les directions verticales et horizontales et un filtrage n'ayant de moments nuls que dans ces directions. Les *directionlets* sont donc multi-directionnelles et les filtres utilisés ont des moments nuls directionnels.

Décomposition en ondelettes anisotropiques

Dans cette décomposition, le nombre de transformations appliquées selon la verticale et l'horizontale n'est pas nécessairement le même. n_1 transformations sont effectuées selon l'horizontale et n_2 selon la verticale à chaque niveau de la décomposition. Les itérations sont répétées sur la bande passe-bas comme dans le cas des ondelettes de première génération.

Cette transformée anisotropique est appelée AWT(n_1,n_2), et le coefficient d'anisotropie est alors donné par : $\rho = n_1/n_2$. Il représente l'élongation des fonctions de base. On peut noter que la transformée en ondelettes classiques est alors AWT(1,1). Un exemple de décomposition AWT(2,1) est donné sur la figure 3.18.

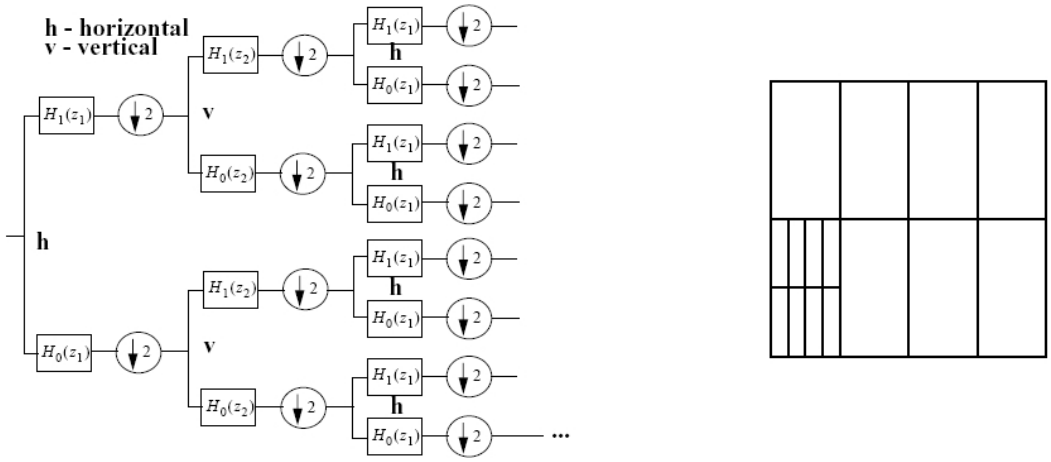


FIG. 3.18 – Exemple de décomposition AWT(2,1) avec une étape de filtrage [VBLVD06]

Transformée en ondelettes oblique basée treillis

Les *directionlets* se veulent séparables, entières et multi-directionnelles. Il est donc nécessaire pour cela de définir un treillis entier.

Un ligne continue est caractérisée par sa pente r et par son intersection avec les abscisses b, elle est définie par : $y = rx + b$.

Une approximation discrète de ces lignes sont les lignes numériques $L(r, n)$, telles que l'ensemble $\{L(r, n) : n \in \mathbb{Z}\}$ partitionne l'espace discret $\mathbb{Z}^2$. Ces lignes discrétisées définissent alors l'ensemble de points (i, j) tels que :

$$\begin{cases} j = \lceil ri \rceil + n, & \forall i \in \mathbb{Z}, \quad pour\ |r| \leq 1, \\ i = \lceil j/r \rceil + n, & \forall j \in \mathbb{Z}, \quad pour\ |r| > 1 \end{cases} \tag{3.15}$$

Les interactions directionnelles ne permettent pas d'appliquer directement le filtrage le long de ces lignes numériques [VBLVD06]. En effet, si deux directions coexistent, le filtrage selon l'une des deux directions génère des défauts sur la seconde. Il convient alors d'appliquer le filtrage suivant un treillis entier.

Un treillis entier Λ est obtenu par combinaisons linéaires de deux vecteurs linéairement indépendants dont les composantes et les coefficients sont entiers. Tout treillis entier Λ est un sous-treillis du treillis entier $\mathbb{Z}^2$ et il est défini par une matrice génératrice non-unique :

$$M_\Lambda = \begin{bmatrix} a_1 & b_1 \\ a_2 & b_2 \end{bmatrix} = \begin{bmatrix} \mathbf{d}_1 \\ \mathbf{d}_2 \end{bmatrix}, \quad \text{où} a_1, a_2, b_1, b_2 \in \mathbb{Z} \tag{3.16}$$

La direction le long du premier vecteur $\mathbf{d}_1$ (avec une pente $r_1 = b_1/a_1$) est appelé direction de la transformée, et celle le long du second vecteur $\mathbf{d}_2$ la direction d'alignement.

Ce treillis Λ, correspondant à la matrice M_Λ (3.16), partitionne alors chaque ligne numérique $L(r_1 = b_1/a_1, n)$ en co-lignes, qui sont les intersections entre ces lignes numériques et les sous-ensembles du treillis (au nombre de $|det(M_\Lambda)|$), définis par le vecteur de décalage $\mathbf{s}_k, k = 0, 1, \ldots, |det(M_\Lambda)| - 1$. Les co-lignes des lignes discrètes $L(r_1 = b_1/a_1, n)$ sont notées $CL_{\mathbf{s}_k}(r_1, n)$, et celles des lignes $L(r_2 = b_2/a_2, n)$ par $CL_{\mathbf{s}_k}(r_2, n)$. Un exemple est donné sur la figure 3.19.

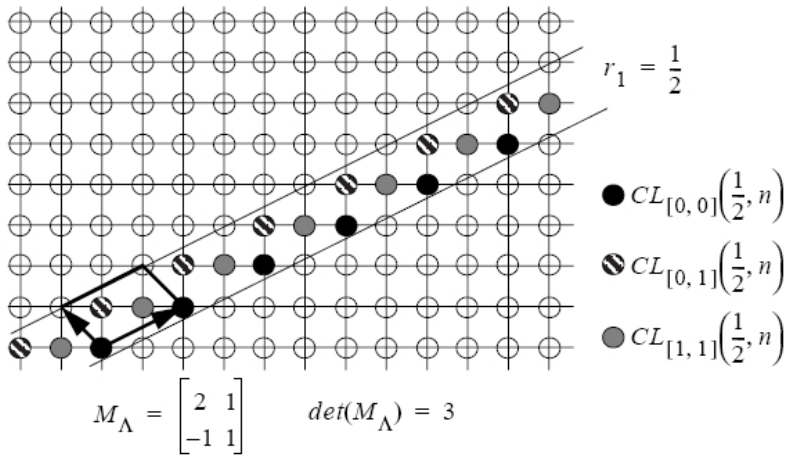


FIG. 3.19 – Exemples de co-lignes obtenues avec un treillis défini par M_Λ et $r_1 = 1/2$ [VBLVD06]

Une décomposition en ondelettes 1D est alors appliquée indépendamment suivant chacune de ces co-lignes. Elle consiste en une étape de filtrage 1D et une étape de sous-échantillonnage, après laquelle les points appartiennent au treillis Λ' (sous-treillis de Λ). La matrice génératrice de ce treillis Λ' est donnée par :

$$M_{\Lambda'} = D_{\mathbf{s}}.M_\Lambda = \begin{bmatrix} 2\mathbf{d}_1 \\ \mathbf{d}_2 \end{bmatrix} \tag{3.17}$$

où $D_{\mathbf{s}}$ est l'opérateur de sous-échantillonnage horizontal :

$$D_{\mathbf{s}} = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} \tag{3.18}$$

comme illustré sur la figure 3.20.

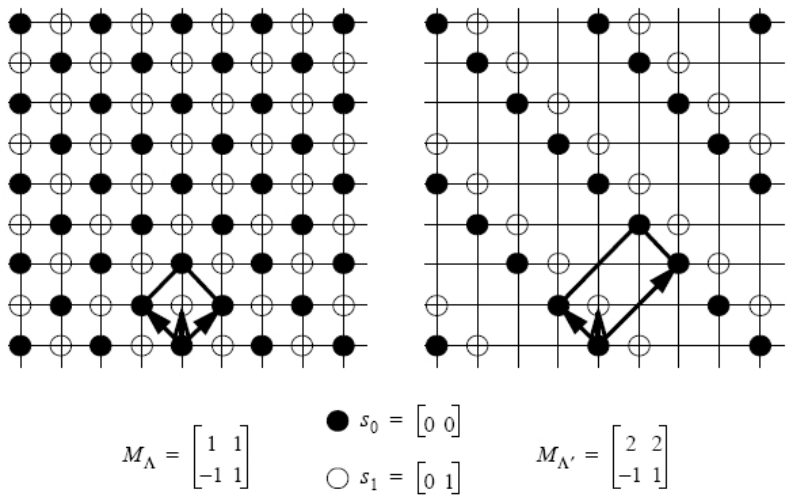


FIG. 3.20 – Exemple de sous-échantillonnage horizontal [VBLVD06]

Le filtrage appliqué ici le long des co-lignes est réalisé par une version oblique de la décomposition en ondelettes anisotropiques AWT. Cette transformée oblique est appliquée à chacune des co-lignes suivant les directions de la transformée et d'alignement du treillis Λ.

La transformée en ondelettes oblique et anisotropique est alors notée S-AWT(M_Λ,n_1,n_2). Elle effectue, par itération, n_1 transformations dans la direction de la transformée, et n_2 transformations dans la direction d'alignement. Les fonctions de base de la S-AWT sont appelées *directionlets*, elles sont anisotropiques et directionnelles.

En pratique, cette transformée qui consiste à filtrer et sous-échantillonner chaque sous-treillis suivant la direction de la transformée est réalisée en trois étapes (cf fig. 3.21) :

- Séparation en sous-treillis,
- Rotation par la matrice génératrice M_Λ,
- Filtrage et sous-échantillonnage dans la direction horizontale.

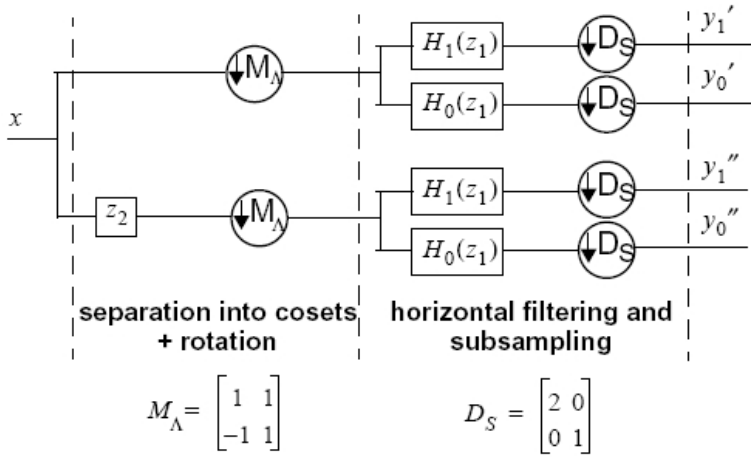


FIG. 3.21 – La transformée *directionlet* en version pratique [VBLVD06]

La sélection des directions est réalisée ici comme dans le cas des bandelettes de première génération, c'est-à-dire dans chaque carré dyadique obtenu à l'aide d'une segmentation spatiale par arbre quaternaire (cf fig. 3.22).

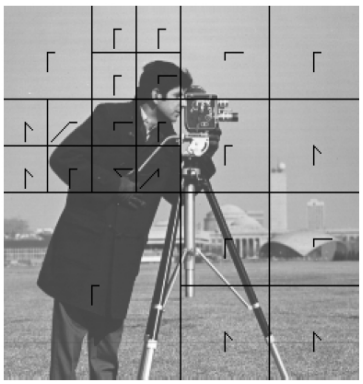


FIG. 3.22 – Les directions de la transformée adaptées aux directions dominantes [VBLVD06]

Quelques résultats d'approximation d'images par ces *directionlets* sont illustrés sur la figure 3.23.

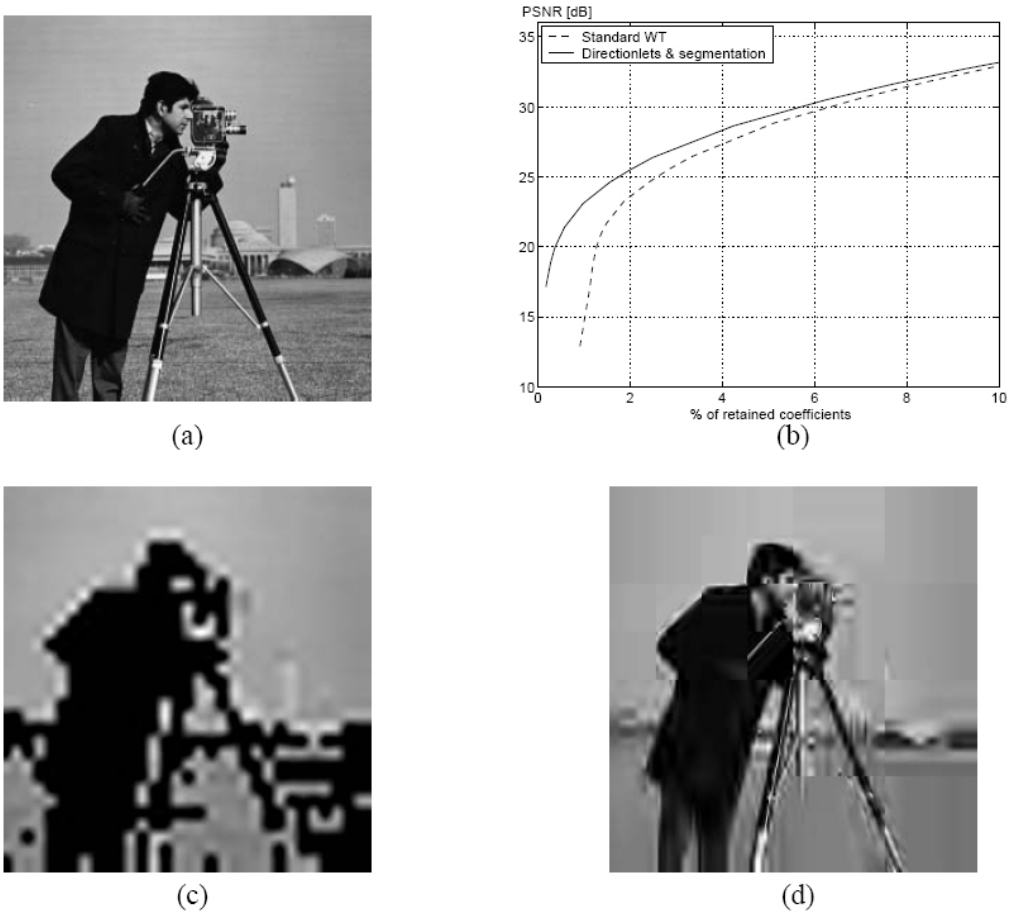


FIG. 3.23 – Comparaison entre les ondelettes et les *directionlets*. (a) Image originale (b) PSNR (c) Image ondelette reconstruite avec 0.98% des coefficients (13.93 dB) (d) Image *directionlet* reconstruite avec 0.98% des coefficients (23.09 dB) [VBLVD06]

3.3 Les DCT exploitant l'orientation

La section précédente 3.2 nous a montré que depuis plusieurs années de nombreux travaux ont été menés afin d'exploiter les orientations dans les transformées en ondelettes.

Plusieurs auteurs ont également effectué des recherches sur la transformée DCT, qui est plus utilisée dans les standards vidéo que les transformées en ondelettes de première ou de seconde génération.

3.3.1 La DCT adaptée aux contours SA-DCT

La norme MPEG-4 partie 2 (cf Annexe A.2.5) utilise une DCT particulière adaptée aux contours, la "Shape-Adaptive DCT" (SA-DCT) [SM95] [KS98] [FKE07].

Cette SA-DCT s'applique à des images segmentées (c.à.d. des images contenant des objets séparés du fond) telles que les images d'objets vidéos de la norme MPEG-4 partie 2 (cf Annexe A.2.5 fig. A.6). Elle nécessite de transmettre aussi une carte de segmentation des blocs. Si elle est appliquée à une image non segmentée, la SA-DCT devient équivalente à la DCT classique.

Cette SA-DCT se déroule comme illustré sur la figure 3.24.

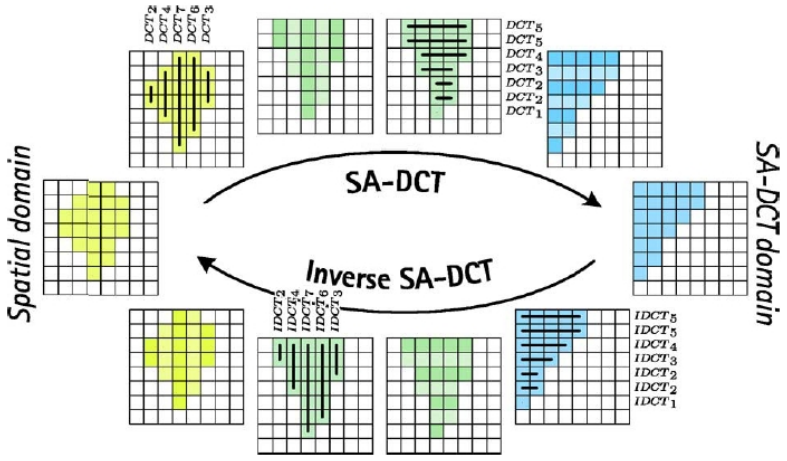


FIG. 3.24 – Illustration de la SA-DCT et de son inverse [FKE07]

Une première transformation est appliquée sur uniquement les colonnes de l'objet, avec des DCT adaptées à la longueur N de ces colonnes, qui auront été calculées au préalable. Les coefficients DCT ainsi obtenus sont alors alignés en haut du bloc.

Une seconde étape de transformation, qui suit le même schéma que la première, est alors appliquée sur les lignes de longueur N obtenues précédemment et les coefficients obtenus sont alignés à gauche du bloc.

Les DCT à longueur variable utilisées ici peuvent s'écrire sous la forme :

$$\textbf{DCT-N}(p,k) = \alpha_p \cdot \cos\left[p\left(k+\frac{1}{2}\right)\cdot\frac{\pi}{N}\right] \qquad k,p = 0 \rightarrow N-1 \tag{3.19}$$

où $\alpha_0 = \sqrt{1/2}$ (si $p = 0$), sinon $\alpha_p = 1$, et p représente la p-ième fonction de base de la DCT.

Les coefficients DCT $\mathbf{c}_j$ d'un vecteur (colonne ou ligne) de longueur N $\mathbf{x}_j$ peuvent alors être obtenus par :

$$\mathbf{c}_j = S_N \cdot \mathbf{DCT\text{-}N} \cdot \mathbf{x}_j \qquad \text{avec } S_N = \sqrt{\frac{2}{N}} \tag{3.20}$$

Ce paramètre S_N initialement fixé à $2/N$ définissait une SA-DCT non normalisée, mais la valeur utilisée ici ($\sqrt{2/N}$) permet de la rendre pseudo-orthonormale.

Le vecteur (colonne ou ligne) reconstruit $\hat{\mathbf{x}}_j$ à partir du vecteur de coefficients reçus $\hat{\mathbf{c}}_j$ au décodage sera alors obtenu par :

$$\hat{\mathbf{x}}_j = \mathbf{DCT\text{-}N}^T \cdot \hat{\mathbf{c}}_j \tag{3.21}$$

Cependant, afin de réaliser la reconstruction réelle du bloc, il est nécessaire d'avoir transmis en même temps que le bloc codé des informations sur le contour (position du contour dans ce bloc). Ces informations permettent d'effectuer les opérations inverses des alignements en haut et à gauche du bloc (étapes 2 et 4 de la SA-DCT inverse fig. 3.24), elles permettent donc de repositionner le contour dans le bloc.

Les résultats de cette SA-DCT [SM95], insérée dans un codeur MPEG-1 [ISO93] et appliquée à une image Foreman dont le fond a été codé au préalable par le même codeur MPEG-1 dans les deux cas, sont présentés sur la figure 3.25.

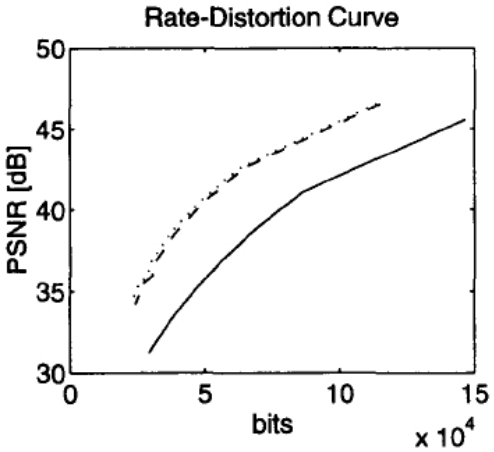


FIG. 3.25 – Résultats de la SA-DCT inséré dans un codeur MPEG-1 (traits pleins MPEG-1, pointillés SA-DCT sans les informations de contour et tirets avec ces informations) [SM95]

Cette courbe montre l'efficacité de la SA-DCT à tous les débits par rapport à une DCT flottante 8×8 classique de MPEG-1. De plus, on peut voir que l'impact des informations de contour est négligeable devant le gain apporté par la SA-DCT.

Cependant, cette construction de la SA-DCT entraîne un défaut de moyenne ("mean weighted defect" [KS98]). En effet, l'utilisation de différentes DCT (de différentes longueurs) donne des poids différents aux coefficients, en modifiant le coefficient S_N de ces transformations DCT (3.20). La véritable moyenne m d'un bloc est de ce fait portée par plusieurs coefficients. Ainsi, la quantification peut avoir un effet désastreux sur cette valeur moyenne m, dégradant considérablement le bloc reconstruit.

Kauff et Schuur [KS98] proposent une solution à ce problème, consistant en une séparation du coefficient DC et une correction ΔDC.
La séparation du coefficient DC est effectuée au codage et elle consiste à soustraire aux pixels du bloc traité sa moyenne m avant tout traitement.
La correction ΔDC est effectuée au décodage et consiste à compenser les erreurs de quantification sur tous les coefficients décodés afin d'obtenir une moyenne nulle, la véritable moyenne m étant ajoutée à posteriori.

La norme MPEG-4 partie 2 utilise la SA-DCT non orthonormale (avec $S_N = 2/N$) décrite précédemment, nommée NO-SA-DCT. Une version pseudo-orthonormale (avec $S_N = \sqrt{2/N}$) utilisant cette méthode de séparation du coefficient DC et correction ΔDC, nommée ΔDC-SA-DCT, a donc été insérée dans ce codeur MPEG-4 par Kauff et al. [KS98], afin d'en évaluer les performances. Les résultats de ces expérimentations sur des images QCIF (Akiyo et Weather) et CIF (les autres) sont présentés sur la figure 3.26.

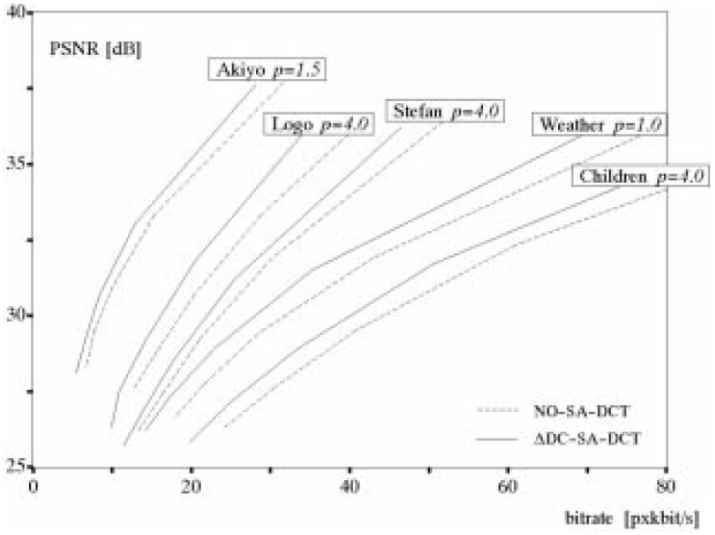


FIG. 3.26 – Résultats de la ΔDC-SA-DCT par rapport à la NO-SA-DCT du codeur MPEG-4 pour des séquences $IPPPP\ldots$(un paramètre p met chacune des séquences à une échelle différente) [KS98]

Cette méthode de séparation du coefficient DC et correction ΔDC permet d'améliorer en moyenne de 1 dB la SA-DCT classique [SM95], et en ne prenant en compte que les blocs de contour, ce gain est supérieur à 2 dB.

3.3.2 La DCT directionnelle

Fu et Zeng ont effectués des travaux sur une DCT directionnelle [ZF06] [FZ07a] [FZ07b], qui permet d'appliquer la DCT flottante 8×8 (type JPEG [ISO94a] et MPEG-2 [ISO94b]) suivant différentes directions.

Les directions sélectionnées pour cette DCT directionnelle sont 8 des 9 directions de la prédiction Intra de H.264/AVC (cf Annexe A.3.2 fig. A.9). Le mode DC (mode 2) n'a pas d'intérêt

ici, il n'est donc pas utilisé. Les modes 0 et 1, qui correspondent à la verticale et à l'horizontale respectivement impliquent que la DCT directionnelle devient équivalente à la DCT classique, qui est appliquée à la verticale et à l'horizontale.

Quelle que soit la direction sélectionnée, cette transformation suit le même déroulement. Elle se décompose en trois étapes. Un exemple de cette DCT directionnelle est donné pour la diagonale vers le bas à droite (mode 3) sur la figure 3.27.

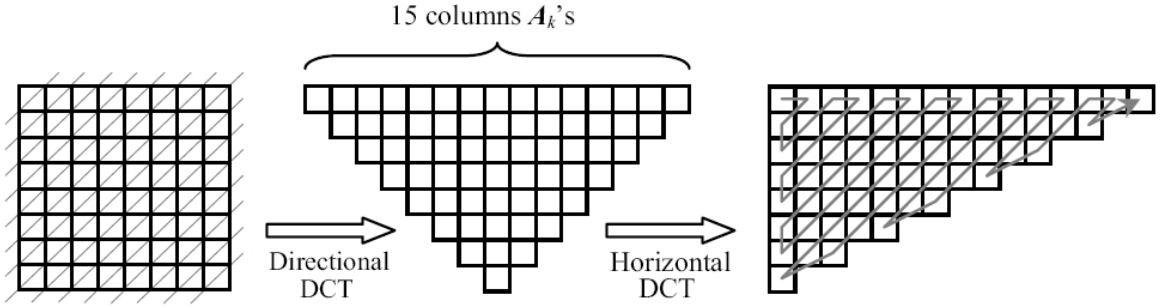


FIG. 3.27 – Décomposition de la DCT directionnelle appliquée selon la diagonale vers le bas à droite (mode 3) [ZF06]

Lors de la première étape, les données sont transformées à l'aide d'une DCT 1D flottante et suivant la direction sélectionnée. Ceci nécessite alors de définir 8 DCT 1D permettant de transformer de 1 à 8 pixels. Par exemple, dans un cas à 45^o, on transforme d'abord 1 pixel puis 2 puis 3... puis 8 puis 7... et enfin 1 pixel. Les coefficients DCT ainsi obtenus sont alors organisés en colonnes que l'on nommera A_k.

Dans la seconde étape, ces coefficients sont transformés à l'aide d'une seconde DCT 1D appliquée selon l'horizontale. En effet, après la première transformation, chacune des colonnes A_k possède un coefficient DC comme premier élément et des coefficients AC par la suite. Ces coefficients sont organisés en lignes et il convient donc de les transformer à l'horizontale.

La dernière étape permet d'aligner les coefficients obtenus à gauche avant qu'ils ne soient quantifiés et scannés avec un zig-zag modifié et adapté à la nouvelle forme du bloc.

Cependant, de par sa construction basée sur des DCT à longueur variable, cette DCT directionnelle souffre comme la SA-DCT d'un défaut de moyenne ("mean weighted defect" [KS98]). Cette transformée utilise donc pour remédier à ce problème la méthode de séparation du coefficient DC et correction ΔDC de la SA-DCT présentée précédemment en section 3.3.1.

Les performances de cette DCT directionnelle par rapport à une DCT flottante 8×8 classique ont été testées par Fu et Zeng [ZF06] [FZ07b]. Ces expérimentations ont été menées en codage d'images naturelles fixes dans deux codeurs différents. Le premier est basé sur JPEG [ISO94a], en conservant sa quantification et son codage entropique. Le second est basé sur H.263 [ITU95], en utilisant sa quantification et son codage entropique, et il est utilisé pour coder des images naturelles extraites de séquences vidéo. Ces résultats sont présentés sur la figure 3.28 et illustrés sur la figure 3.29. Ils incluent les modes directionnels codés par un codage à longueur variable.

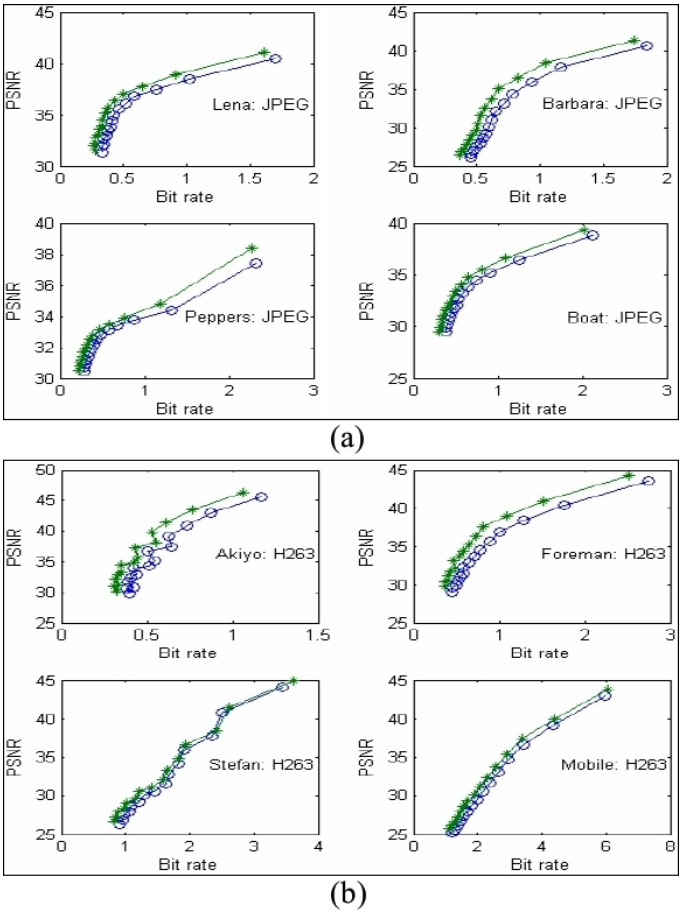


FIG. 3.28 – Comparaison entre la DCT directionnelle et la DCT flottante 8×8 (a) de JPEG et (b) de H.263 [ZF06]

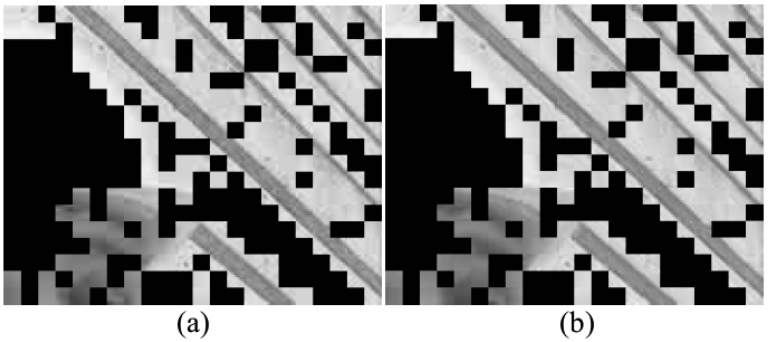


FIG. 3.29 – Comparaison subjective entre (a) la DCT flottante 8×8 de H.263 et (b) de la DCT directionnelle avec uniquement les blocs orientés (mode $\neq$ 0,1) [ZF06]

Ces résultats montrent bien que cette DCT directionnelle permet d'améliorer le traitement DCT que ce soit pour JPEG ou pour H.263 d'en moyenne 1 dB à haut débit à 2 dB dans les bas débits.

3.3.3 La BinDCT orientée

Xu et al. [XXW07] ont récemment étudié une version orientée de la BinDCT présentée en section 2.3.2.

Cette méthode s'insère dans un codeur de type JPEG [ISO94a] à la place de la DCT flottante 8×8 (les étapes de quantification et codage entropique restent identiques à JPEG). Elle utilise un schéma de la BinDCT proche de celui de la figure 2.15, mais dont les entrées sont sélectionnées sur une grille directionnelle, comme illustré sur la figure 3.30.

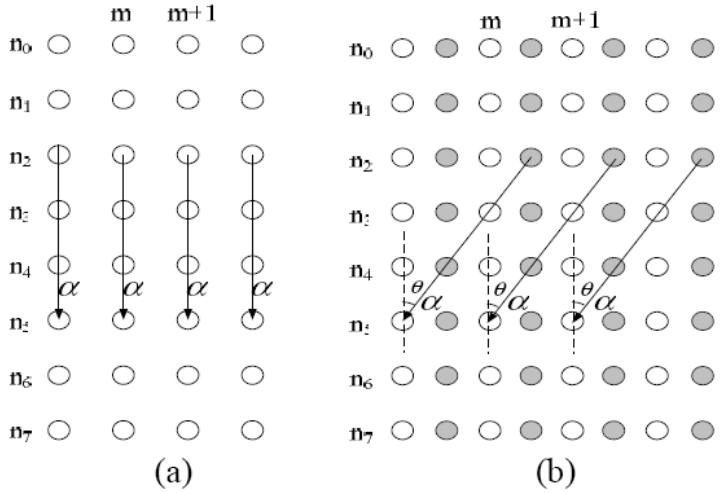


FIG. 3.30 – Exemples d'opérations effectués (a) sur une grille non-directionnelle (b) sur une grille directionnelle où les cercles blancs représentent les pixels entiers et les cercles gris les demi-pixels [XXW07]

Cette BinDCT orientée est appliquée suivant plusieurs directions prédéfinies, données par :

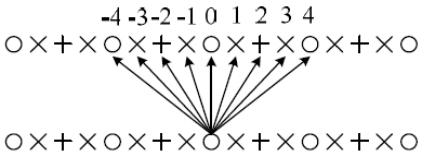


FIG. 3.31 – Directions prédéfinies pour la BinDCT orientée (∘ entier, + demi et × quart de pixels) [XXW07]

Ces directions sont sélectionnées pour chaque bloc de l'image à l'aide d'un critère débit-distorsion. Elles sont ensuite codées sans perte par un codage VLC prédit, avant d'être transmises au décodeur.

Cette BinDCT orientée a été testée par rapport à un codeur classique JPEG, les résultats pour les images Barbara (512×512), Foreman (512×512), Lena (512×512) et Peppers (512×512) [XXW07] sont présentés sur la figure 3.32 et illustrés sur la figure 3.33.

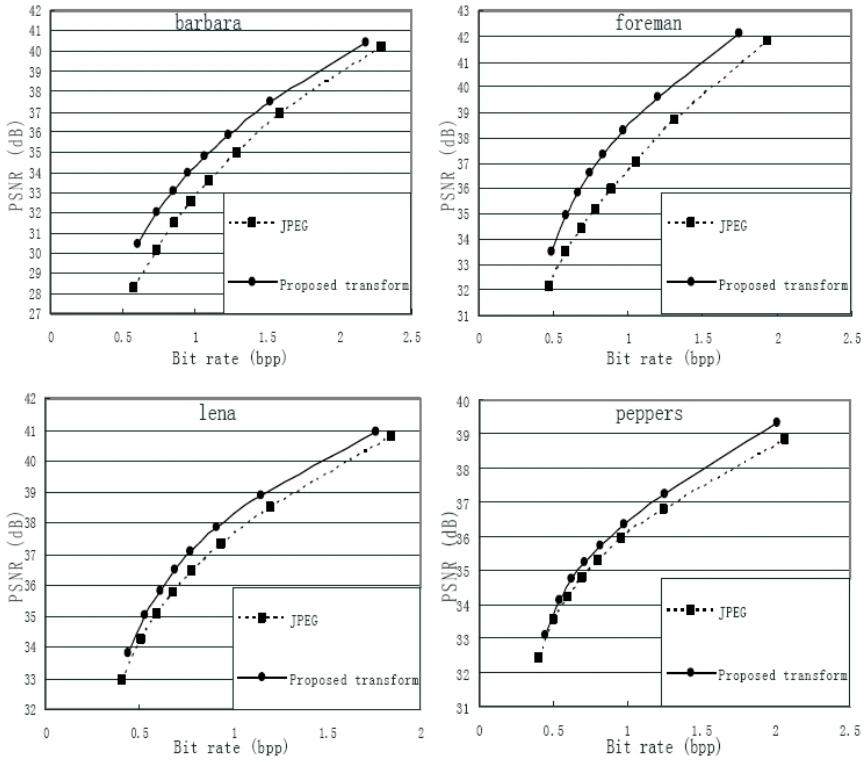


FIG. 3.32 – Comparaison entre la BinDCT orientée et la DCT flottante 8×8 insérées dans une codeur JPEG classique [XXW07]

FIG. 3.33 – Résultat visuel pour une partie de l'image Foreman codée avec la BinDCT orientée (à droite) comparée à la DCT flottante de JPEG (à gauche) [XXW07]

Ces résultats montrent que la sélection d'entrées orientées pour la BinDCT permet de dépasser la DCT flottante 8×8 classique avec un gain qui peut atteindre 2 dB (suivant l'image). Cependant, cette méthode nécessite une étape d'interpolation ("2T-tap interpolation filter") qui implique des opérations non entières afin de calculer les valeurs des demis et quarts de pixels.

3.4 Conclusion

Ce chapitre présente des transformées qui exploitent avantageusement les structures géométriques des images, c'est-à-dire leurs orientations. Elles permettent de représenter plus efficacement les contours améliorant ainsi leurs codages.

Les ondelettes de seconde génération sont les plus anciennes et les plus nombreuses de ces transformées exploitant les orientations.
Les premières de ces ondelettes permettent de représenter les singularités le long de droites : les *ridgelets*. Mais rapidement, en se basant sur ces *ridgelets*, sont apparus des versions représentant les singularités le long de courbes : dans le cas continu avec les *curvelets* et dans le cas discret avec les *contourlets*. Les décompositions sont alors obtenues à l'aide de bancs de filtres directionnels. Plus récemment, d'autres transformées en ondelettes de seconde génération ont été développées, mais elles sont généralement beaucoup plus complexes. Il s'agit entre autres des bandelettes de première et seconde génération et des *directionlets*.

Bien que ces ondelettes de seconde génération permettent de représenter efficacement les contours, il subsiste encore certains inconvénients à toutes ces méthodes.
Les *ridgelets*, les *curvelets* et les bandelettes de première génération sont définies dans le cas continu, elles sont donc difficilement applicables aux images discrètes. Les *contourlets* (comme les *curvelets*) sont redondantes ce qui a pour effet de diminuer leur efficacité. Les bandelettes nécessitent des étapes de détections de contours très complexes et de ré-échantillonnage par interpolation non réversible. Les *directionlets* quant à elle nécessite d'effectuer des rotations par interpolation.

Cependant, il existe aussi d'autres techniques orientées qui peuvent être basées sur la transformation DCT plus courante (dans les schémas normalisés).
La plus ancienne de ces transformées est la SA-DCT qui applique des DCT de taille variable à des images segmentées. Ces mêmes DCT de taille variable ont été réutilisées, par la suite, par la DCT directionnelle qui les appliquent selon différentes directions. Une BinDCT orientée a aussi été développée en utilisant une grille d'entrée directionnelle.

Ces méthodes ont elles aussi, malgré leurs performances accrues par rapport aux transformées classiques, quelques inconvénients.
La SA-DCT nécessite de transmettre en plus des coefficients DCT une carte de segmentation. De plus, elle utilise, comme la DCT directionnelle, des DCT flottantes de taille variable dont les problèmes liés aux arrondis sont bien connus. Cette DCT directionnelle déforme les blocs et implique de modifier la quantification et le parcours des coefficients. La BinDCT utilise, quant à elle, une grille orientée définit à l'aide d'une interpolation non réversible.

Toutes les transformées présentées utilisent avantageusement les orientations des images, mais elle souffrent toutes d'un certain nombre d'inconvénients. Afin de palier ces problèmes, nous avons cherché à définir de nouvelles transformations, et notamment pour le codeur H.264/AVC : en se basant sur des méthodes existantes comme les ondelettes ou les transformées à recouvrement, ou en les concevant complètement.

Troisième partie

Nos contributions

Chapitre 4

Compléments à la transformée DCT entière

4.1 La DCT de H.264/AVC sous forme lifting

4.1.1 Introduction

Comme on a pu le voir dans les sections précédentes 3.2 et 3.3, les techniques de transformations orientées sont utilisées essentiellement par les ondelettes de seconde génération. Cependant, ces ondelettes sont décomposables en bancs de filtres ou en étages lifting. Ces schémas lifting nous permettent de définir des transformations, vues comme des ondelettes (possédant les mêmes propriétés) et auxquelles on pourra associer des étapes orientées issues de la section 3.2.

La section 2.3 nous a montré qu'il existe plusieurs techniques qui permettent de réaliser la transformation DCT rapidement, à l'aide de décompositions en lifting : la BinDCT et la IntDCT. Cependant, on a vu aussi que ces techniques ne sont que des approximations de la DCT flottante 8×8 de JPEG et/ou de H.263, réalisées par calculs entiers.

C'est pourquoi il nous semble important de définir une transformation en étages lifting réalisant exactement la transformation DCT de H.264 MPEG-4/AVC.

4.1.2 Mise en place du lifting de la DCT de H.264/AVC

H.264 MPEG-4/AVC utilise une transformation DCT matricielle entière sur des blocs 4×4 qui s'écrit, sans prendre en compte les facteurs de normalisation, sous la forme :

$$X = H^T \,.\, x \,.\, H \qquad \text{avec} \quad H = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix} \tag{4.1}$$

où X est la matrice 4×4 des coefficients DCT issue de la transformation du bloc 4×4 par H.

Dans le cas unidimensionnel, on peut écrire :

$$\begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

soit,

$$\begin{cases} X_1 = x_1 + x_2 + x_3 + x_4 \\ X_2 = 2x_1 + x_2 - x_3 - 2x_4 \\ X_3 = x_1 - x_2 - x_3 + x_4 \\ X_4 = x_1 - 2x_2 + 2x_3 - x_4 \end{cases}$$

Avec plusieurs étages de filtres de Haar, il est donc possible d'obtenir les coefficients recherchés, selon le schéma lifting de la figure 4.1 :

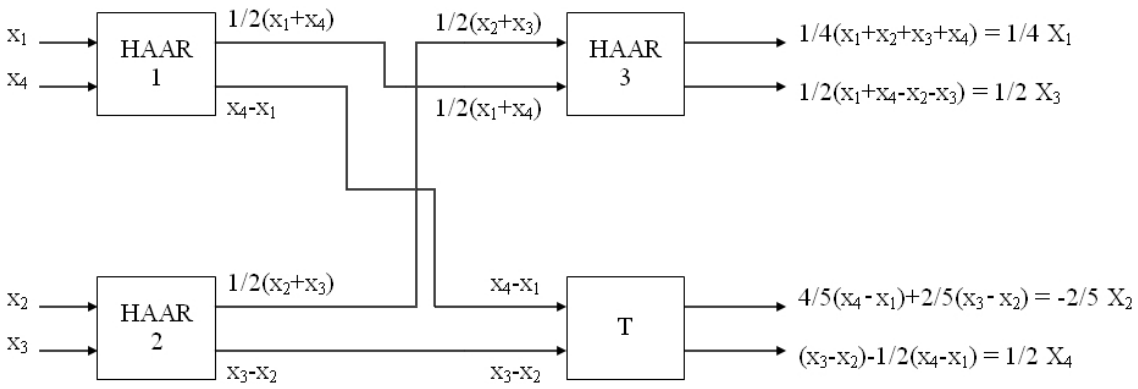


FIG. 4.1 – Représentation de la DCT de H.264/AVC en étages lifting

où Haar correspond à un filtre de Haar classique comme utilisé en ondelettes, et T est un filtre correspondant à la matrice :

$$T = \begin{bmatrix} 2 & 1 \\ -1 & 2 \end{bmatrix}$$

Chacun des filtres de Haar présents dans cette représentation peut être réalisé sous forme lifting selon :

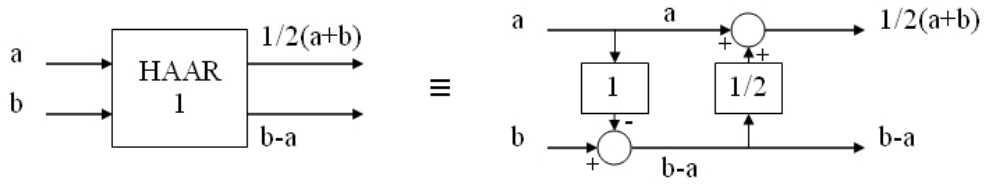


FIG. 4.2 – Le filtre de Haar sous forme lifting

et la transformation T peut elle aussi être décomposée en étages lifting selon :

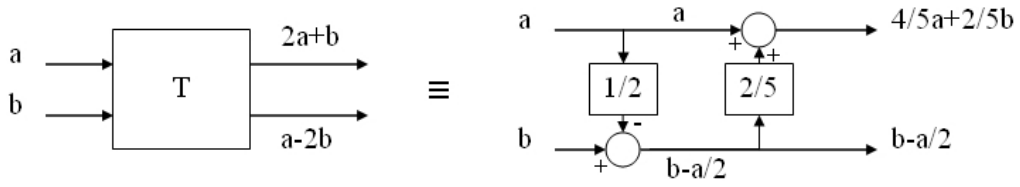


FIG. 4.3 – La transformation T sous forme lifting

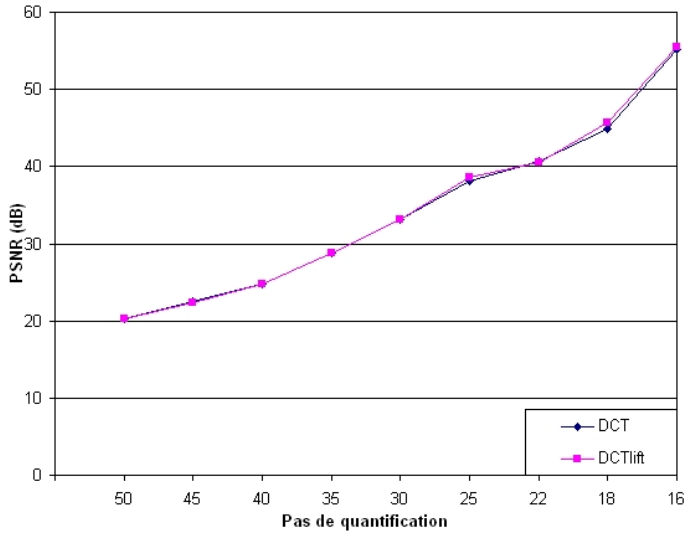
A la sortie de ce schéma, les coefficients $[1/4X_1, -2/5X_2, 1/2X_3, 1/2X_4]$ sont normalisés avec les facteurs inverses : $[4, -5/2, 2, 2]$ afin d'obtenir les coefficients $[X_1, X_2, X_3, X_4]$ souhaités.

4.1.3 Les résultats expérimentaux

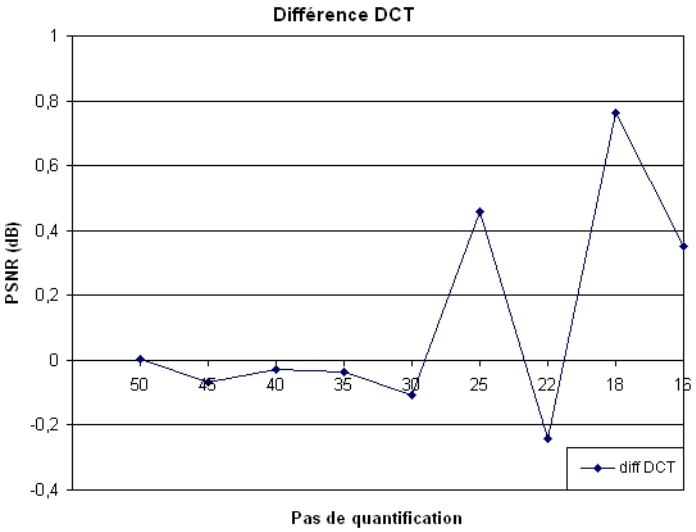
Cette DCT sous forme lifting (DCTlift) (cf fig. 4.1) a été insérée dans le codeur H.264/AVC à la place de sa DCT entière 4×4.

Nous avons réalisé plusieurs expérimentations afin de valider l'équivalence entre cette DCT sous forme lifting et la DCT classique de H.264/AVC.

Pour cela, plusieurs séquences ont été testées avec un ensemble de pas de quantification H.264/AVC QP (variant logarithmiquement cf Annexe A.3.5). Les résultats de quelques uns de ces tests sont présentés sur les figures 4.4 et 4.5.

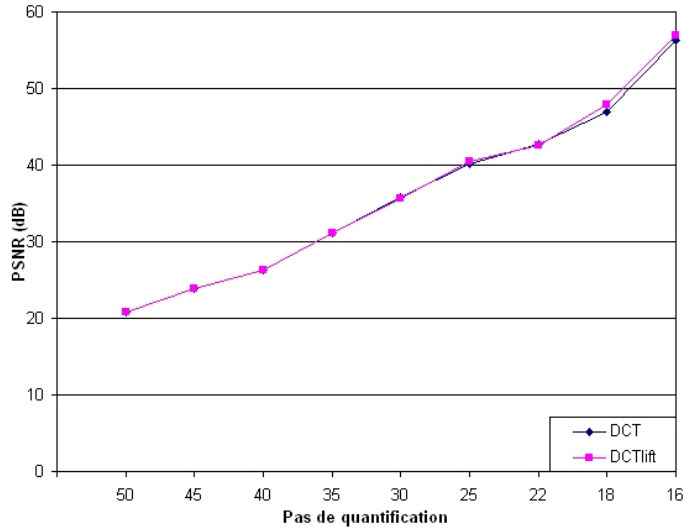


(a) Courbe Pas de quantification-Distorsion

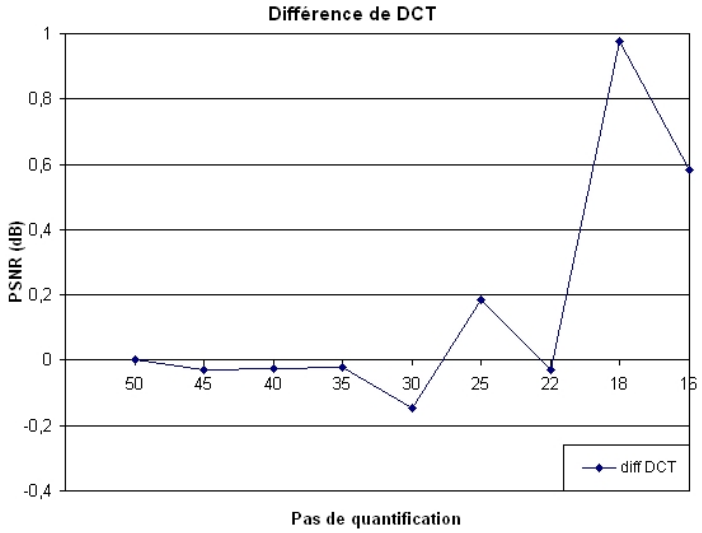


(b) Différence entre les DCT

FIG. 4.4 – Comparaison entre la DCT sous forme lifting (DCTlift) et la DCT classique H.264/AVC pour la séquence Akiyo



(a) Courbe Pas de quantification-Distorsion



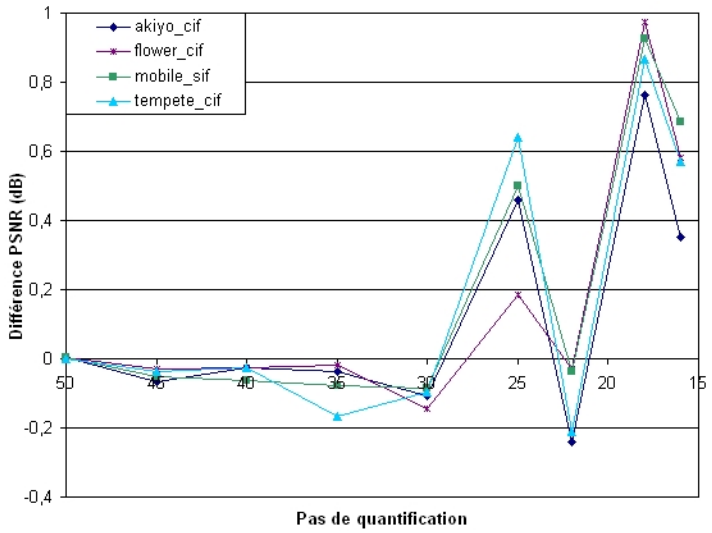
(b) Différence entre les DCT

FIG. 4.5 – Comparaison entre la DCT sous forme lifting (DCTlift) et la DCT classique H.264/AVC pour la séquence Flower

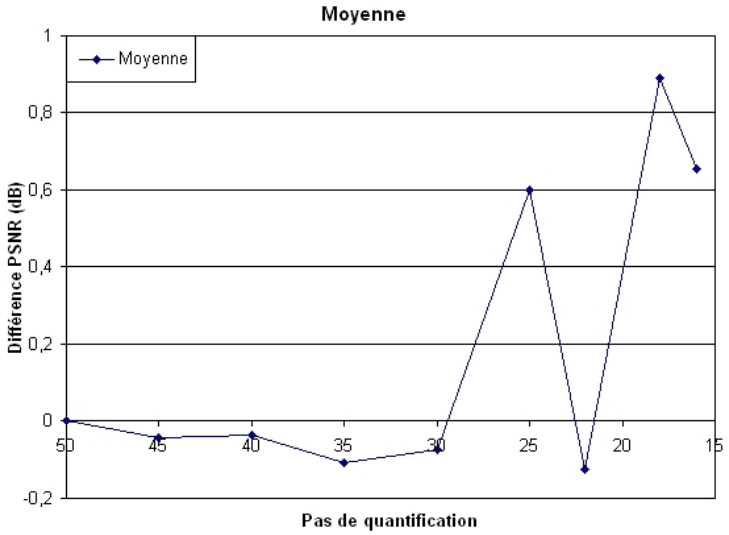
Sur ces figures, les premiers graphiques (4.4(a) et 4.5(a)) représentent l'évolution du PSNR entre la séquence codée/décodée et la séquence originale, en fonction des pas de quantification. La DCT sous forme lifting testée doit reproduire le comportement de la DCT H.264/AVC, c'est-à-dire qu'elle doit fournir les mêmes coefficients et donc le même débit.

Ces premières courbes (4.4(a) et 4.5(a)) permettent de calculer la différence entre la DCT sous forme lifting et la DCT H.264/AVC, comme illustré sur les secondes courbes (4.4(b) et 4.5(b)). Cette différence permet de mieux interpréter le comportement de notre DCT sous forme lifting vis-à-vis de la DCT H.264/AVC.

Afin de mieux comparer ces résultats, nous avons réalisé une moyenne des différences entre les deux DCT sur les séquences Akiyo, Flower, Mobile et Tempete
Cette moyenne est représentée sur la figure 4.6(b).



(a) Ensemble de différences entre DCT



(b) Moyenne des différences entre les DCT

FIG. 4.6 – Moyenne des comparaisons entre la DCT sous forme lifting et la DCT classique H.264/AVC pour un ensemble de séquences

On peut remarquer que la DCT sous forme lifting suit quasiment le même comportement, quelle que soit la séquence testée.

A bas débit, c'est-à-dire pour les QP supérieurs à 25 (soit des débits inférieurs à 1000 kbits/s), on voit sur cette figure 4.6 que la DCT sous forme lifting approche bien la DCT H.264/AVC, la différence étant inférieure 0.1 dB (en valeur absolue).

Par contre, pour les hauts débits (les QP inférieurs à 25, soit des débits supérieurs à 1000 kbits/s), la DCT sous forme lifting n'approche plus aussi bien la DCT H.264/AVC.

En effet, l'utilisation du facteur 2/5 dans la transformation T (cf fig. 4.1 et fig. 4.3) et du facteur $-5/2$ de normalisation à la sortie de la DCT sous forme lifting entraîne une dérive entre les deux DCT testées. Ces facteurs non entiers font que la DCT sous forme lifting génère des coefficients non entiers qu'il est nécessaire d'arrondir (par exemple à l'entier le plus proche) pour garder la cohérence avec la DCT H.264/AVC. Ces coefficients après arrondis ne sont pas forcément les mêmes que ceux fournis par la DCT H.264/AVC.
De plus, pour ces débits, les QP sont inférieurs à 25, c'est-à-dire que les pas de quantification correspondants sont inférieurs à 10 (cf Annexes A.3.5 Tab.A.5). La quantification affecte donc directement la première décimale des coefficients quantifiés, pouvant ainsi modifier l'arrondi effectué après quantification.
Comme on a pu le voir, les coefficients de la DCT sous forme lifting peuvent être légèrement différents de ceux de la DCT H.264/AVC, et les pas de quantification étant inférieurs à 10 dans ce cas de hauts débits impliquent que les coefficients quantifiés peuvent eux aussi être légèrement différents.

ex : - pour la DCT H.264/AVC, un coefficient 5 est obtenu, et après quantification par 10 on obtient 0.5 soit 0 (arrondi au plus proche entier).
- pour la DCT sour forme lifting, ce coefficient vaut 5.6 soit 6 (arrondi au plus proche entier), et après quantification par 10 on obtient 0.6 soit 1 (arrondi au plus proche entier).

Cet exemple montre bien que les coefficients transformés après la DCT sous forme lifting peuvent être légèrement différents de ceux obtenus après la DCT H.264/AVC. Dans ce cas, si on est à hauts débits, la quantification est faible, elle peut entraîner une différence entre les coefficients quantifiés, ce qui modifie le train binaire et donc le PSNR et le débit de la séquence (soit une amélioration du PSNR associée à une augmentation du débit, soit une dégradation du PSNR associée à une réduction du débit). Sinon, à bas débits, la quantification est beaucoup plus forte limitant ainsi ces différences entre coefficients quantifiés : dans notre exemple, avec un QP de 40 soit un pas de quantification de 64 (cf Annexes A.3.5 Tab.A.5), ces coefficients DCT 5 et 6 deviennent tous deux des coefficients dont la valeur quantifiée est nulle.

Ceci montre bien que la DCT sous forme lifting permet d'approcher correctement la DCT H.264/AVC à bas débits (avec une quantification forte), mais que cette approximation dérive lorsqu'on monte dans les hauts débits (avec une quantification de plus en plus faible).

4.1.4 Conclusion

Cette méthode de DCT sous forme lifting est intéressante puisqu'elle permet de simuler la DCT H.264/AVC dans les bas débits. Cette mise sous forme lifting permet alors de voir la DCT comme une transformée ondelette à laquelle il serait possible d'ajouter des outils de seconde génération, tels que l'exploitation de l'orientation (cf section 3.2).

De plus, elle est simple, puisque le schéma (cf fig. 4.1) n'utilise que des transformées de Haar et une transformée T, soit une multiplication (par 5 dans la transformée T) et une division (par 5 lors de la normalisation en sortie de cette DCT sous forme lifting), les autres opérations étant réalisées à l'aide de décalages binaires.

Comme on a pu le voir dans la section précédente, cette DCT sous forme lifting utilise une division (par 5 en sortie) qui nécessite un arrondi et qui peut générer une différence entre les coefficients transformés issus de cette DCT sous forme lifting et ceux de la DCT H.264/AVC. A bas débits, la quantification qui suit est suffisamment forte pour supprimer les effets de cet arrondi. Par contre à hauts débits, la quantification est trop faible pour annuler cet effet, ce qui entraîne alors une dérive entre les coefficients quantifiés issus des deux méthodes de DCT. La DCT sous forme lifting n'approche alors plus efficacement la DCT H.264/AVC.

Cette méthode, bien qu'intéressante, puisque sa forme lifting permet de l'associer à des ondelettes et donc à des ondelettes de seconde génération orientées, ne vérifie cependant pas l'équivalence avec la DCT H.264/AVC à tous les débits. Il ne nous a donc pas semblé judicieux de continuer dans cette voie, partant déjà avec un handicap (la non-équivalence). Nous avons plutôt cherché d'autres outils permettant d'utiliser l'orientation des images pour en améliorer le codage.

4.2 Les transformées LBT associées à la DCT de H.264/AVC

4.2.1 La LBT en pré- et post-traitements de la DCT de H.264/AVC

Les techniques de transformées à recouvrement présentées en section 2.4 (les Lapped Transforms) sont intéressantes puisque, dans leur version classique (cf section 2.4.3), elles s'appuient sur la DCT pour calculer des coefficients moins corrélés. Dans la version du domaine spatial (cf section 2.4.3), elles proposent des pré- et post-traitements permettant d'améliorer la transformation utilisée (DCT, ondelettes,...).

De plus, cette version dans le domaine temporel peut être complètement réalisée en nombres entiers si l'on utilise le schéma en étages lifting (cf fig. 4.7) couplé par exemple à la BinDCT ou à l'IntDCT.

Notre travail nous a conduit à insérer dans le codeur H.264/AVC la version en pré- et post-traitements décomposée en étages lifting et couplée à la DCT entière 4×4 de H.264/AVC.

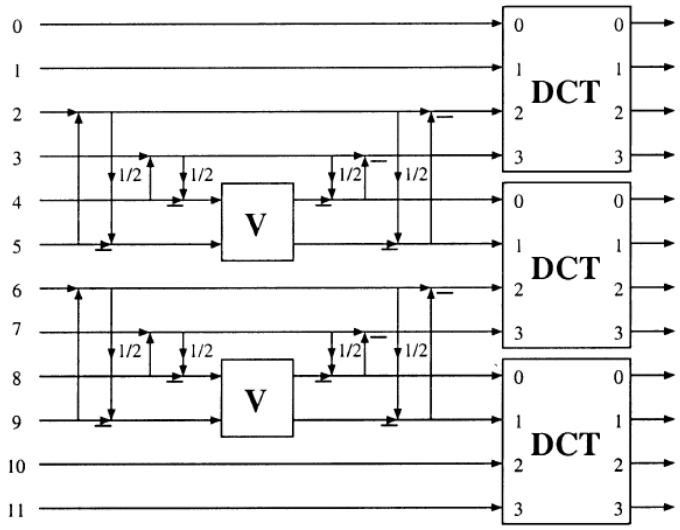


FIG. 4.7 – Schéma de la LBT en pré-traitement sous forme lifting [TLT03]

La matrice V utilisée ici sera une approximation de la matrice $V = JC^{II^T}_{M/2}C^{IV}_{M/2}J$ (cf section 2.4) et sera introduite sous forme lifting comme décrit dans [DT03], soit :

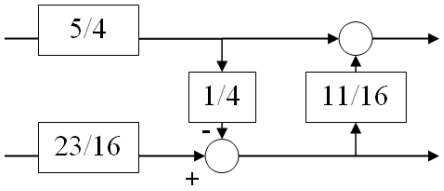


FIG. 4.8 – Schéma lifting de la transformation V du pré-traitement LBT

Cette matrice V est inversible (on est dans le cas biorthogonal), son inverse réalisable en étages

lifting sera donc utilisée pour le post-traitement selon [DT03] :

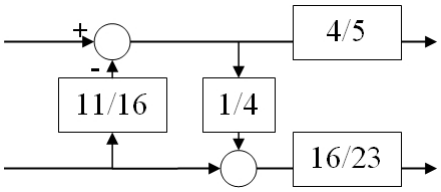


FIG. 4.9 – Schéma lifting de la transformation inverse V^{-1} du post-traitement LBT

De plus, il est intéressant de noter que la DCT 4×4 de H.264/AVC est appliquée successivement sur les lignes, puis les colonnes. Deux applications de cette LBT sont alors possibles. On peut soit réaliser directement les pré- et post-traitements de ces lignes et colonnes avant de transformer le bloc. Soit, on peut réaliser cette LBT en pré- et post-traitements d'abord sur les lignes avant de les transformer par DCT, puis sur les colonnes, afin de tirer avantage de cette LBT dans les deux directions de la transformée DCT. Cette seconde possibilité n'a cependant que peu de sens puisque la LBT définie ici s'applique dans le domaine spatial et non transformé.

4.2.2 Les résultats expérimentaux

Les résultats dans un codeur d'images fixes

Dans un premier temps, afin de valider les performances de cette méthode, un codeur d'images fixes a été mis en place. Ce codeur utilise la DCT 4×4 de H.264/AVC, la quantification H.264/AVC et un codeur entropique de Huffmann, comme celui utilisé dans JPEG [ISO94a].

Ce codeur d'images fixes a été évalué en codant des images naturelles avec ou sans l'aide de cette LBT en pré- et post-traitements. Les résultats de cette expérimentation sont donnés dans le tableau 4.1.

Compression	DCT	DCT+pre/post	DCT	DCT+pre/post
	Barbara (512×512)		Boat (512×512)	
1 : 8	32.58	**34.35**	32.52	**34.02**
1 : 32	25.47	**27.40**	26.56	**27.59**
	Peppers (512×512)		Lena (128×128)	
1 : 8	31.90	**33.82**	32.45	**34.36**
1 : 32	26.53	**28.03**	25.44	**27.12**
	Akiyo (QCIF)		Flower (QCIF)	
1 : 8	32.16	**32.20**	29.95	**30.02**
1 : 32	**26.53**	25.22	22.42	**23.10**
	Foreman (QCIF)			
1 : 8	32.99	**33.91**		
1 : 32	25.93	**26.74**		

TAB. 4.1 – Résultats de la LBT en pré- et post-traitements d'un codage d'images naturelles

Ce tableau montre bien que l'ajout de la LBT en pré- et post-traitements de la DCT H.264/AVC pour le codage d'images naturelles permet d'en améliorer les performances. En moyenne, ce gain est de 1.2 dB et reste positif quelle que soit la compression utilisée (à l'exception de l'image Akiyo compressée à 1 : 32).

Insertion dans le codeur H.264/AVC

Nous avons inséré cette méthode LBT dans le codeur H.264/AVC qui bénéficie d'une optimisation débit-distorsion [WSJ+03] (cf sections 5.3.2 et 5.4.3). Cette optimisation permet de mettre en compétition la DCT avec la DCT pré- et post-traitée par la LBT. Dans ce cas, le codeur sélectionne la plus performante des deux méthodes (avec ou sans LBT) pour le codage de chacun des blocs des images de la séquence vidéo.

Lors du codage H.264/AVC, seuls les blocs causaux (au dessus et à gauche) au bloc courant sont déjà codés/décodés et connus. Cependant, la LBT telle que définie ici (cf fig. 4.7) nécessite la connaissance des blocs causaux et non causaux au bloc courant dans les deux sens (vertical et horizontal).
La seule solution alors envisageable est de limiter cette LBT aux pré- et post-traitements des blocs courants avec uniquement les blocs causaux à ceux-ci, c'est-à-dire en ne conservant que le traitement des coefficients 2, 3, 4 et 5 du schéma 4.7. On aboutit à une "LBT causale" tel qu'illustré sur le schéma de la figure 4.10.

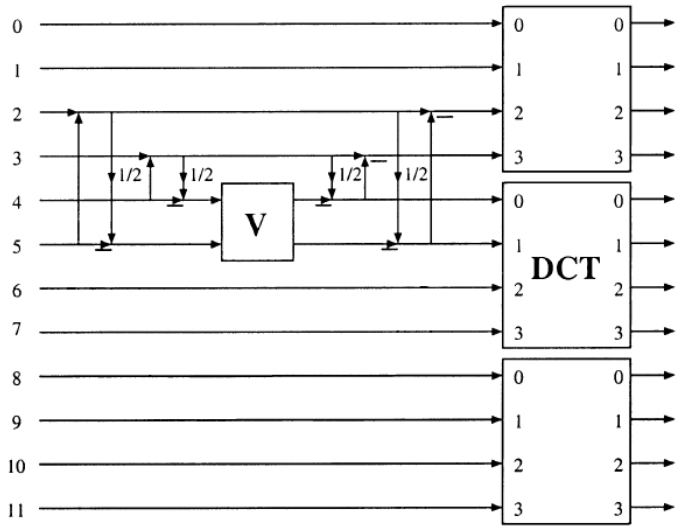


FIG. 4.10 – Schéma de la "LBT causale" en pré-traitement sous forme lifting

De plus, H.264/AVC utilise déjà un post-traitement des blocs : le "Deblocking Filter" [Ric03c] (cf Annexes A.3.7) qui agit à la frontière des blocs afin de lisser les effets de blocs. Sa structure est très proche du post-traitement de la LBT décrite ici, réduisant ainsi son impact.

Les résultats de l'insertion de la LBT en pré- et post-traitements de la DCT H.264/AVC dans sa version "moitié" pour la séquence Flower en CIF à 15 f.p.s. sont présentés sur la figure 4.11.

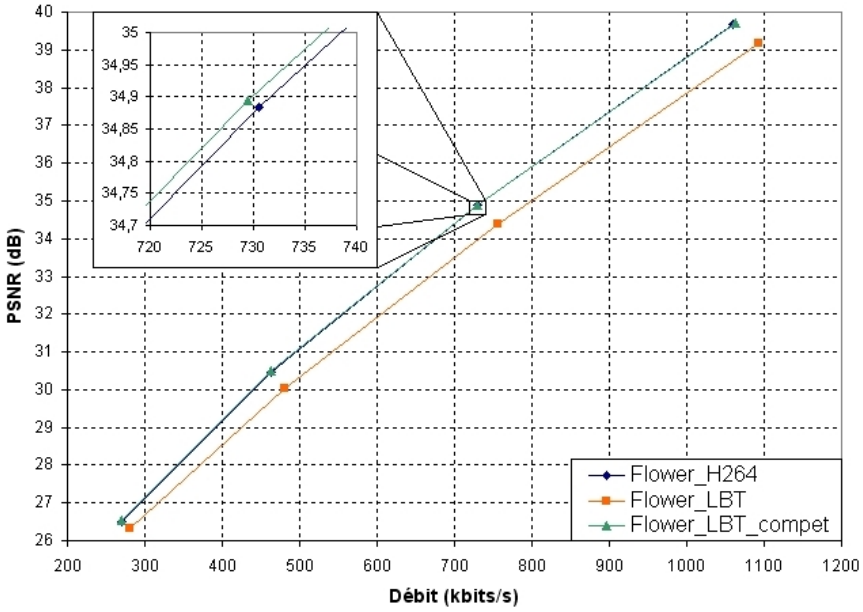


FIG. 4.11 – Résultats de la "LBT causale" associée et en compétition avec la DCT H.264/AVC sur la séquence Flower (CIF 15 f.p.s.)

La courbe associant la LBT en pré- et post-traitements à la DCT H.264/AVC (Flower LBT) a été obtenue en codant chacun des blocs des images en utilisant cette "LBT causale". Cette courbe montre que cette LBT associée à la DCT ne permet pas d'en améliorer les performances.

Cependant, en utilisant l'optimisation débit-distorsion de H.264/AVC [WSJ+03] (cf sections 5.3.2 et 5.4.3), il est possible de mettre en compétition une méthode de LBT en pré- et post-traitements de la DCT H.264/AVC et une méthode de DCT H.264/AVC classique. Le codeur choisit alors pour chacun des blocs la meilleure transformation à appliquer. La compétition permet donc de sélectionner la LBT uniquement pour les blocs où elle est réellement efficace (de 16 à 20% des macroblocs), les autres blocs étant codés classiquement par H.264/AVC. Les résultats de cette compétition sont présentés sur la figure 4.11 (Flower LBT compet).

Cette courbe montre que la compétition permet d'améliorer très légèrement le codage H.264/AVC. En effet, en utilisant ici une mesure de Bjöntegaard [Bjö01] qui calcule (par une différence d'aires sous les courbes) le gain en PSNR et le pourcentage de débit conservé par la compétition par rapport à H.264/AVC seul sur une plage de 4 valeurs de QP (ici 25-30-35-40), on vérifie que la compétition apporte 0.01 dB ou permet de conserver 0.15% du débit par rapport à H.264/AVC.

Les résultats de cette LBT insérée dans le codeur H.264/AVC ne sont pas significatifs, alors qu'ils l'étaient pour JPEG et le codage d'images fixes. La différence majeure entre ces deux codages est que les images traitées dans le codeur H.264/AVC ne sont pas naturelles. Ce sont des erreurs de prédiction (aussi bien en Intra qu'en Inter) (cf Annexes A.3). Nous avons cherché à connaître l'efficacité de la LBT sur ces images résiduelles, en réutilisant le codeur d'images fixes précédent (cf section 4.2.2), mais appliqué à des images résiduelles extraites du codeur H.264/AVC.

Problème lié aux images résiduelles

Comme on a pu le voir en section 1.2.6, les images traitées par le codeur vidéo H.264/AVC ne sont pas naturelles, elles sont toutes prédites : dans le cas Intra, elles subissent une prédiction spatiale, et dans le cas Inter, elles subissent une prédiction temporelle (cf Annexes A.3). Les images à traiter sont donc toutes résiduelles, comme illustré sur la figure 4.12.

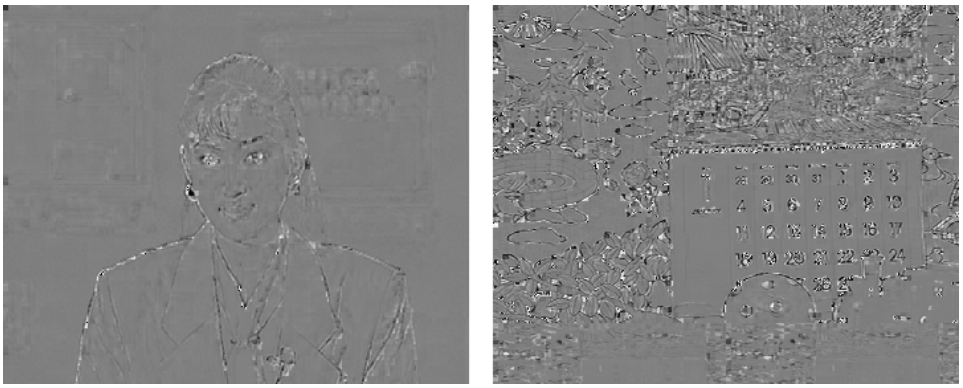


FIG. 4.12 – Exemples d'images résiduelles H.264/AVC après prédiction Intra (recentré sur 128) pour Akiyo (CIF) et Mobile (CIF)

Ces images résiduelles ont été codées à l'aide du codeur d'images fixes présenté précédemment afin d'évaluer les performances de la LBT en pré- et post-traitements sur ces images. Les résultats de cette expérimentation sont donnés dans le tableau 4.2.

Compression	DCT	DCT+pre/post	DCT	DCT+pre/post
	Akiyo (CIF)		Flower (CIF)	
1 : 8	**37.69**	37.64	**32.61**	32.57
1 : 32	**32.99**	32.15	24.92	**25.05**
	Foreman (CIF)		Mobile (CIF)	
1 : 8	**35.61**	35.59	**31.80**	31.77
1 : 32	31.47	**31.53**	24.51	**24.62**

TAB. 4.2 – Résultats de la LBT en pré- et post-traitements d'un codage d'images résiduelles

Ce tableau montre que le codage d'images résiduelles n'est pas amélioré (ou très peu) par l'utilisation de la LBT en pré- et post-traitements de la DCT H.264/AVC. En effet, la LBT dégrade très légèrement la DCT quand la compression est faible (en moyenne 0.03 dB) et ne l'améliore que très peu quand la compression est plus forte (en moyenne 0.1 dB sauf pour Akiyo).

En effet, la LBT réalise une harmonisation des valeurs des pixels du bloc traité, comme présenté sur la figure 2.24, afin d'améliorer la décorrélation de la DCT. Cette harmonisation est alors inefficace sur ces images résiduelles. En reprenant cette harmonisation (cf fig. 2.24) et en l'appliquant aux deux premiers blocs des images Flower naturelle et résiduelle, cette harmonisation avec $s = 2$ de ces données nous donne :

Avant LBT								Après LBT							
153	154	152	**147**	**145**	145	142	136	153	154	152	**148**	**144**	145	142	136
162	162	159	**156**	**159**	164	163	159	162	162	159	**154.5**	**160.5**	164	163	159
163	161	155	**152**	**158**	169	175	175	163	161	155	**149**	**161**	169	175	175
129	146	164	**175**	**178**	174	164	154	129	146	164	**173.5**	**179.5**	174	164	154

TAB. 4.3 – Premiers blocs de l'image naturelle Flower avant et après le pré-traitement LBT (valeurs des luminances des pixels)

Avant LBT								Après LBT							
87	88	85	**92**	**0**	5	6	9	87	88	85	**138**	**-46**	5	6	9
91	92	92	**98**	**1**	7	8	8	91	92	92	**146.5**	**-47.5**	7	8	8
93	104	108	**111**	**9**	10	8	8	93	104	108	**162**	**-42**	10	8	8
92	108	107	**108**	**8**	8	8	7	92	108	107	**158**	**-42**	8	8	7

TAB. 4.4 – Premiers blocs de l'image résiduelle Flower avant et après le pré-traitement LBT (valeurs des luminances des pixels)

Dans ces exemples, le pré-traitement est appliqué selon les lignes des blocs. L'harmonisation associée peut alors être représentée par l'écart-type de chacune des lignes des blocs avant et après traitement.

(a) Image naturelle Flower

Avant LBT		Après LBT	
2.67	3.00	2.33	3.33
2.00	3.33	2.50	2.83
3.67	5.67	4.67	4.67
15.33	8.00	14.83	8.50

(b) Image résiduelle Flower

Avant LBT		Après LBT	
3.67	3.00	19.00	18.33
2.33	2.33	18.50	18.50
6.00	1.00	23.00	18.00
6.00	0.33	22.67	17.00

TAB. 4.5 – Écart-types des lignes des images naturelle et résiduelle Flower avant et après le pré-traitement LBT

Ces tableaux 4.5 montrent bien que la LBT est capable d'améliorer le codage des images naturelles. Par contre, il montre aussi l'inefficacité de cette méthode sur des images résiduelles où ces écart-types deviennent très élevés après traitements.

4.2.3 Conclusion

La méthode proposée de transformée à recouvrement LBT est intéressante parce qu'elle permet d'améliorer la DCT H.264/AVC pour le codage d'images fixes naturelles, quelle que soit la compression utilisée. Elle réalise une harmonisation des données avant transformation afin d'en améliorer l'efficacité.
De plus, chacune des étapes de pré- et post-traitements est réalisable sous forme lifting, permettant ainsi à cette méthode d'être simple et à reconstruction parfaite.

Dans le codeur H.264/AVC, l'optimisation débit-distorsion permet de mettre en compétition

la DCT H.264/AVC associée à la LBT avec cette DCT seule. Cette compétition permet alors à la LBT d'agir localement et ainsi d'améliorer légèrement la DCT H.264/AVC.

Dans cette version en pré- et post-traitements, la LBT est décorrélée de la transformation utilisée. Elle peut donc permettre d'utiliser une autre transformée que la DCT H.264/AVC, telle que des ondelettes de première (cf section 2.2) ou de seconde génération (cf section 3.2), ou bien d'utiliser une DCT modifiée, telle qu'une DCT sous forme lifting (cf section 2.3 et chapitre 4.1) ou une DCT utilisant l'orientation (cf section 3.3), sans avoir besoin de modifier le reste du schéma.

Comme on a pu le voir dans la section 2.4.3, la LBT réalise une harmonisation des données avant LBT, comme illustré sur la figure 2.24. Cependant, dans le codeur H.264/AVC, les images traitées sont résiduelles et cette harmonisation n'est pas efficace, comme représenté sur l'exemple 4.4.

Le codeur H.264/AVC suit un schéma de codage vidéo hybride qui est réalisé en boucle fermée. Il utilise donc, pour réaliser le codage courant, des informations passées, c'est-à-dire des images passées (dans le cas Inter) ou des données de l'image courante déjà codées/décodées (dans le cas Intra).
Cette boucle fermée nous a obligé à faire de même pour la LBT, en n'utilisant pour les traitements que des informations passées de l'image courante, soit en la rendant causale, afin de se conformer au codeur H.264/AVC et de pouvoir l'y insérer.

De plus, ce codeur H.264/AVC possède un filtre limitant les effets de blocs (le "Deblocking Filter" cf Annexes A.3.7) très ressemblant à celui du post-traitement de la LBT. Ce filtre limite l'efficacité de ce post-traitement.

L'insertion de cette méthode LBT dans le codeur H.264/AVC dans sa version causale et mise en compétition avec la DCT H.264/AVC seule permet d'améliorer très légèrement et localement l'efficacité de cette dernière. Cependant, ce faible gain ne justifie pas d'aller plus avant dans nos recherches sur cette méthode de transformée à recouvrement LBT.

Chapitre 5

Exploitation des orientations résiduelles avant la transformée

5.1 Les limites des méthodes existantes

Comme on a pu le voir auparavant, la transformation tient une place importante dans le codage vidéo. Elle permet par des opérations mathématiques de décorréler les données, c'est-à-dire de diminuer la quantité d'information à coder.

Cette transformation peut être de différent type : DCT, des ondelettes de première génération, DCT en lifting (à mi-chemin entre les deux), ou transformée à recouvrement (cf chapitre 2).

Depuis plusieurs années, de nombreux efforts ont été menés afin de rendre ces transformées plus performantes en exploitant, par exemple, les orientations des données traitées (cf chapitre 3).

Parmi les ondelettes de seconde génération (cf section 3.2), les *ridgelets*, les *curvelets* et les bandelettes de première génération sont définies dans le cas continu, elles ne s'adaptent donc pas bien au cas des images et des vidéos numériques (discrètes). De plus, les *curvelets* et les *contourlets* sont redondantes, réduisant ainsi leur efficacité. Par ailleurs, les bandelettes nécessitent une détection de contours afin de définir des flots géométriques et un ré-échantillonnage des données par interpolation. Les *directionlets*, quant à elles, nécessitent une étape de rotation (par interpolation) non réversible.

Les DCT exploitant les orientations (cf section 3.3) souffrent des mêmes problèmes que les ondelettes de seconde génération. La BinDCT orientée nécessite elle aussi une étape d'interpolation. La SA-DCT implique de transmettre une carte de contours en plus des coefficients. De plus, cette transformée, comme la DCT directionnelle, est basée sur plusieurs DCT flottantes de taille variable dont les problèmes liés aux arrondis sont bien connus. Enfin, la DCT directionnelle modifie aussi la forme des blocs traités, la quantification et le parcours des coefficients qui doivent donc être modifiés en conséquence.

D'autres méthodes utilisent directement des rotations [UTY95], mais ces dernières nécessitent des interpolations et génèrent des "trous" dans les coins des images ou blocs traités et donc des pertes.

La méthode que nous proposons ici tient compte des structures géométriques des données sans déformation ni interpolation du signal. Elle consiste en un pré-traitement des images ou des séquences vidéo avant transformation et en un post-traitement après la transformation inverse. Cette méthode s'insère alors dans un schéma de codage vidéo basé bloc, ici un codeur H.264/AVC, sans en modifier la structure, c'est-à-dire que les différents modules : prédiction, transformation, quantification et codage entropique sont conservés tels quels.

5.2 Définition des pré- et post-traitements

Notre pré-traitement vise à exploiter l'orientation des blocs de la transformée sans utiliser de rotation qui nécessiterait une interpolation, mais en effectuant des permutations circulaires entre les pixels. Il effectue un cisaillement ("shear"), simulant ainsi une pseudo-rotation des blocs permettant de les redresser vers l'horizontale ou la verticale.

Les pseudo-rotations de notre pré-traitement s'appliquent au niveau pixel, il est donc intéressant d'utiliser les co-lignes numériques comme proposées dans les *directionlets* de Vetterli et al. (cf section 3.2.3). Ces co-lignes numériques $L(r, n)$ sont définies par r (pente) et n (abscisse à l'origine). On exprime alors un ensemble de points (i, j) (comme indiqué en (3.15)) par :

$$\begin{cases} j = \lceil ri \rceil + n, & pour\ |r| \leq 1, \\ i = \lceil j/r \rceil + n, & pour\ |r| > 1 \end{cases} \tag{5.1}$$

On peut alors définir cette pente r des co-lignes numériques à partir de l'angle θ formé par ces co-lignes avec l'axe des abscisses. On en déduit que :

$$r = \tan\theta \qquad pour\ 0 \leq |\theta| \leq \pi/2 \tag{5.2}$$

Les équations (5.1) deviennent alors :

$$\begin{cases} j = \lceil i \tan\theta \rceil + n, & pour\ |\theta| \leq \pi/4, \\ i = \lceil j/\tan\theta \rceil + n, & pour\ \pi/4 < |\theta| \leq \pi/2 \end{cases} \tag{5.3}$$

Un exemple de ces co-lignes numériques est donné sur la figure 5.1 pour les angles $\theta = -\pi/6$ et $\theta = \pi/3$.

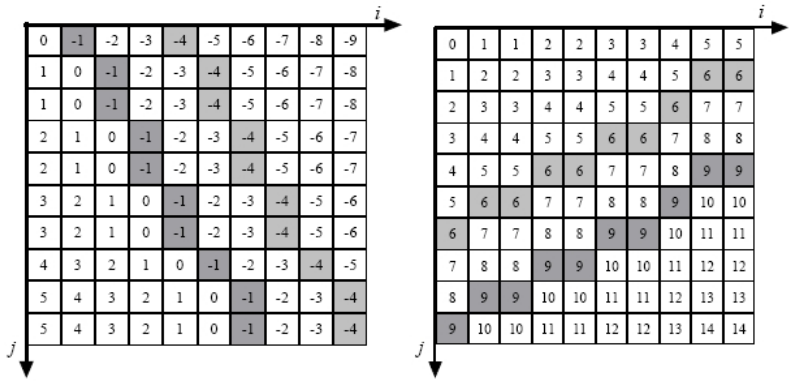


FIG. 5.1 – Exemples de co-lignes numériques pour $\theta = -\pi/6$ et $\theta = \pi/3$

Le pré-traitement que nous proposons réalise une pseudo-rotation des blocs de la transformée en effectuant des opérations de cisaillement ("shear") sur ces co-lignes numériques. Ces opérations de cisaillement [TZ94] sont définies matriciellement par :

$$S_H(\theta) = \begin{bmatrix} 1 & 0 \\ \tan\theta & 1 \end{bmatrix}, \quad \text{pour le cisaillement horizontal} \tag{5.4}$$

$$S_V(\theta) = \begin{bmatrix} 1 & 1/\tan\theta \\ 0 & 1 \end{bmatrix}, \quad \text{pour le cisaillement vertical} \tag{5.5}$$

Les opérations de pseudo-rotation associées à ces cisaillements sont alors définies par :

$$y(\mathbf{n}) = x\left(< S_k(\theta)\mathbf{n} >_N\right) \tag{5.6}$$

où $\mathbf{n}$ est un pixel $\mathbf{n} = (i,j)^T$, k l'orientation du cisaillement $k = H$ si $|\theta| \leq \pi/4$ ou $k = V$ si $\pi/4 < |\theta| \leq \pi/2$ et $< t >_N$ l'opérateur défini par :

$$< t >_N = \lceil t \rceil - \left\lfloor \frac{\lceil t \rceil}{N} \right\rfloor \cdot N \tag{5.7}$$

Cet opérateur permet de garder les indices des coefficients dans le bloc de taille N.

Cette pseudo-rotation permet de redresser les blocs vers l'horizontale (si $|\theta| \leq \pi/4$) ou vers la verticale (si $\pi/4 < |\theta| \leq \pi/2$) sans modifier la forme du bloc. Elle réalise des permutations circulaires au niveau pixel dans le bloc c'est-à-dire des changements d'indices de ces pixels.
Un exemple de chacune de ces pseudo-rotations vers la verticale et vers l'horizontale sont donnés sur la figure 5.2.

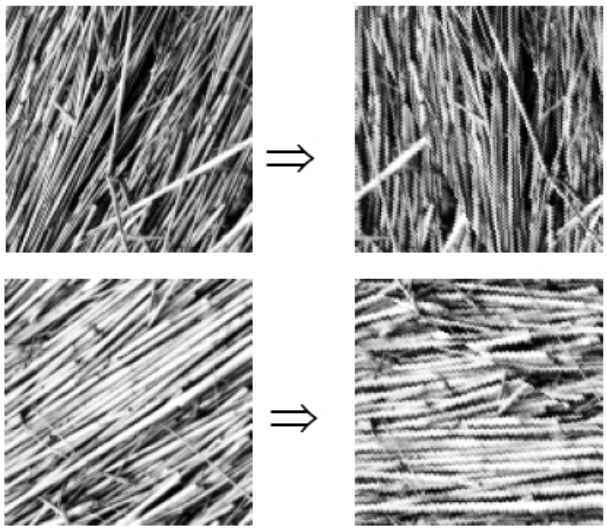


FIG. 5.2 – Exemples de pseudo-rotations vers la verticale et l'horizontale

Le post-traitement de notre méthode réalise les opérations inverses du pré-traitement et permet de rétablir les blocs reconstruits après la transformation inverse. Il effectue les pseudo-rotations d'angles $(-\theta)$ en utilisant le même opérateur (5.6) que le pré-traitement.

5.3 Orientation des blocs de la transformée Intra H.264/AVC

Le chapitre 1 présente les principes de base du codage vidéo. Plus précisément, les sections 1.2 et 1.2.6 présentent le codage vidéo hybride et plus particulièrement le codeur H.264/AVC MPEG-4/AVC. Ce codeur H.264/AVC (cf section 1.2.6 et Annexes A.3) [JVT05] [JVT04] est donc utilisé par la suite comme base d'insertion et de comparaison pour nos travaux.

Ce codeur H.264/AVC traite deux grands types d'images : les Intra et les Inter. Les principales différences entre ces images sont le type de prédiction utilisée et la taille des blocs traités. Cette prédiction peut être spatiale dans le cas Intra ou temporelle dans le cas Inter. Les résidus issus de ces prédictions sont par conséquent eux aussi différents, ils ne sont pas traités tout à fait de la même manière par notre méthode. Ces deux cas (Intra et Inter) sont donc étudiés séparément ici en section 5.3 pour l'Intra et en section 5.4 pour l'Inter.

5.3.1 Les images résiduelles Intra H.264/AVC

Les images Intra H.264/AVC subissent une prédiction spatiale (cf Annexes A.3.2) : chaque bloc 4×4 ou 8×8 est prédit à partir de ces voisins et dans une des 9 directions possibles (cf fig. 5.3), le bloc résiduel à coder est alors la différence entre le bloc original et sa prédiction.

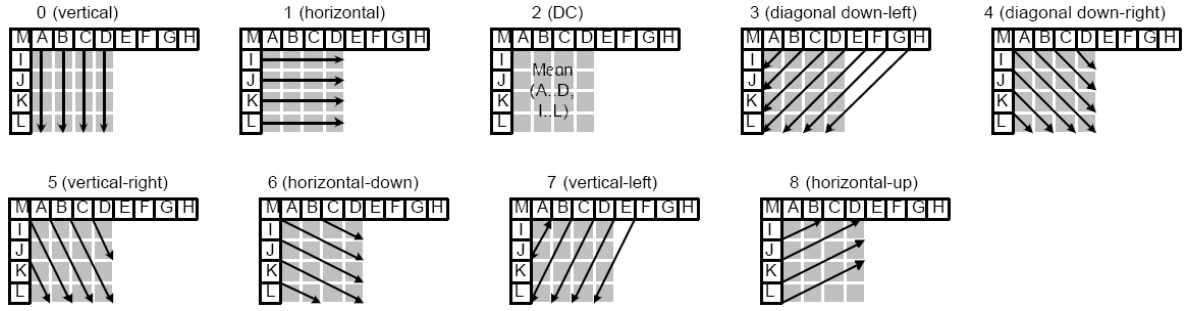


FIG. 5.3 – Les 9 modes de prédiction 4×4 et 8×8 Intra [Ric03b]

Pour les macroblocs 16×16, le même procédé est appliqué, mais il n'existe que 4 directions (cf fig. 5.4).

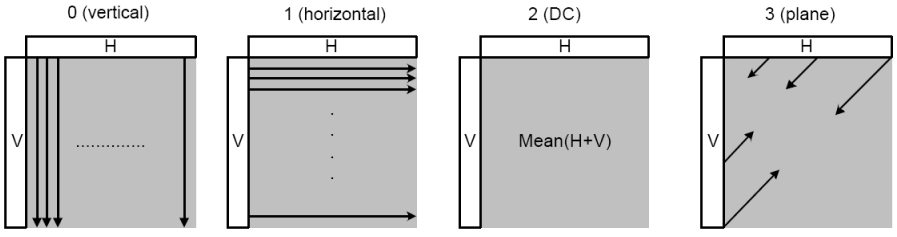


FIG. 5.4 – Les 4 modes de prédiction 16×16 Intra [Ric03b]

Dans tous ces cas, les blocs de coefficients à traiter présentent des motifs réguliers orientés comme représenté sur la figure 5.5 (et fig. 4.12).

FIG. 5.5 – Extrait de l'image Flower (CIF) et son résidu de prédiction Intra H.264/AVC

5.3.2 Orientation des blocs de la transformée

Comme on a vu précédemment dans la section 5.2, notre méthode réalise un pré-traitement des blocs avant la transformée. Ce pré-traitement correspond à une pseudo-rotation des blocs d'un angle prédéfini θ réalisée par cisaillement ("shear"), c'est-à-dire des permutations circulaires appliquées au niveau pixel.

Les blocs 4×4

Dans le cas des blocs 4×4, nous avons défini sept états différents qui correspondent à sept co-lignes numériques. Elles sont prédéfinies par l'orientation du bloc (θ) et sont associées à des permutations circulaires.

A l'état 0, aucune opération n'est à effectuer parce que soit les blocs sont non-orientés (leur direction est horizontale ou verticale ($\theta = 0$ ou $\pm\pi/2$)), soit ils n'ont pas de direction acceptable : leurs orientations sont trop éloignées (plus de $\pm 3^o$) des angles prédéfinis pour être associées à un des autres états.
Les autres états spécifient les blocs dont la direction θ est proche (moins de $\pm 3^o$) des angles : $\pm 27^o$, $\pm 45^o$ et $\pm 63^o$, soit pour des cisaillements :

$$S_{H_{4\times4}}(\theta) = \begin{bmatrix} 1 & 0 \\ \pm 1/2 \text{ ou } \pm 1 & 1 \end{bmatrix}, \quad \text{pour } \theta = \pm 27^o \text{ ou } \pm 45^o \text{ respect.} \tag{5.8}$$

$$S_{V_{4\times4}}(\theta) = \begin{bmatrix} 1 & \pm 1/2 \\ 0 & 1 \end{bmatrix}, \quad \text{pour } \theta = \pm 63^o \tag{5.9}$$

Pour chacun de ces états, des permutations circulaires sont appliquées au niveau pixel afin de simuler une pseudo-rotation par cisaillement. Ces permutations permettent de s'affranchir d'une étape d'interpolation inhérente à tout processus réel de rotation (matricielle [UTY95]). De plus,

par ces simples réarrangements de pixels nous simulons une rotation sans créer de trous dans les coins des blocs.

Dans l'état 1 (cf fig. 5.6, en haut à gauche), une permutation circulaire est réalisée sur les deux premières colonnes, et dans l'état 2, son opposé, sur les deux dernières colonnes. Dans les états 5 et 6, on applique les mêmes permutations que dans les états 1 et 2 mais sur les lignes. Les états 3 et 4 utilisent des réarrangements de pixels plus complexes : l'état 3 correspond à des permutations circulaires appliquées sur la première et la dernière colonne, puis à la même permutation que celle de l'état 1. L'état 4 est similaire à l'état 3 mais les opérations sont réalisées sur les lignes : la première et la dernière ligne sont réarrangées avant de subir la permutation de l'état 2 (cf fig. 5.6).

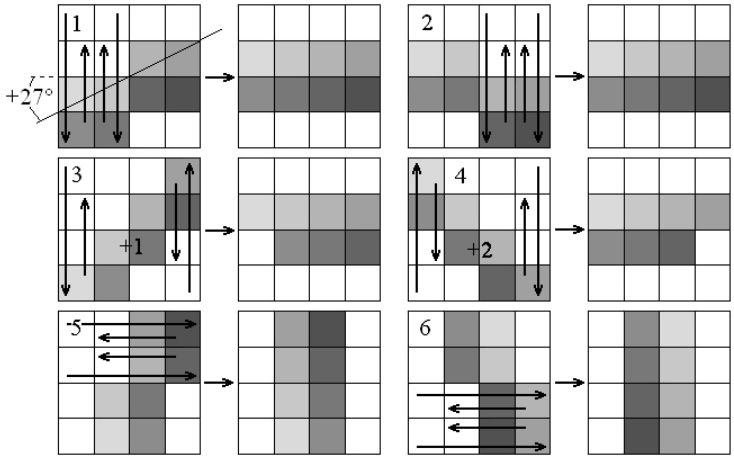


FIG. 5.6 – Permutations circulaires du cas 4×4

Cette figure 5.6 montre que notre pré-traitement redresse bien les blocs vers l'horizontale ou la verticale. Ces permutations circulaires permettent donc de simuler une rotation réelle sans ses désavantages.

Les blocs 8×8

Comme dans le cas 4×4, on peut définir 15 états d'orientation pour les blocs 8×8 et la DCT 8×8 [JVT04] en se basant sur les co-lignes numériques. Ils sont définis (sauf l'état 0) par leur orientation θ telle que : $\theta = \pm 14^o$, $\pm 27^o$, $\pm 37^o$, $\pm 45^o$, $\pm 53^o$, $\pm 63^o$ et $\pm 76^o$, soit pour des cisaillements :

$$S_{H_{8\times 8}}(\theta) = \begin{bmatrix} 1 & 0 \\ k_{H_{8\times 8}} & 1 \end{bmatrix} \tag{5.10}$$

$$S_{V_{8\times 8}}(\theta) = \begin{bmatrix} 1 & k_{V_{8\times 8}} \\ 0 & 1 \end{bmatrix} \tag{5.11}$$

avec $k_{H_{8\times 8}} = \pm 1/4$, $\pm 1/2$, $\pm 3/4$ ou 1 pour $\theta = \pm 14^o$, $\pm 27^o$, $\pm 37^o$ ou $\pm 45^o$ respectivement, et $k_{V_{8\times 8}} = \pm 3/4$, $\pm 1/2$ ou $\pm 1/4$ pour $\theta = \pm 53^o$, $\pm 63^o$ ou $\pm 76^o$ respectivement.

Cependant, afin de limiter le coût de codage de l'information d'orientation qui dépend du nombre d'états définis, nous avons réalisé une étude statistique sur l'utilisation de ces états dans

le codeur H.264/AVC. Cette étude a été menée sur un ensemble de séquences (Akiyo, Flower, Foreman, Mobile,...) et une moyenne a été réalisée. Elle est illustrée sur la figure 5.7, elle reflète bien l'utilisation des ces états puisque chaque histogramme généré (pour chacune des séquences testées) a la même forme que cette moyenne.

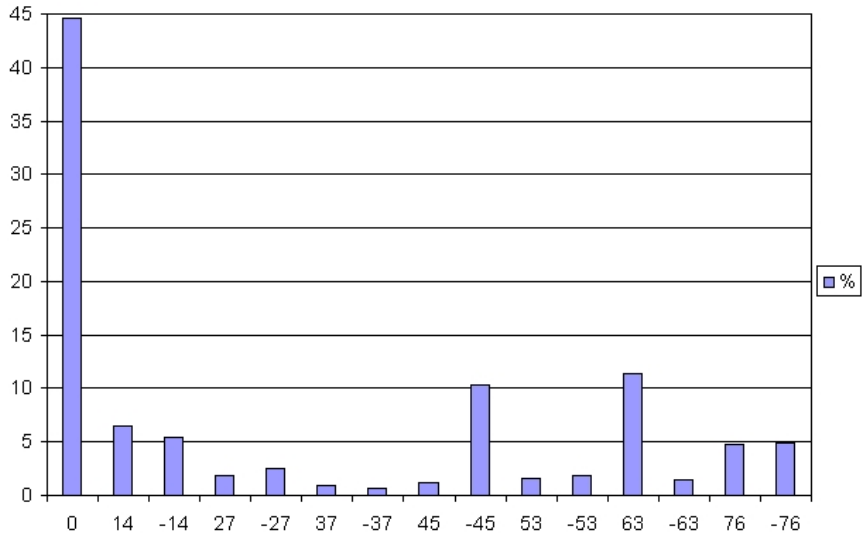


FIG. 5.7 – Fréquence d'utilisation des états 8×8

Cette étude a permis de limiter le nombre d'états d'orientation à seulement 9 (les plus utilisés statistiquement) : 0, $\pm 14^o$, $\pm 45^o$, $\pm 63^o$ et $\pm 76^o$ (cf fig. 5.8).

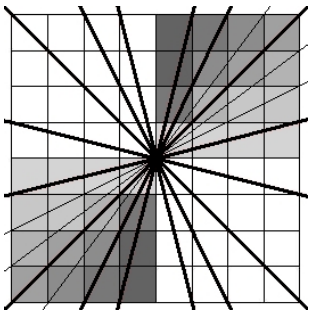


FIG. 5.8 – Orientations conservées dans le cas 8×8 (en gras)

Dans le cas 8×8, comme dans le cas 4×4, l'état 0 reflète les blocs non-orientés et les blocs dont l'orientation est trop éloignée (plus de $\pm 3^o$) des angles de rotation prédéfinis.

Et dans chacun des autres états, les orientations sont associées à des réarrangements des pixels appliqués aux blocs par des permutations circulaires semblables à celles du cas 4×4, mais étendues à des blocs de taille 8×8.

Les blocs 16×16

Comme dans les cas 4×4 et 8×8, on peut définir 31 états d'orientation pour les blocs 16×16 ou macroblocs selon les co-lignes numériques d'angles : $\theta = \pm 7^o$, $\pm 14^o$, $\pm 20^o$, $\pm 27^o$, $\pm 32^o$, $\pm 37^o$, $\pm 41^o$, $\pm 45^o$, $\pm 48^o$, $\pm 53^o$, $\pm 58^o$, $\pm 63^o$, $\pm 69^o$, $\pm 76^o$ ou $\pm 83^o$, soit pour des cisaillements :

$$S_{H_{16\times16}}(\theta) = \begin{bmatrix} 1 & 0 \\ k_{H_{16\times16}} & 1 \end{bmatrix} \tag{5.12}$$

$$S_{V_{16\times16}}(\theta) = \begin{bmatrix} 1 & k_{V_{16\times16}} \\ 0 & 1 \end{bmatrix} \tag{5.13}$$

avec $k_{H_{16\times16}} = \pm 1/8$, $\pm 1/4$, $\pm 3/8$, $\pm 1/2$, $\pm 5/8$, $\pm 3/4$, $\pm 7/8$ ou 1 pour $\theta = \pm 7^o$, $\pm 14^o$, $\pm 20^o$, $\pm 27^o$, $\pm 32^o$, $\pm 37^o$, $\pm 41^o$ ou $\pm 45^o$ respectivement, et $k_{V_{16\times16}} = \pm 7/8$, $\pm 3/4$, $\pm 5/8$, $\pm 1/2$, $\pm 3/8$, $\pm 1/4$ ou $\pm 1/8$ pour $\theta = \pm 48^o$, $\pm 53^o$, $\pm 58^o$, $\pm 63^o$, $\pm 69^o$, $\pm 76^o$ ou $\pm 83^o$ respectivement.

Comme dans le cas 8×8, nous avons réalisé ici une étude statistique sur l'utilisation de ces états dans le codeur H.264/AVC afin d'en limiter le nombre et ainsi le coût de codage. Les résultats sont présentés sur la figure 5.9, il s'agit de la moyenne des statistiques obtenues sur un ensemble de séquences.

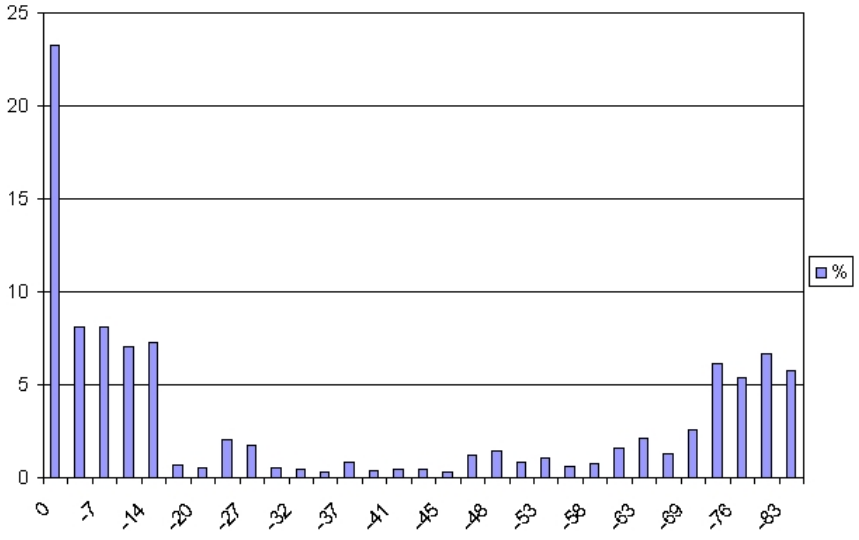


FIG. 5.9 – Fréquence d'utilisation des états 16×16

Il est clair ici que seulement 9 états d'orientation sont principalement utilisés (les plus utilisés statistiquement) : 0, $\pm 7^o$, $\pm 14^o$, $\pm 76^o$ et $\pm 83^o$, les plus proches de l'horizontale et de la verticale. En effet, pour que le mode de codage Intra 16×16 (prédiction 16×16 et codage 4×4) soit plus efficace que le mode de codage 4×4 (prédiction 4×4 et codage 4×4), il faut que la prédiction 16×16 soit efficace, c'est-à-dire que le bloc 16×16 ou les macroblocs soient homogènes. Or, dans le cas de macroblocs homogènes, les orientations sont généralement proches de l'horizontale ou de la verticale.

L'état 0 correspond toujours aux blocs non-orientés et aux blocs dont la direction est trop éloignée (plus de $\pm 3^o$) des angles prédéfinis.

Dans chacun des autres états, ces directions sont associées à des réarrangements des pixels appliqués aux macroblocs grâce à des permutations circulaires comparables à celles des cas 4×4 et 8×8, mais étendues aux blocs de taille 16×16.

Nous avons choisi ici d'orienter directement les macroblocs, mais alternativement nous aurions pu choisir de les orienter par blocs 4×4. En effet, la transformation appliquée ensuite à ces macroblocs est la DCT 4×4 entière de H.264/AVC. Il est donc indifférent de les redresser par blocs 16×16 ou par blocs 4×4 comme illustré sur la figure 5.10.

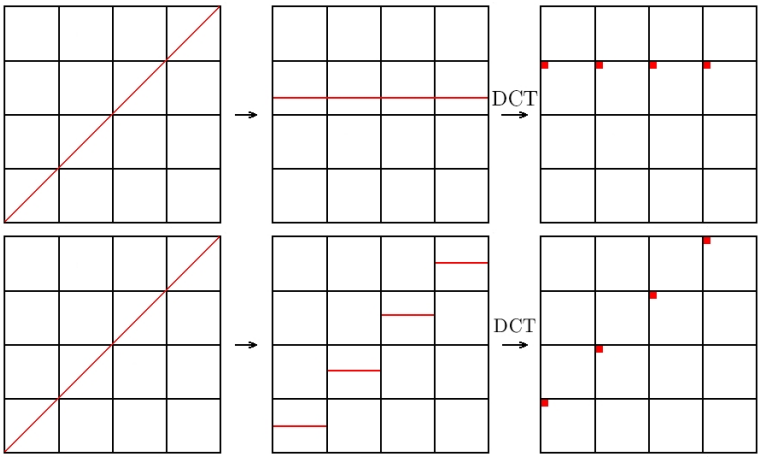


FIG. 5.10 – Exemple de redressements par blocs 16×16 (en haut) ou 4×4 (en bas)

Les redressements 4×4 et 16×16 sont donc identiques, on obtient les mêmes coefficients DCT et en même nombre, seules leurs positions dans le macrobloc diffèrent.

Il est à noter que ces macroblocs subissent après la DCT une seconde transformation. Il s'agit d'une transformée de Hadamard appliquée aux 16 coefficients DC des blocs 4×4 le composant (cf Annexe A.3.4 (A.3)).
L'orientation directe des macroblocs prend alors son sens. Il est évident sur l'exemple de la figure 5.10 que, dans ce cas, les coefficients DC sont eux aussi redressés améliorant ainsi l'efficacité de cette transformée de Hadamard.

Remarque : Cette double transformation appliquée aux macroblocs peut être vue comme une décomposition multi-échelle (dans l'esprit des ondelettes). En effet, à la reconstruction, si on ne peut décoder que les coefficients DC, on obtient alors une version sous-échantillonnée de ces macroblocs.

La sélection des orientations

Afin d'améliorer les transformations DCT 4×4 et 8×8 grâce à notre méthode d'orientation des blocs résiduels Intra par pré- et post-traitements, il faut sélectionner la bonne orientation pour chacun des blocs de la séquence vidéo traitée.

Cette sélection est basée sur l'optimisation débit-distorsion de H.264/AVC (RDO "Rate-Distortion Optimization") [JVT05] [WSJ+03] [WSBL03] [TR03]. Elle consiste à tester tous les modes de codage disponibles et à coder les macroblocs avec le plus performant, c'est-à-dire celui

qui permet d'obtenir la plus faible distorsion pour un débit donné ou le meilleur débit pour une distorsion donnée.

Le coût de codage se calcule pour un macrobloc de taille 16×16 uniquement et dépend de deux variables : le débit et la distorsion.
Le débit d'un macrobloc est la somme des débits des blocs le composant. Ces débits sont eux-mêmes la combinaison des débits des coefficients et de l'en-tête de bloc contenant, le cas échéant, les informations d'orientation.
La distorsion est toujours la distorsion globale du macrobloc quelle que soit la taille des blocs utilisée pour le codage. Elle est donnée par l'erreur quadratique du macrobloc reconstruit selon :

$$D = \sum_{i=0}^{15} \sum_{j=0}^{15} \left(I_{MB}(i,j) - \hat{I}_{MB}(i,j) \right)^2 \tag{5.14}$$

où $I_{MB}(i,j)$ est le pixel (i,j) du macrobloc original et $\hat{I}_{MB}(i,j)$ son correspondant dans le macrobloc reconstruit.

Pour le codage Intra d'un macrobloc, chaque taille de blocs (16×16, 8×8 et 4×4) est testée avec tous les modes de prédiction Intra et tous les états d'orientation disponibles. En d'autres termes, pour chaque taille de blocs, au lieu de coder chaque résidu de prédiction (4 dans le cas 16×16, et 9 dans les cas 8×8 et 4×4) une seule fois, notre méthode le code autant de fois qu'il y a d'états d'orientations possibles (9 dans les cas 16×16 et 8×8, et 7 dans le cas 4×4), comme illustré sur l'organigramme de la figure 5.11

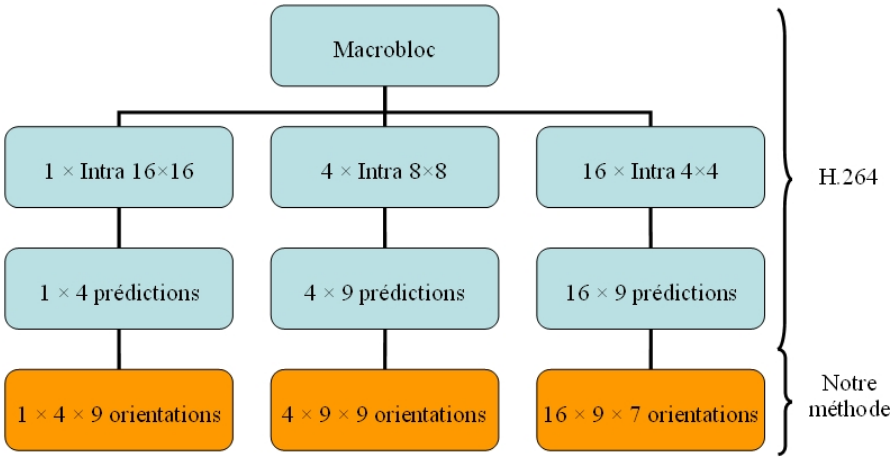


FIG. 5.11 – Déroulement du codage d'un macrobloc Intra H.264/AVC

Ces solutions sont alors comparées afin d'en isoler la meilleure combinaison (taille de bloc, prédiction(s), orientation(s)) au sens débit-distorsion. Cette combinaison sera utilisée, par la suite, pour le codage réel du macrobloc.

Par rapport à un codeur classique, notre pré-traitement ne fait qu'ajouter des combinaisons à tester. Cette méthode de sélection des orientations est efficace, mais complexe en nombres d'évaluations de couples débit-distorsion. Cette complexité ne touche cependant ici que la sélection des modes de prédiction Intra qui ne représente qu'une partie du codage H.264/AVC. La complexité globale du codeur n'en est que peu affectée.

Résultats expérimentaux

Cette méthode de pré- et post-traitements des blocs résiduels Intra 4×4, 8×8 et 16×16 a été insérée dans un codeur H.264/AVC afin d'en vérifier les performances, comme représenté sur le schéma 5.12 (blocs Pré-t^t, Post-t^t et Information Orientation).

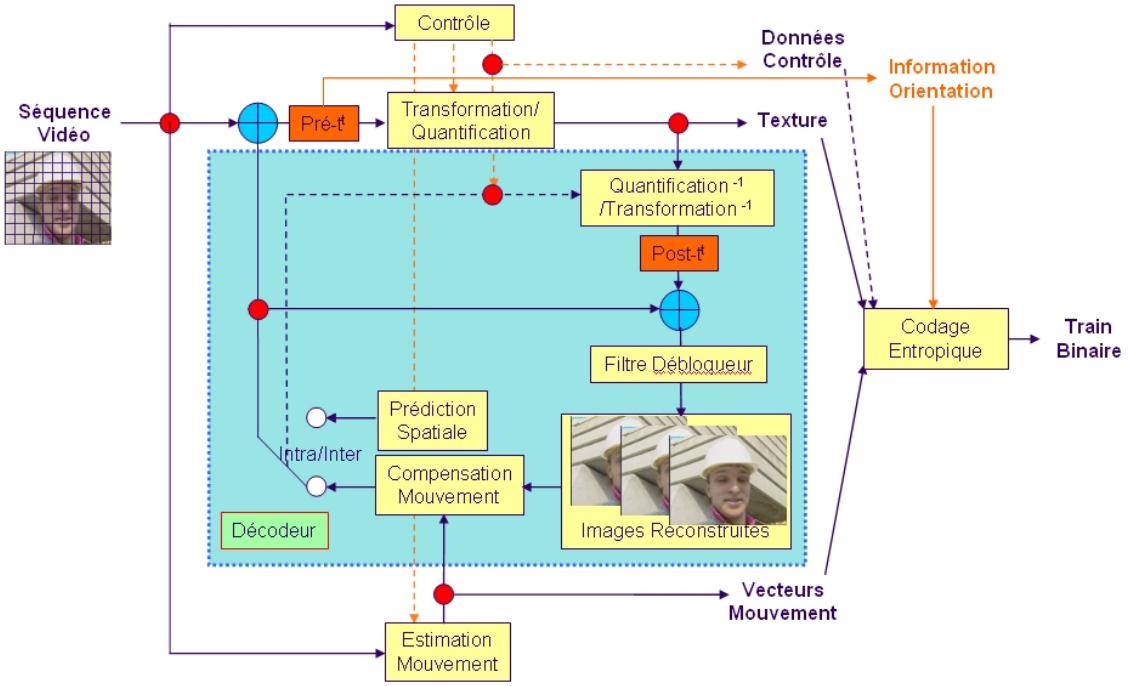
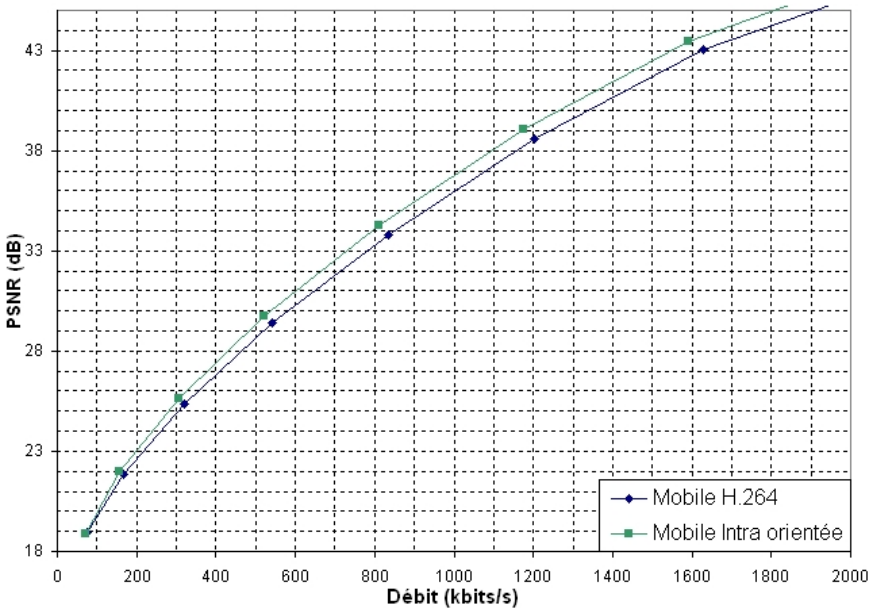


FIG. 5.12 – Schéma de l'insertion de nos pré- et post-traitements dans le codeur H.264/AVC

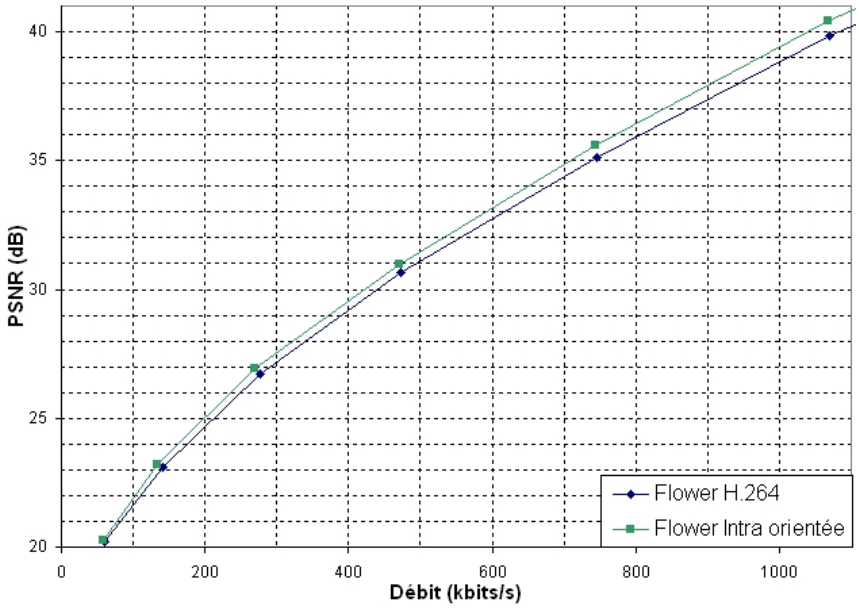
Dans un premier temps, les informations d'orientation de chacun des blocs ne sont pas codées, ce sera le sujet de la section suivante 5.3.3. Nous cherchons seulement pour l'instant à vérifier que notre méthode d'orientation permet effectivement d'améliorer le codage Intra H.264/AVC.

Notre méthode s'insère dans le JM10 [jm06] fourni par le JVT (Joint Video Team de l'ISO et de l'ITU (cf Annexe A.1)). Les tests ont été réalisés en "High Profile" et à "Level 4.0" permettant ainsi l'utilisation de la DCT entière 8×8 [JVT04] et de CABAC [JVT05] [TR03] [WSBL03] [SWS03] [Ric02a] (cf Annexe A.3) pour le codage des macroblocs.
Ces tests ne s'appliquent donc qu'aux images résiduelles de luminances Intra. Ils sont obtenus en faisant varier le pas de quantification (QPI) de ces images.

Les expérimentations ont été menées sur un ensemble de séquences CIF à 15 f.p.s. et SD à 60 f.p.s.. Les résultats de deux de ces séquences : Mobile et Flower en CIF à 15 f.p.s., sont illustrés pour les bas débits sur la figure 5.13 et pour les hauts débits sur la figure 5.14.

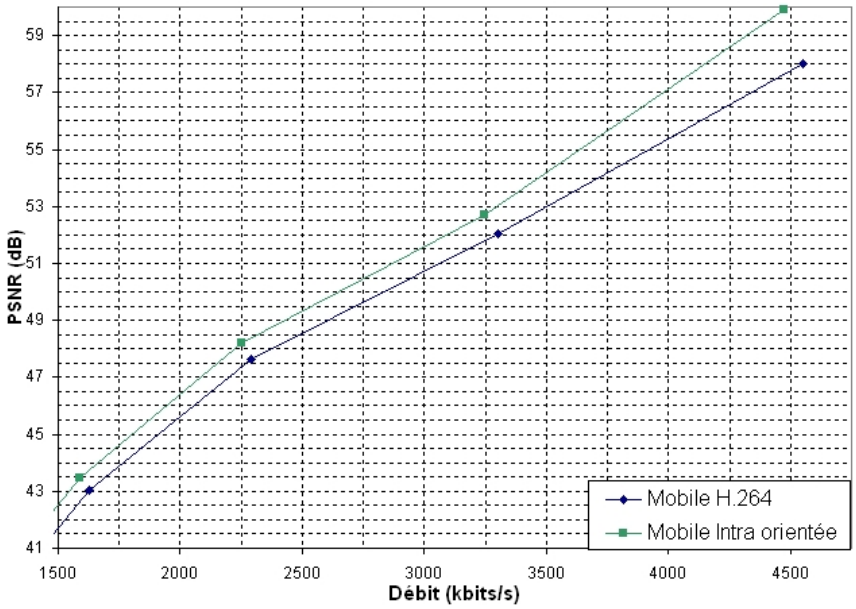


(a) Séquence Mobile

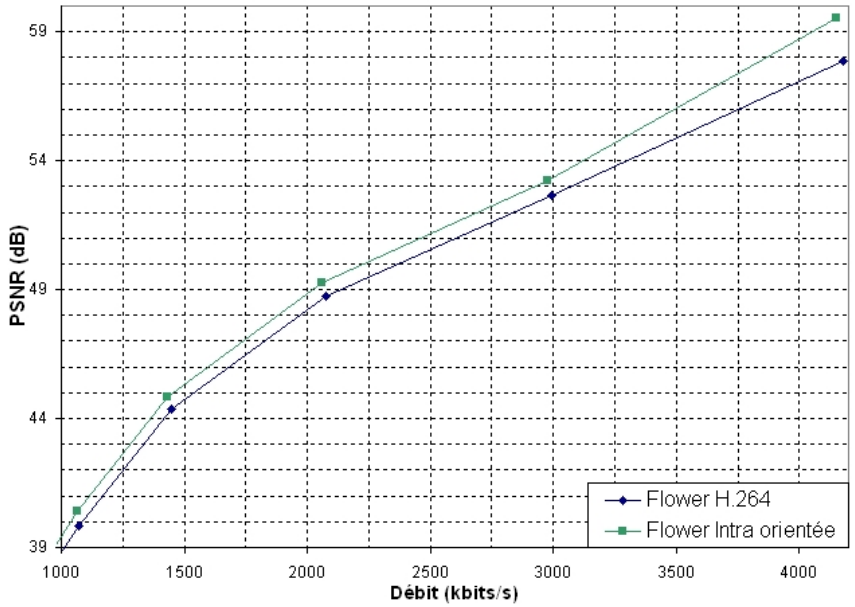


(b) Séquence Flower

FIG. 5.13 – Exemples de résultats Intra dans les bas débits ($<$ 2 Mbits/s)



(a) Séquence Mobile

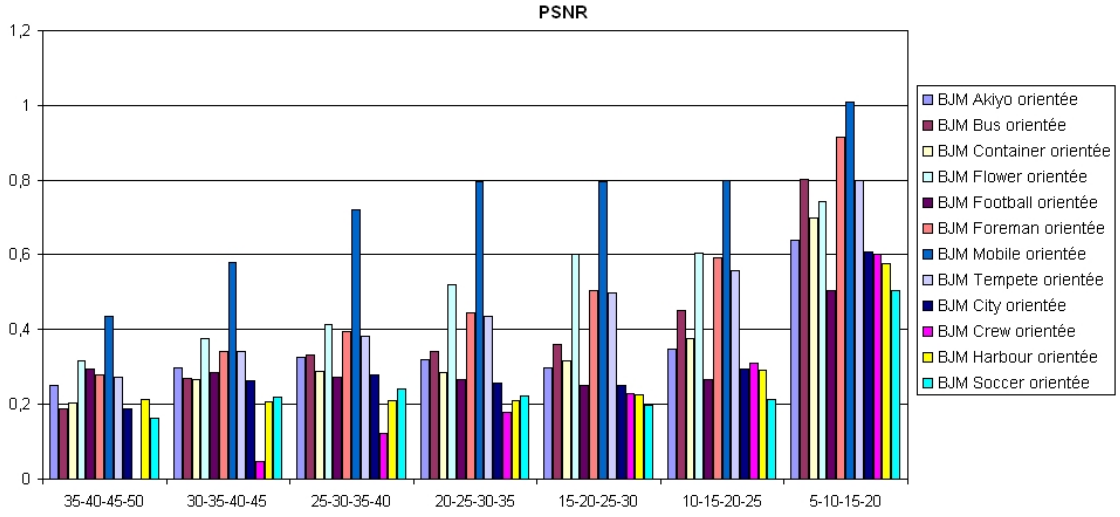


(b) Séquence Flower

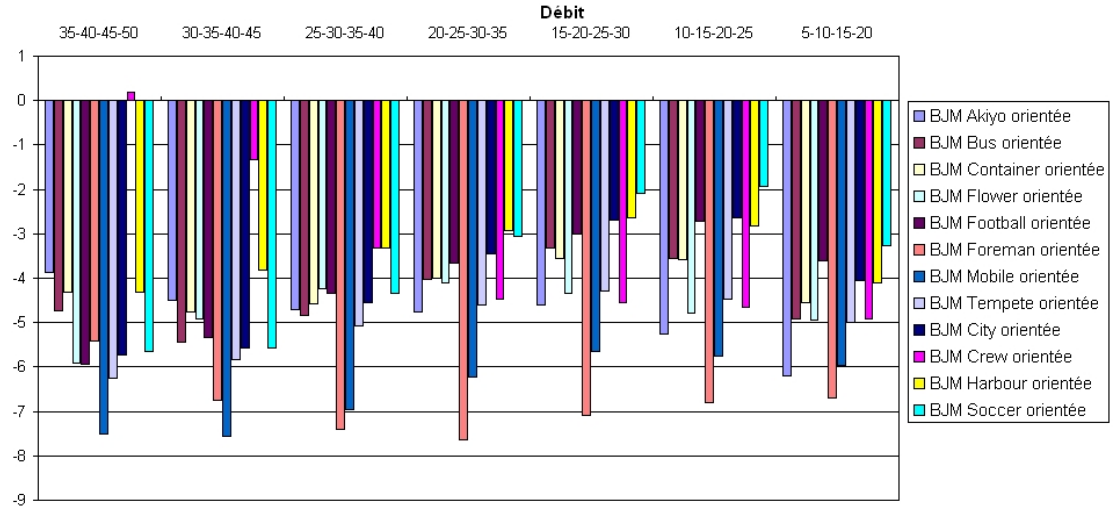
FIG. 5.14 – Exemples de résultats Intra dans les hauts débits (> 1 Mbits/s)

Ces courbes 5.13 et 5.14 illustrent bien que notre méthode d'orientation par pré- et post-traitements des coefficients résiduels Intra H.264/AVC permet d'en améliorer le codage aussi bien dans les hauts débits (cf fig. 5.14) que dans les bas débits (cf fig. 5.13). Ce gain est croissant avec le débit et atteint pratiquement 2 dB pour la séquence Mobile (cf fig. 5.14(a)) et plus de 1.5 dB pour la séquence Flower (cf fig. 5.14(b)).

L'ensemble des résultats peut être résumé en utilisant la métrique de Bjöntegaard [Bjö01]. Cette métrique calcule sur des plages de 4 QP (ici QPI) successifs les aires entre les deux courbes à comparer et définit à partir de ces aires les différences en PSNR et en débit. Ces différences permettent alors d'obtenir le gain en PSNR et le pourcentage de débit conservé par l'une des méthodes comparées.
Ces résultats peuvent alors, selon cette métrique de Bjöntegaard, être représentés par des histogrammes de gain en PSNR et en pourcentage de débit, comme présenté sur la figure 5.15.



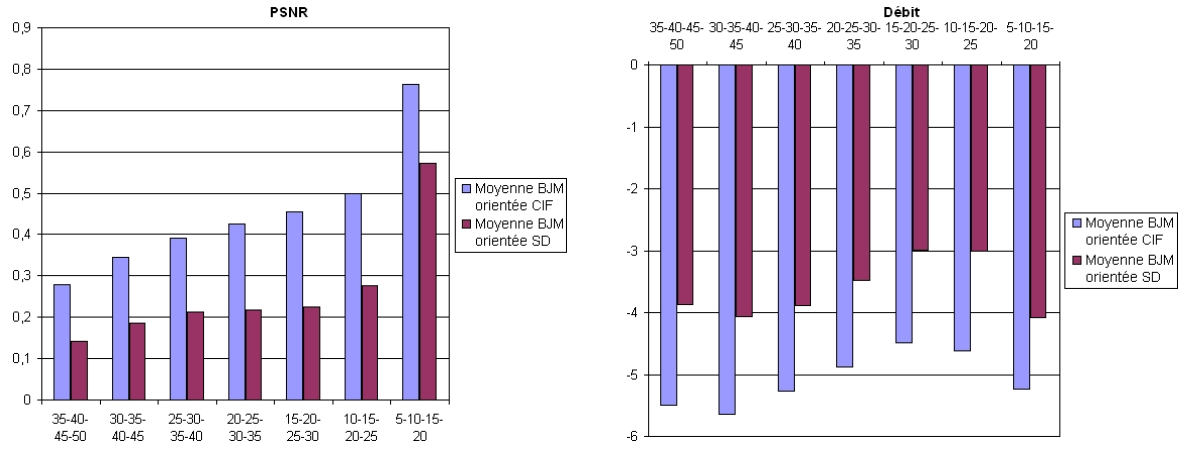
(a) Gain en PSNR



(b) Pourcentage de débit conservé

FIG. 5.15 – Résultats Intra selon la métrique de Bjöntegaard

Des histogrammes moyens des gains en PSNR et débit peuvent alors être calculés pour les séquences CIF et SD selon cette même métrique, comme illustré sur la figure 5.16.



(a) Gain en PSNR

(b) Pourcentage de débit conservé

FIG. 5.16 – Moyenne des résultats Intra selon la métrique de Bjöntegaard

Ces histogrammes permettent d'affirmer que notre méthode d'orientation par pré- et post-traitements des images résiduelles Intra H.264/AVC en améliore le codage quelle que soit la séquence traitée (quelles que soient sa taille et sa fréquence).
On obtient par moyennage de ces résultats :

Type de séquence	Gain en PSNR	Pourcentage de débit
CIF à 15 f.p.s.	0.45 dB	-5.09%
SD à 60 f.p.s.	0.26 dB	-3.63%

TAB. 5.1 – Performances moyennes de notre méthode d'orientation par rapport à H.264/AVC

Les performances de notre méthode d'orientation des images résiduelles Intra sont significatives. En effet, le gain en PSNR est d'environ 0.5 dB et notre méthode permet de conserver un peu plus de 5% de débit par rapport à H.264/AVC sur le traitement des séquences CIF à 15 f.p.s.. Ce gain est d'environ 0.25 dB pour les séquences SD à 60 f.p.s. avec une conservation de 3.5% de débit.

Cependant, on peut remarquer que notre méthode est légèrement moins performante sur les séquences de type SD.
En effet, ces séquences contiennent plus d'images (fréquence plus élevée, 60 f.p.s.), images qui sont aussi plus grandes. Il y a donc beaucoup plus de macroblocs à traiter pour ces séquences, et ceux-ci contiennent plus de détails (orientés). Il devient alors moins aisé d'exploiter ces orientations efficacement. Ceci implique que le gain apporté par notre méthode est moins représentatif sur ces séquences que sur les séquences CIF.

5.3.3 Codage de l'information d'orientation

Notre méthode d'orientation en pré- et post-traitements présentée dans les sections précédentes (5.2 et 5.3.2) a montré qu'elle permet d'améliorer significativement le codage des images résiduelles Intra H.264/AVC.

Dans cette section, nous nous attachons à coder les informations d'orientation nécessaire au décodeur pour réaliser la reconstruction.

La prédiction des informations d'orientation

Cas 4×4 et 8×8

Quand un macrobloc est codé en 4×4, il est composé de 16 blocs, et dans le cas 8×8 de 4 blocs. Il est alors possible d'effectuer une prédiction spatiale des informations d'orientation avant codage (comme pour les modes de prédiction Intra de H.264/AVC).

Lorsque l'on code un de ces blocs (C), les états d'orientation des blocs adjacents au-dessus (A) et à gauche (B) sont comparés s'ils sont disponibles sinon ils sont considérés comme nuls. Celui possédant la plus grande valeur définit l'état d'orientation le plus probable pour le bloc à coder (C).

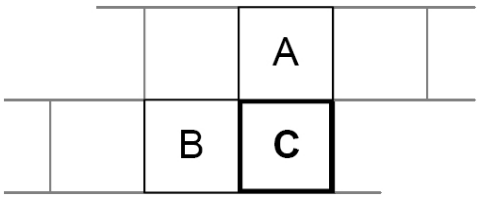


FIG. 5.17 – Voisinage utilisé pour la prédiction des informations d'orientation 4×4 et 8×8

Si l'orientation du bloc à coder (C) est égale à cette orientation la plus probable, on ne transmet qu'un drapeau. Sinon si elle est supérieure à cette orientation la plus probable, on code l'état d'orientation du bloc (C) moins 1 sinon cet état est complètement codé. On a donc au codage :

Si (etat_ A ≥ etat_ B)
orientation_ probable = etat_ A
Sinon orientation_ probable = etat_ B

Si (etat_ C = orientation_ probable)
codage = drapeau
Sinon Si (etat_ C > orientation_ probable)
codage = etat_ C - 1
Sinon codage = etat_ C

A la reconstruction, les blocs (A) et (B) sont déjà décodés et permettent de calculer le mode d'orientation le plus probable pour le bloc à décoder (C). Si on a reçu le drapeau, l'état d'orientation du bloc (C) est alors l'orientation la plus probable sinon il vaut la valeur reçue,

plus 1 si cette valeur est supérieure ou égale à cette orientation la plus probable. On a donc au décodage :

$$
\begin{aligned}
&Si\ (etat\_A \geq etat\_B)\\
&\qquad orientation\_probable = etat\_A\\
&Sinon\ orientation\_probable = etat\_B\\
&\\
&Si\ (decodage = drapeau)\\
&\qquad\qquad etat\_C = orientation\_probable\\
&Sinon\ Si\ (decodage \geq orientation\_probable)\\
&\qquad\qquad etat\_C = decodage + 1\\
&\qquad Sinon\ etat\_C = decodage
\end{aligned}
$$

Cette prédiction permet de diminuer le coût de codage de ces informations d'orientation. En effet, lorsque cette prédiction est efficace, on ne transmet qu'un drapeau (1 bit).
Dans les autres cas, la transmission de l'état d'orientation ou de cet état moins 1 permet de ne coder que 6 états dans le cas 4×4 et 8 états dans le cas 8×8. Par exemple, le dernier état (6 en 4×4 et 8 en 8×8) est codé soit par le drapeau si la prédiction a été efficace, soit par cet état moins 1 (5 en 4×4 et 7 en 8×8) puisqu'il est nécessairement supérieur à l'orientation la plus probable.

Cas 16×16

Dans le cas des macroblocs 16×16, il est difficile d'avoir un voisinage 16×16, on ne peut donc pas appliquer directement la prédiction présentée précédemment.

On voit sur l'histogramme de la figure 5.9 que l'état d'orientation le plus utilisé est l'état 0. Cet état est alors utilisé comme orientation la plus probable pour le macrobloc. Ensuite, on applique la suite de la prédiction présentée précédemment.

Il est intéressant de noter que, dans ce cas, l'état d'orientation à coder est toujours supérieur ou égal à l'orientation la plus probable (0). On transmet donc soit un drapeau soit cet état d'orientation moins 1.
Cette manipulation permet, comme dans les cas précédents, de limiter le nombre d'états d'orientation 16×16 à 8 (3 bits) au lieu des 9 initiaux (4 bits).

Le codage entropique des informations d'orientation prédites

Le codeur arithmétique externe

Afin de réaliser le codage des informations d'orientation précédemment prédites, nous avons utilisé un codeur arithmétique classique externe à H.264/AVC.

Ce codeur arithmétique possède plusieurs options :
- Il peut être équiprobable, c'est-à-dire que tous les symboles ont la même probabilité d'apparition. Ou il peut être adaptatif, auquel cas ces probabilités sont mises à jour en fonction des symboles codés.
- Il peut utiliser ou non une initialisation. Elle permet d'affecter au départ une valeur à chaque probabilité d'apparition des symboles.

L'initialisation permet alors de débuter le codage des informations d'orientation prédites avec

comme probabilités des symboles des valeurs moyennes calculées sur un ensemble de séquences et d'ainsi améliorer les performances du codeur arithmétique.
L'adaptation permet ensuite d'affiner ces probabilités indépendamment pour chacune des séquences traitées et d'améliorer aussi les performances du codeur.

Les statistiques des symboles

L'initialisation de ce codeur est donc très importante, comme on a pu le voir précédemment. Nous avons donc recherché à en définir une bonne valeur pour notre codeur arithmétique et pour chaque type d'informations à coder.

Il est nécessaire ici de définir 4 initialisations, une pour chacun des types de blocs traités (4×4, 8×8 et 16×16) et une supplémentaire pour le drapeau issu de la prédiction des informations d'orientation.

Pour cela, nous avons réalisé des tests de codage des informations d'orientation prédites avec ce codeur arithmétique sans initialisation, soit avec l'équiprobabilité des symboles. De plus, nous avons utilisé l'adaptation afin de recueillir ces probabilités affinées après une passe de codage sur un ensemble de séquences (Akiyo, Bus, Container, Flower, Football, Foreman, Mobile et Tempete en CIF à 15 f.p.s.).

Ces tests nous ont alors permis par moyennage d'obtenir les probabilités d'initialisation pour chaque symbole de chaque type selon :

(a) Blocs 16×16

Symbole	Probabilité
0	0.41
1	0.09
2	0.08
3	0.08
4	0.09
5	0.09
6	0.08
7	0.08

(b) Blocs 8×8

Symbole	Probabilité
0	0.34
1	0.11
2	0.09
3	0.09
4	0.09
5	0.10
6	0.09
7	0.09

(c) Blocs 4×4

Symbole	Probabilité
0	0.52
1	0.11
2	0.09
3	0.09
4	0.09
5	0.10

(d) Drapeau

Symbole	Probabilité
0	0.17
1	0.83

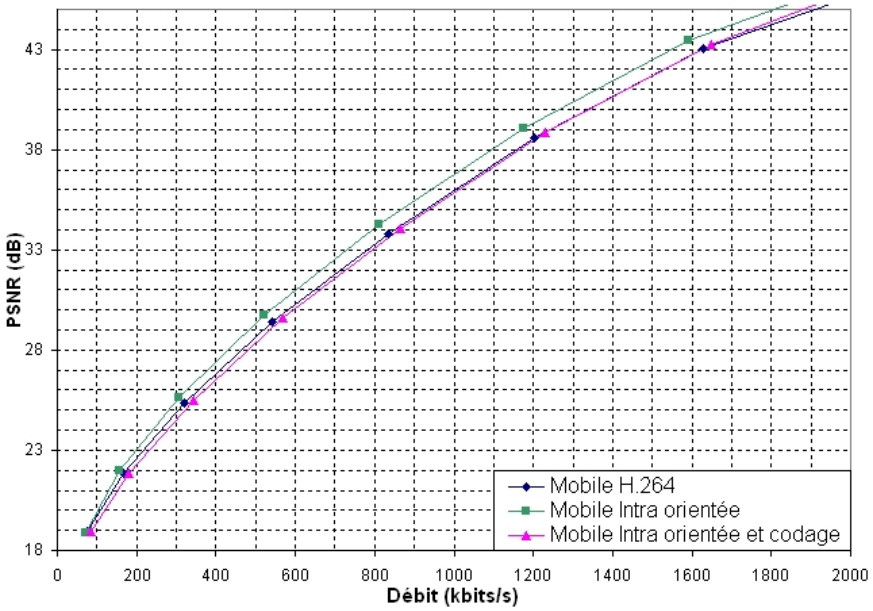
TAB. 5.2 – Probabilités moyennes des symboles suivant chaque type de blocs traités

On vérifie bien dans les tableaux 5.2(a) et 5.2(b) que ces probabilités moyennes des symboles sont homogènes avec les histogrammes des figures 5.9 et 5.7 respectivement.

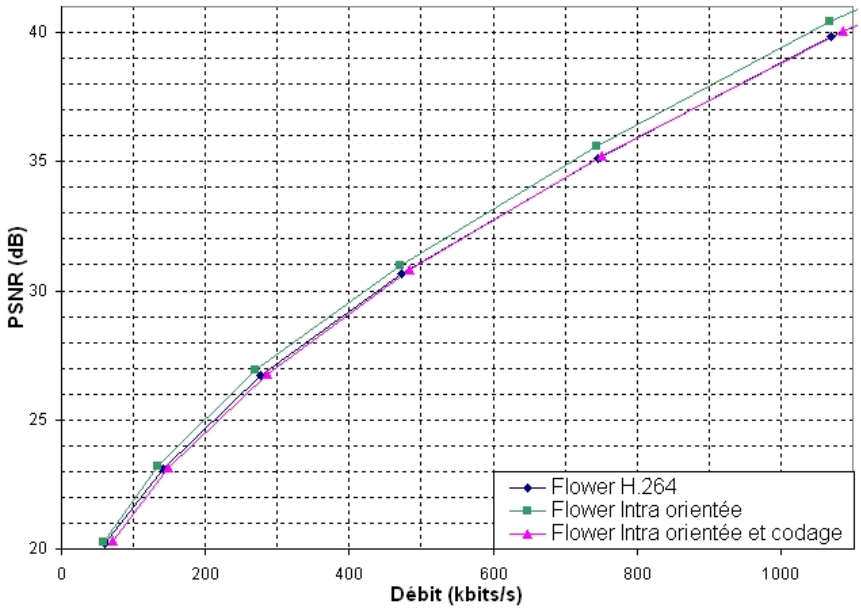
Résultats expérimentaux

Nous avons réalisé ici les mêmes expérimentations que dans la section 5.3.2 en utilisant la même configuration du codeur H.264/AVC, mais en codant cette fois-ci les informations d'orientation nécessaire au décodeur.

Les résultats de deux de ces séquences : Mobile et Flower en CIF à 15 f.p.s., sont illustrés pour les bas débits sur la figure 5.18 et pour les hauts débits sur la figure 5.19.

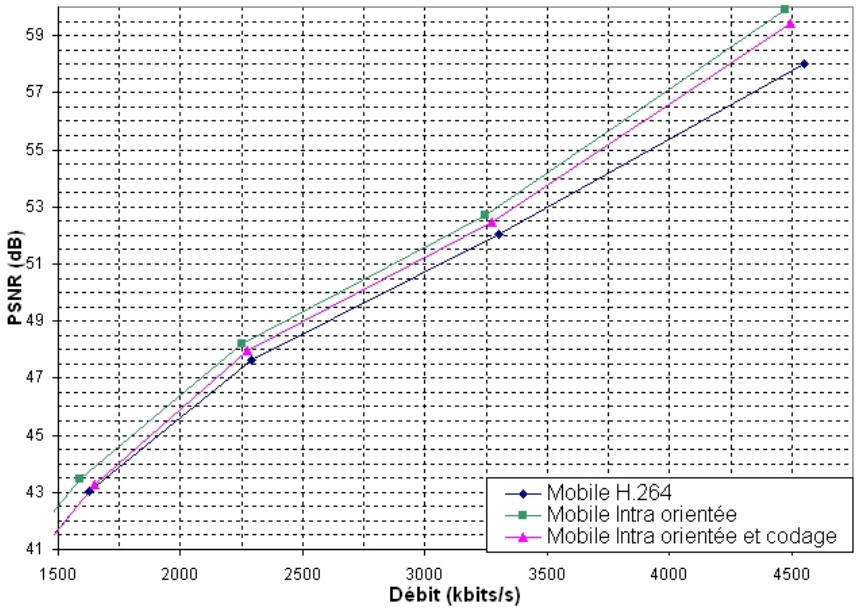


(a) Séquence Mobile

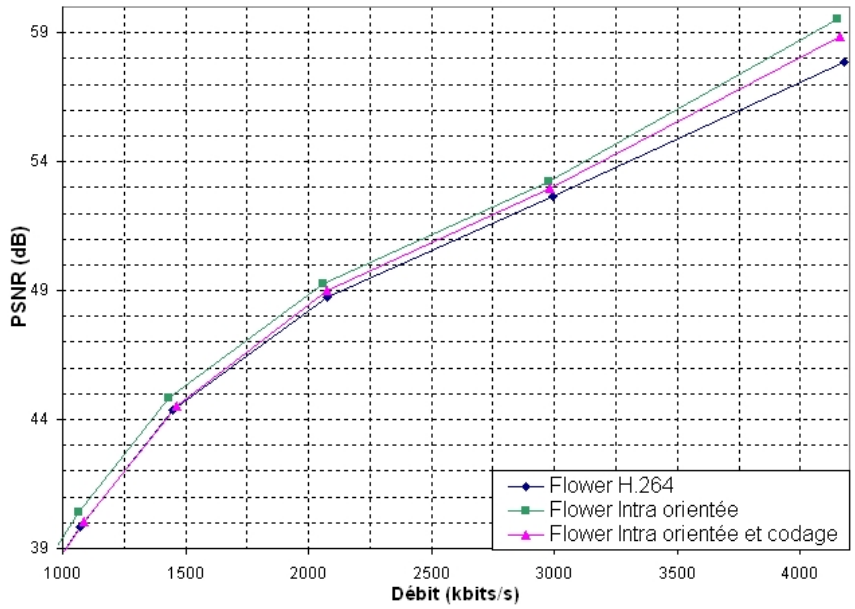


(b) Séquence Flower

FIG. 5.18 – Exemples de résultats Intra dans les bas débits (< 2 Mbits/s) avec le codage des informations d'orientation



(a) Séquence Mobile



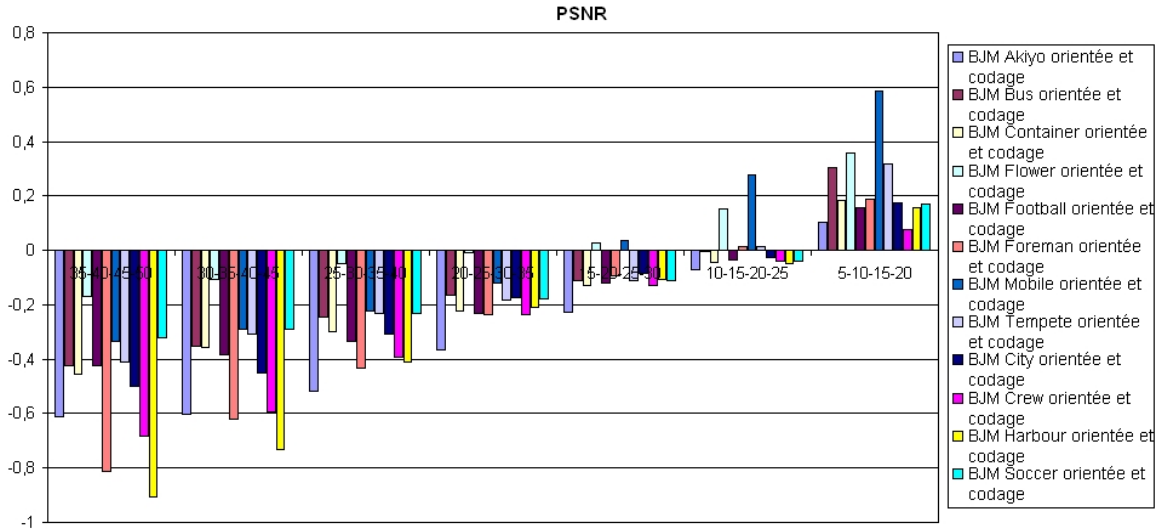
(b) Séquence Flower

FIG. 5.19 – Exemples de résultats Intra dans les hauts débits (> 1 Mbits/s) avec le codage des informations d'orientation

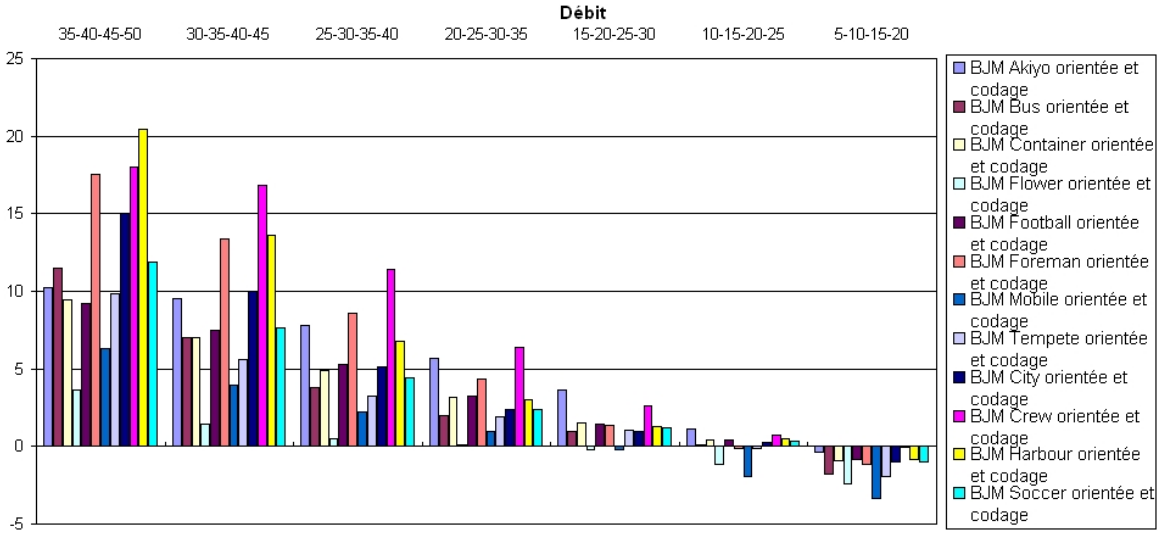
Les quatre courbes des figures 5.18 et 5.19 montrent que le codage des informations d'orientation dégrade considérablement les performances de notre méthode et notamment dans les bas débits (cf fig. 5.18(a) et 5.18(b)).

Cependant, nos pré- et post-traitements semblent toujours améliorer le codage des images résiduelles Intra H.264/AVC dans les hauts débits où le coût des informations d'orientation devient alors moins important que le gain apporté par la méthode (cf fig. 5.19(a) et 5.19(b)).

Afin de mieux interpréter ces expérimentations, l'ensemble des résultats peut ici aussi être résumé en utilisant la métrique de Bjöntegaard sous forme d'histogrammes de gain en PSNR et en pourcentage de débit, comme présenté sur la figure 5.20.



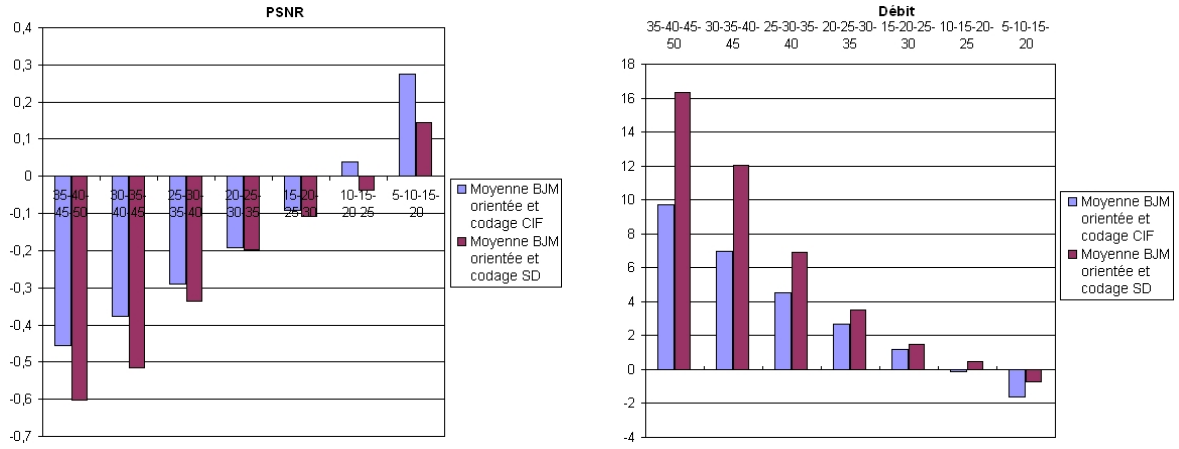
(a) Gain en PSNR



(b) Pourcentage de débit conservé

FIG. 5.20 – Résultats Intra avec le codage des informations d'orientation selon la métrique de Bjöntegaard

Des histogrammes moyens des gains en PSNR et débit peuvent alors être calculés pour les séquences CIF et SD selon cette même métrique, comme illustré sur la figure 5.16.



(a) Gain en PSNR (b) Pourcentage de débit conservé

FIG. 5.21 – Moyenne des résultats Intra avec le codage des informations d'orientation selon la métrique de Bjöntegaard

Ces histogrammes permettent de mettre en évidence un coût élevé de codage des informations d'orientation. En effet, ce coût dégrade les performances de notre méthode en la rendant plus coûteuse que la norme H.264/AVC dans les bas débits, soit pour des QPI supérieur à 25 pour les séquences CIF et à 20 pour les séquences SD. Cependant, dans les hauts débits, soit pour des QPI inférieur à 25 pour les séquences CIF et à 20 pour les séquences SD, notre méthode permet toujours d'améliorer le codage H.264/AVC.

Un calcul d'entropie réalisé sur chacun des types de symboles utilisés, suivant :

$$H = -\sum_{i=0}^{N} p_i \cdot \log_2 p_i \qquad \text{avec } N \text{ le nombre de symboles} \tag{5.15}$$

nous aurait donné, selon les probabilités initiales (cf Tab. 5.2), les bornes inférieures de ce coût de codage des informations d'orientation prédites suivant chaque type (comparées avec celles de l'équiprobabilité entre parenthèses), soit :

$$\begin{aligned}
H_{16\times16} &= 2.63 \text{ bits} \quad (H^{equi}_{16\times16} = 3.17 \text{ bits}) \\
H_{8\times8} &= 2.77 \text{ bits} \quad (H^{equi}_{8\times8} = 3.17 \text{ bits}) \\
H_{4\times4} &= 2.11 \text{ bits} \quad (H^{equi}_{4\times4} = 2.81 \text{ bits}) \\
H_{Drapeau} &= 0.66 \text{ bits} \quad (H^{equi}_{Drapeau} = 1 \text{ bit})
\end{aligned}$$

On aurait alors pu voir d'ores et déjà ici que le codage de ces informations a un coût très élevé quelle que soit la taille du bloc traité.

5.4 Orientation des blocs de la transformée Inter H.264/AVC

5.4.1 Les images résiduelles Inter H.264/AVC

En mode Inter, les images H.264/AVC subissent une prédiction temporelle réalisée par une étape d'estimation de mouvement et une étape de compensation en mouvement (cf Annexes A.3.3). Cette prédiction s'applique au macrobloc 16×16 selon différentes partitions, comme illustré en figure 5.22.

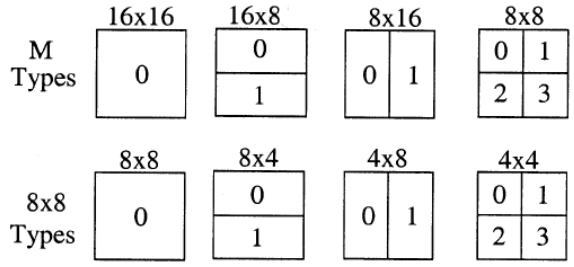


FIG. 5.22 – Les différentes partitions possibles d'un macrobloc Inter [SWS03]

Cette prédiction peut, de plus, utiliser pour son estimation de mouvement, une ou plusieurs images de référence qui peuvent être : backward (image de référence passée), comme représenté sur la figure 5.23, forward (image de référence future) ou les deux.

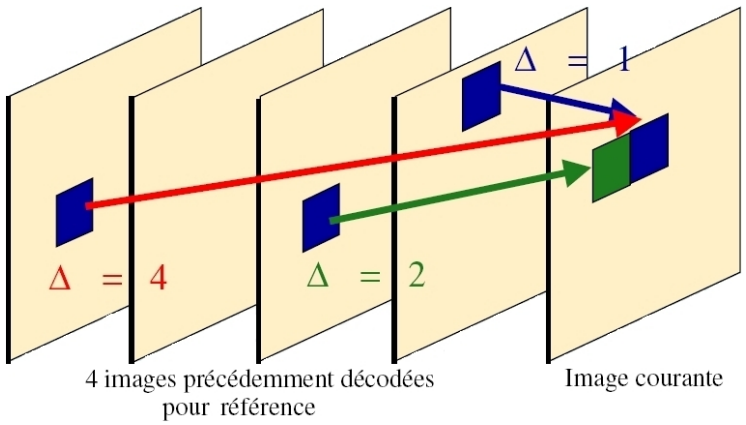


FIG. 5.23 – Les images de référence multiples Inter H.264/AVC [SWS03]

Les blocs de coefficients résiduels générés par cette prédiction temporelle Inter présentent alors encore des motifs réguliers orientés, comme représenté sur la figure 5.24.

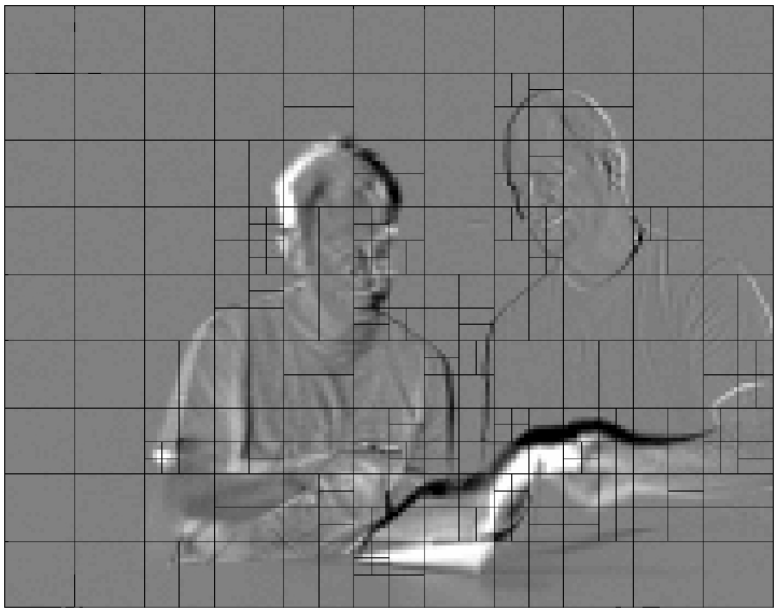


FIG. 5.24 – Image résiduelle Inter H.264/AVC avec sa découpe après prédiction temporelle [Ric03a]

5.4.2 Orientation des blocs de la transformée

Pour les blocs de tailles 4×4, 8×8 et 16×16 en Inter, on applique les mêmes pré- et post-traitements que ceux définis pour le cas Intra (cf section 5.3.2). On utilise les mêmes permutations circulaires.

Par contre, il faut aussi, dans ce mode de codage, définir nos pré- et post-traitements pour toutes les autres tailles de partitions : 16×8, 8×16, 8×4 et 4×8 (cf fig. 5.22).

Partitions 16×8 et 8×16

Les macroblocs sont généralement codés en mode 16×16 s'ils appartiennent à une zone uniforme. Les partitions 16×8 et 8×16 permettent alors d'isoler certaines parties du macrobloc qui ne sont pas homogènes. Par exemple, une partition 16×8 dans la zone uniforme et une seconde partition 16×8 dans une zone non-uniforme qui contient alors généralement un contour proche de l'horizontale (cf fig. 5.26 a)).

Il est donc naturel d'utiliser, pour ces partitions 16×8 et 8×16, les états d'orientation des macroblocs 16×16 (cf section 5.3.2) en réadaptant les permutations circulaires correspondantes à une taille rectangulaire de bloc.
De plus, nous pouvons limiter ces états d'orientation, pour les partitions horizontales 16×8 à ceux proches de l'horizontale : 0, $\pm 7^o$ et $\pm 14^o$, et respectivement pour les partitions verticales 8×16 à ceux proches de la verticale : 0, $\pm 76^o$ et $\pm 83^o$.

On obtient, par exemple, dans le cas d'une partition 16×8 orientée à 14^o un redressement tel qu'illustré sur la figure 5.25.

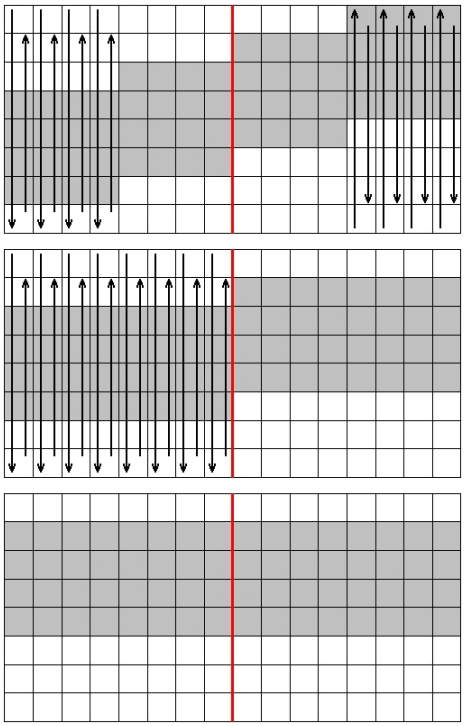


FIG. 5.25 – Exemple d'un redressement 16×8 à 14^o

Remarque : On peut voir, sur cet exemple, que le pré-traitement des partitions 16×8 à 14^o peut se séparer en pré-traitements de deux blocs 8×8. Il en est de même pour les autres orientations possibles de cette partition (la seconde étape de cet exemple correspond à 7^o) et pour les états d'orientation proches de la verticale des partitions verticales 8×16. Cette séparation facilite l'insertion de nos pré- et post-traitements dans le codeur H.264/AVC qui effectue le codage Inter (après prédiction) bloc 8×8 par bloc 8×8.

Partitions 8×4 et 4×8

Comme précédemment, les partitions 8×4 et 4×8 permettent d'isoler les zones uniformes des autres zones (cf fig. 5.26 b)) ou de séparer des parties qui n'ont pas la même orientation (cf fig. 5.26 c)) dans un bloc 8×8.

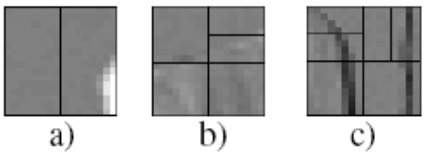


FIG. 5.26 – Exemples de partitionnements Inter H.264/AVC d'un macrobloc

Les états d'orientation utilisés ici sont ceux du cas 8×8 (cf section 5.3.2). Ils sont, ici aussi, limités à ceux proches de l'horizontale pour les partitions horizontales 8×4 : 0, $\pm 14^o$ et $\pm 27^o$, et

à ceux proches de la verticale pour les partitions verticales 4×8 : 0, $\pm 63^o$ et $\pm 76^o$.

L'utilisation des états horizontaux pour les partitions horizontales 8×4 et des états verticaux pour les partitions verticales 4×8 permet, comme pour les cas 16×8 et 8×16, de séparer les traitements de ces partitions en pré- et post-traitements de deux blocs 4×4 pour faciliter leur insertion dans le codeur H.264/AVC.

5.4.3 Sélection des orientations

Comme dans le cas Intra, nous utilisons ici l'optimisation débit-distorsion du codeur H.264/AVC afin de sélectionner nos orientations pour chaque partition (cf section 5.3.2).

Le coût de codage d'un macrobloc 16×16 dépend toujours de son débit et de sa distorsion. Sa distorsion est ici aussi donnée par (5.14), et son débit par la somme des débits des blocs composant ce macrobloc. Ces débits sont toujours la combinaison des débits des coefficients et de l'en-tête du bloc qui contient les informations d'orientation le cas échéant, mais aussi, dans le cas Inter, du débit des vecteurs de mouvement :

$$\begin{aligned} R_{macrobloc} &= \sum_{i=0}^{N} R_{bloc\_i} \qquad \text{avec } N \text{ le nombre de blocs composant ce macrobloc} \\ &= \sum_{i=0}^{N} \left(R_{bloc\_i}^{coef} + R_{bloc\_i}^{en-tete} + R_{bloc\_i}^{mvt} \right) \end{aligned} \tag{5.16}$$

Pour le codage Inter d'un macrobloc, chaque partition résiduelle (16×16, 16×8, 8×16, 8×8, 8×4, 4×8 et 4×4) est testée avec tous les états d'orientation disponibles.

En d'autres termes, au lieu de coder chaque partition résiduelle une seule fois, notre méthode le code autant de fois qu'il y a d'états d'orientations possibles (9 dans les cas 16×16 et 8×8, 5 dans les cas 16×8, 8×16, 8×4 et 4×8 et 7 dans le cas 4×4), comme illustré sur l'organigramme de la figure 5.27.

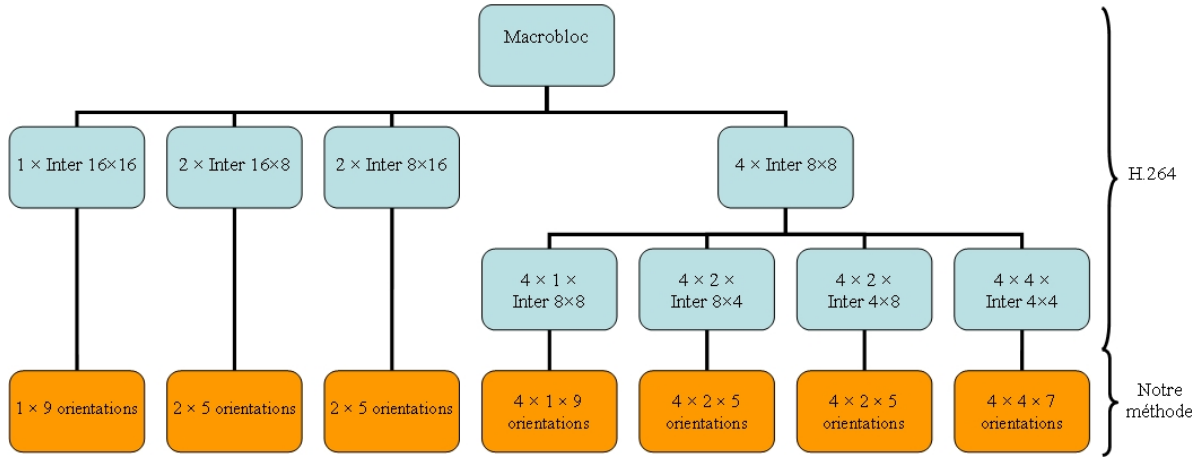


FIG. 5.27 – Déroulement du codage d'un macrobloc Inter H.264/AVC

Comme dans le cas Intra, ces combinaisons sont alors comparées afin d'en isoler la meilleure combinaison (taille de partitions, mouvement(s), orientation(s)) au sens débit-distorsion. Cette combinaison sera utilisée, par la suite, pour le codage réel du macrobloc.

Remarque importante : Le codeur H.264/AVC effectue un codage au fil de l'eau, c'est-à-dire qu'un macrobloc ne peut pas être codé si ses prédécesseurs (dans l'image courante pour le cas Intra et dans les images précédentes pour le cas Inter) n'ont pas déjà été codés et décodés.

- Cela implique que nos pré- et post-traitements ne peuvent pas s'appliquer à des résidus de taille supérieure au macrobloc 16×16 dans le codeur H.264/AVC.
- Un macrobloc codé le plus efficacement au sens débit-distorsion n'implique pas forcément qu'il sera meilleur prédicteur pour ses successeurs (dans l'image courante pour le cas Intra et dans les images suivantes pour le cas Inter). Une amélioration locale grâce à l'optimisation débit-distorsion peut ne pas améliorer globalement le codage, voire peut entraîner une dégradation globale. Cette optimisation débit-distorsion est donc un outil puissant, mais à manipuler avec précaution.

5.4.4 Codage de l'information d'orientation

Dans cette section, nous nous attacherons à coder les informations d'orientation nécessaires au décodeur pour réaliser la reconstruction.

La prédiction des informations d'orientation

Nous utilisons ici, dans le cas Inter, la même prédiction des états d'orientation que celle utilisée dans le cas Intra 16×16 présentée en section 5.3.3.

En effet, cette prédiction est utilisée en Intra 16×16 puisqu'il est difficile de trouver un voisinage codé avec la même taille de bloc. Il en est de même en Inter où il devient difficile de trouver un voisinage de la même taille que la partition courante quelle que soit sa taille.

Cette prédiction utilise l'état 0 comme orientation la plus probable pour la partition courante. L'état d'orientation à coder est alors toujours supérieur ou égal à cette orientation la plus probable (0). On transmet donc soit un drapeau en cas d'égalité soit cet état d'orientation moins 1.
Cette manipulation permet, comme dans le cas Intra, de limiter le nombre des états d'orientation suivant :

Partitions Inter	Nombres d'états d'orientation	Nombre de symboles à coder
16×16 et 8×8	9 (4 bits)	8 (3 bits)
16×8, 8×16, 8×4 et 4×8	5 (3 bits)	4 (2 bits)
4×4	7 (3 bits)	6 (3 bits)

TAB. 5.3 – Nombre de symboles à coder suivant chaque type de partition Inter

Le codage entropique des informations d'orientation prédites

Le même codeur arithmétique externe que pour le cas Intra (cf section 5.3.3) est utilisé ici pour le cas Inter.

Comme dans le cas Intra, l'initialisation de ce codeur est essentielle. Il est nécessaire ici de définir 8 initialisations, une pour chacune des partitions traitées (16×16, 16×8, 8×16, 8×8, 8×4, 4×8 et 4×4) et une supplémentaire pour le drapeau issu de la prédiction des informations d'orientation.

Pour cela, nous avons réalisé des tests de codage de ces informations d'orientation prédites avec ce codeur arithmétique, l'équiprobabilité des symboles et l'adaptation afin de recueillir ces probabilités affinées après une passe de codage sur un ensemble de séquences (Bus, Container, Flower, Foreman, Mobile et Tempete en CIF à 15 f.p.s.).
Ces tests nous ont alors permis par moyennage d'obtenir les probabilités d'initialisation pour chaque symbole de chacune des partitions selon :

(a) Blocs 16×16

Symbole	Probabilité
0	0.28
1	0.10
2	0.09
3	0.12
4	0.11
5	0.11
6	0.09
7	0.10

(b) Blocs 8×8

Symbole	Probabilité
0	0.25
1	0.11
2	0.10
3	0.10
4	0.10
5	0.11
6	0.11
7	0.12

(c) Blocs 4×4

Symbole	Probabilité
0	0.22
1	0.14
2	0.10
3	0.14
4	0.23
5	0.17

(d) Blocs 16×8

Symbole	Probabilité
0	0.25
1	0.24
2	0.27
3	0.24

(e) Blocs 8×16

Symbole	Probabilité
0	0.26
1	0.25
2	0.25
3	0.24

(f) Blocs 8×4

Symbole	Probabilité
0	0.41
1	0.12
2	0.23
3	0.24

(g) Blocs 4×8

Symbole	Probabilité
0	0.39
1	0.16
2	0.20
3	0.25

(h) Drapeau

Symbole	Probabilité
0	0.42
1	0.58

TAB. 5.4 – Probabilités moyennes des symboles suivant chaque partition Inter

Les tableaux 5.4(a)–(h) nous montrent qu'en utilisant la prédiction des états d'orientation présentée précédemment, les statistiques des symboles des partitions rectangulaires Inter (16×8, 8×16, 8×4 et 4×8) restent globalement proche de l'équiprobabilité. Il n'est donc pas nécessaire de définir des initialisations pour ces cas.

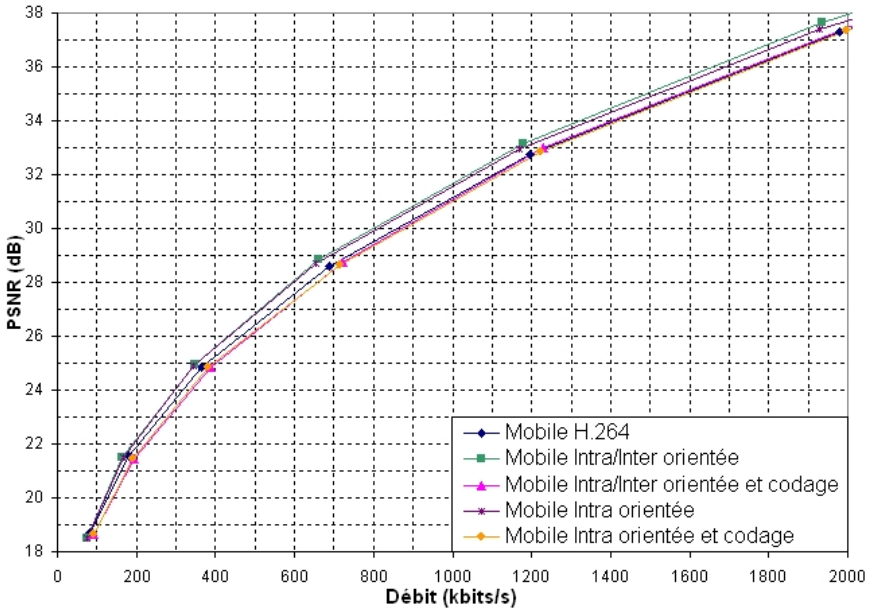
5.4.5 Résultats expérimentaux

Nous avons alors réalisé des expérimentations semblables à celles de la section 5.3 en utilisant la même configuration du codeur H.264/AVC, c'est-à-dire avec le JM10 en "High Profile" et à "Level 4.0" permettant ainsi l'utilisation de la DCT entière 8×8 et de CABAC.

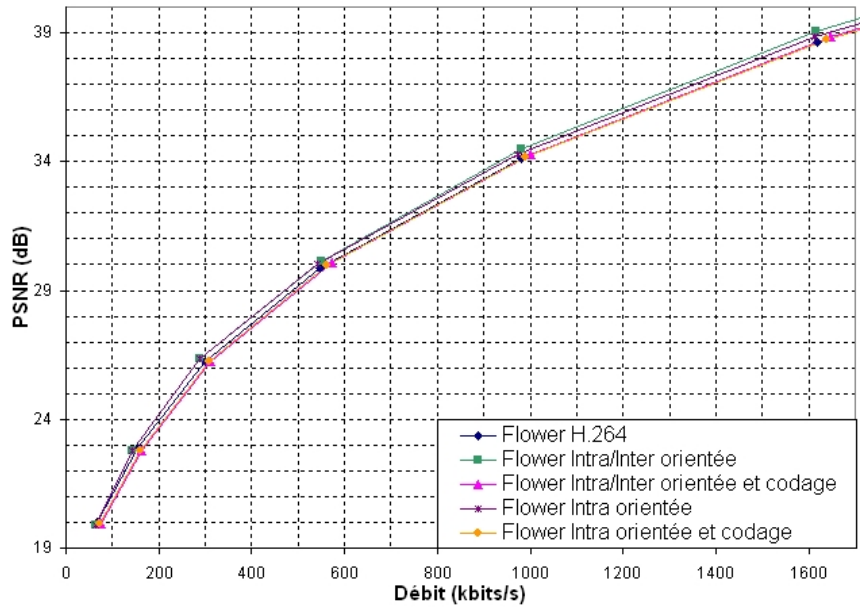
Ces tests s'appliquent maintenant aux images résiduelles de luminances Intra et Inter. Ils sont obtenus en faisant varier le pas de quantification des images résiduelles Intra (QPI) sur toute la plage disponible (de 1 à 51 (cf Annexes A.3.5 tab. A.5)) et en fixant le pas de quantification des images résiduelles Inter (QPP) à celui de l'Intra plus 1 (soit, QPP = QPI + 1). Cette spécification des pas de quantification provient des "Standard Conformance Tests" introduit par le JVT comme conditions standardisées de tests pour la norme H.264/AVC [JVT05] [ISO05].

Ces expérimentations ont été ici menées sur un ensemble de séquences CIF à 15 f.p.s..
Elles ont été réalisées en pré- et post-traitant les images résiduelles Intra et Inter, mais aussi en orientant uniquement les images résiduelles Intra. La comparaison de ces traitements permet alors de mesurer les gains apportés par l'orientation des blocs Inter.
De plus, comme dans le cas Intra, nous avons effectué ces tests avec et sans le codage des informations d'orientation afin d'identifier les pertes liées au codage de ces informations d'orientation.

Les résultats de deux de ces séquences : Mobile et Flower en CIF à 15 f.p.s., sont illustrés pour les bas débits sur la figure 5.28 et pour les hauts débits sur la figure 5.29.

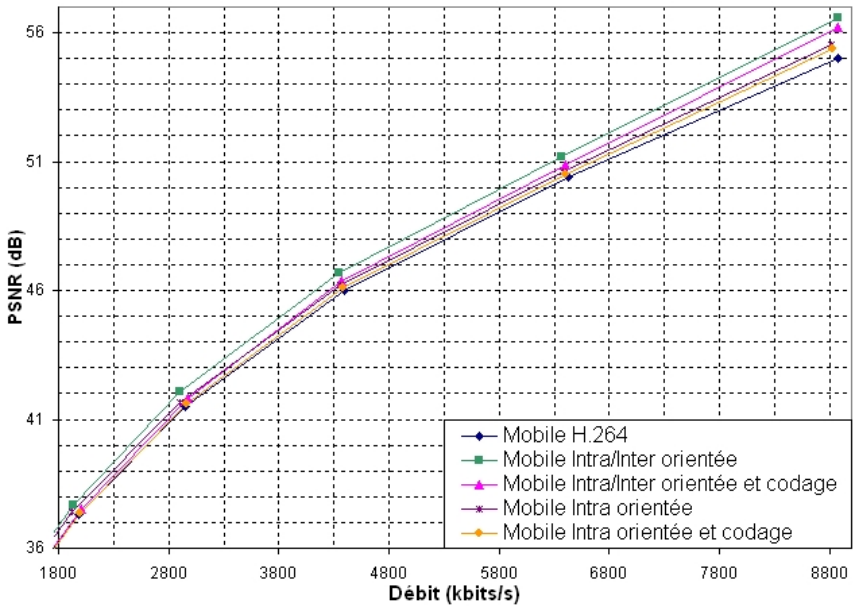


(a) Séquence Mobile

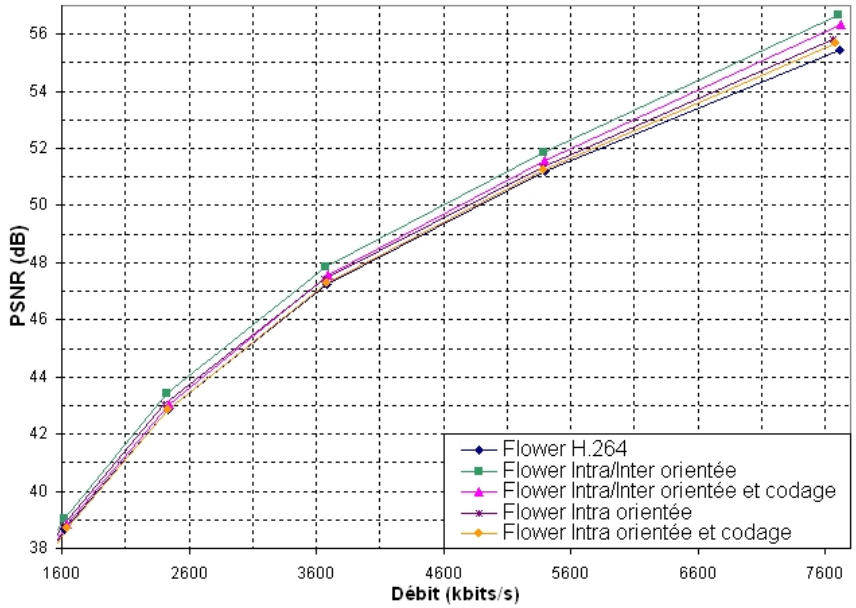


(b) Séquence Flower

FIG. 5.28 – Exemples de résultats Intra/Inter dans les bas débits (< 2 Mbits/s) avec et sans le codage des informations d'orientation



(a) Séquence Mobile

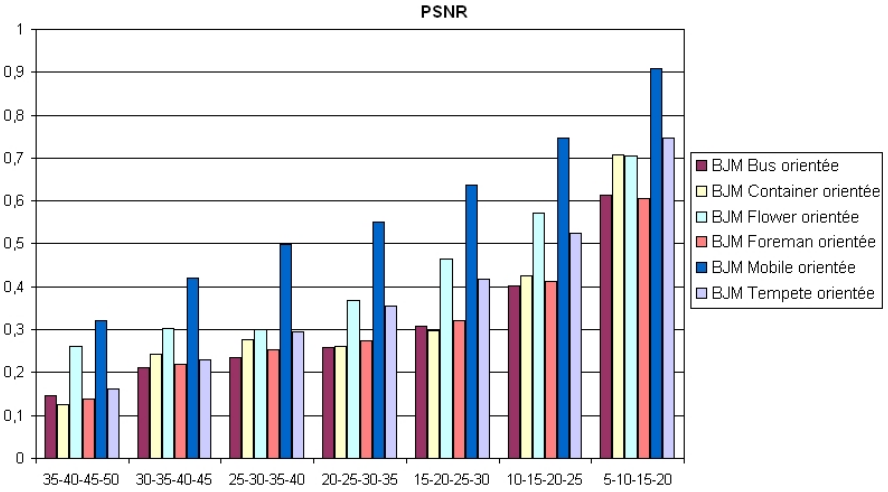


(b) Séquence Flower

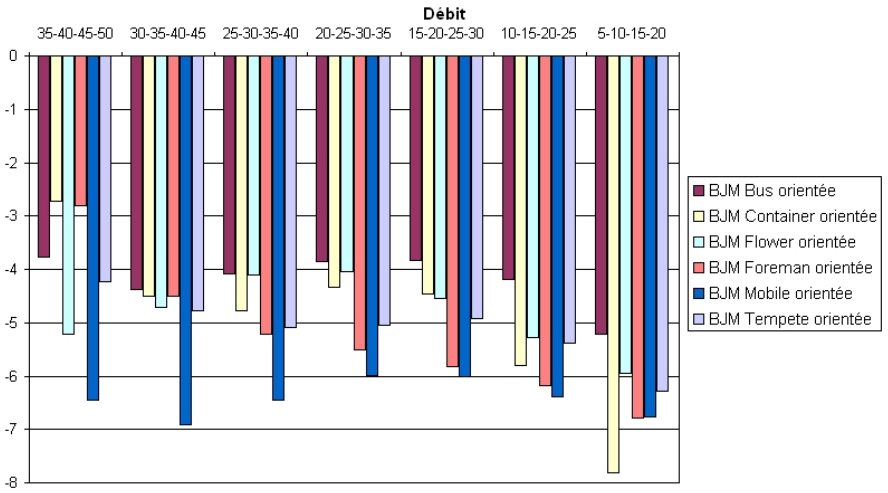
FIG. 5.29 – Exemples de résultats Intra/Inter dans les hauts débits ($>$ 1.6 Mbits/s) avec et sans le codage des informations d'orientation

Les quatre courbes des figures 5.28 et 5.29 montrent que les pré- et post-traitements des images résiduelles Inter permettent d'accroître les performances de notre méthode. En effet, elle semble plus efficace qu'en traitant uniquement les images résiduelles Intra, et cela aussi bien sans qu'avec le codage des informations d'orientation.
Cependant, ce codage des informations d'orientation continue à dégrader les performances de notre méthode et notamment dans les bas débits (cf fig. 5.28(a) et 5.28(b)). Par ailleurs, nos pré- et post-traitements semblent toujours améliorer le codage H.264/AVC dans les hauts débits où le coût des informations d'orientation devient alors moins important que le gain apporté par la méthode (cf fig. 5.29(a) et 5.29(b)).

Afin de mieux interpréter ces expérimentations, l'ensemble des résultats peut ici encore être résumé sous forme d'histogrammes de gain en PSNR et en pourcentage de débit en utilisant la métrique de Bjöntegaard, comme présenté sans le codage des informations d'orientation sur la figure 5.30 et avec ce codage sur la figure 5.31.

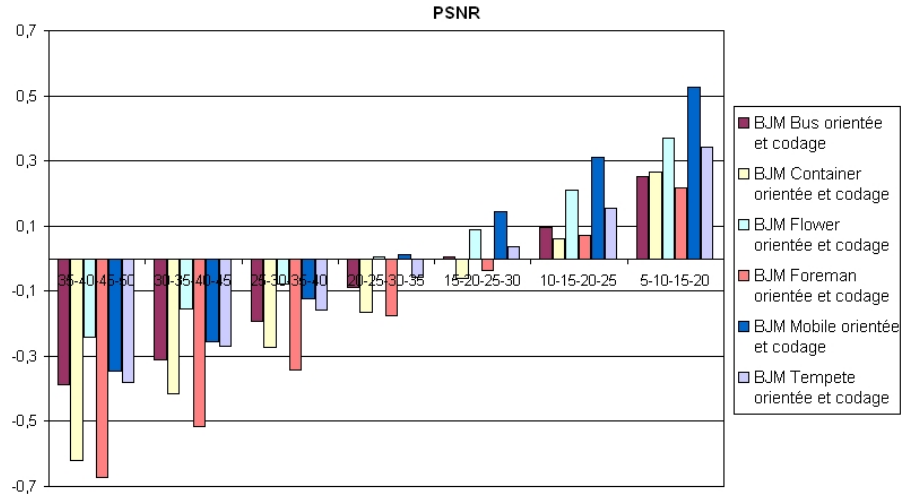


(a) Gain en PSNR

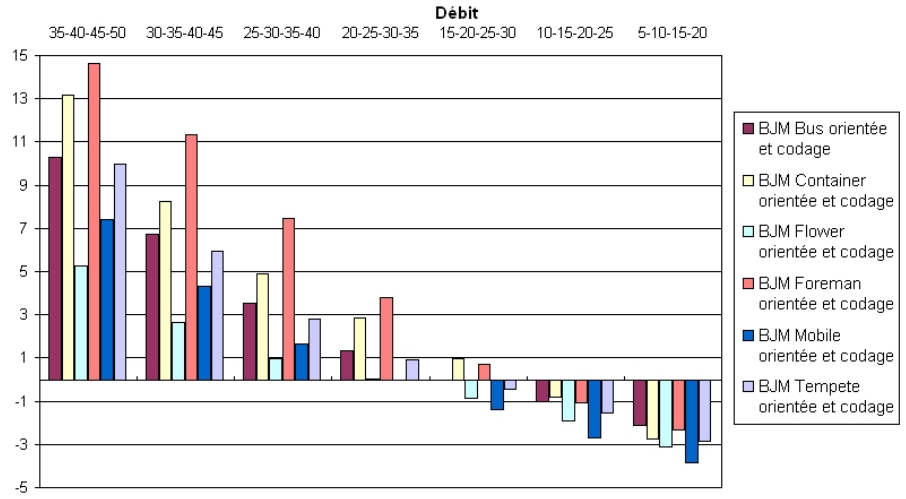


(b) Pourcentage de débit conservé

FIG. 5.30 – Résultats Intra/Inter sans le codage des informations d'orientation selon la métrique de Bjöntegaard



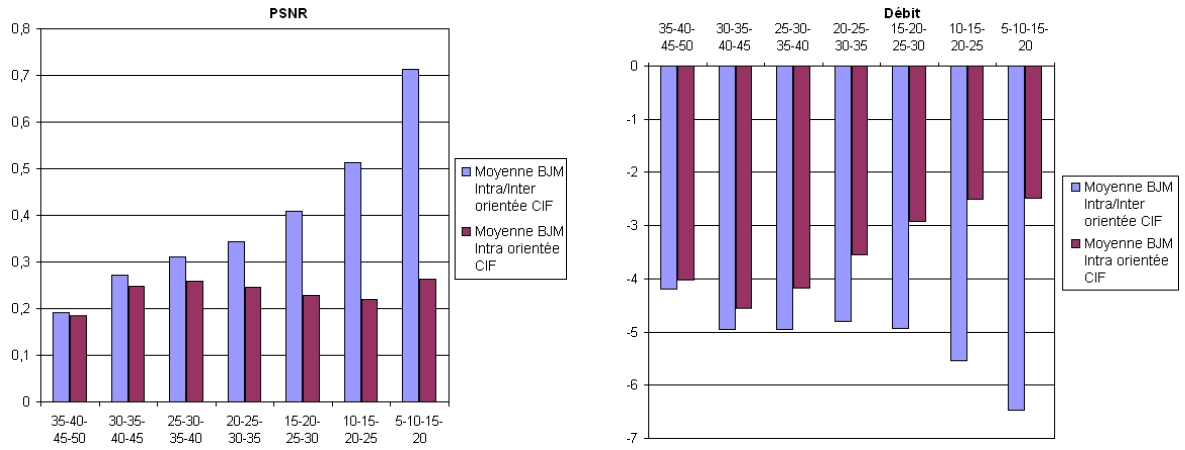
(a) Gain en PSNR



(b) Pourcentage de débit conservé

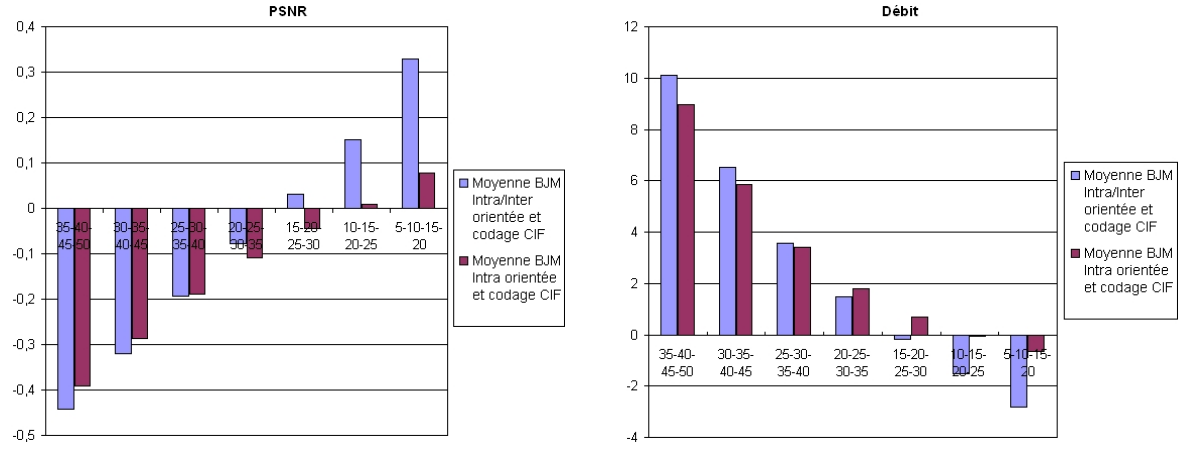
FIG. 5.31 – Résultats Intra/Inter avec le codage des informations d'orientation selon la métrique de Bjöntegaard

Des histogrammes moyens des gains en PSNR et débit peuvent alors être calculés pour ces séquences CIF selon cette même métrique, comme illustré sur la figure 5.32 sans le codage des informations d'orientation et sur la figure 5.33 avec ce codage.



(a) Gain en PSNR (b) Pourcentage de débit conservé

FIG. 5.32 – Moyenne des résultats Intra/Inter sans le codage des informations d'orientation selon la métrique de Bjöntegaard



(a) Gain en PSNR (b) Pourcentage de débit conservé

FIG. 5.33 – Moyenne des résultats Intra/Inter avec le codage des informations d'orientation selon la métrique de Bjöntegaard

Ces histogrammes 5.32 et 5.33 permettent de mettre en évidence le gain apporté par les pré- et post-traitements des images résiduelles Inter par rapport à H.264/AVC, mais aussi par rapport à notre méthode appliquée uniquement sur les images résiduelles Intra.

On voit clairement ici, et notamment sur les histogrammes de la figure 5.32, que l'orientation des images Inter permet d'améliorer les performances de notre méthode aussi bien en PSNR qu'en débit par rapport au traitement unique des images résiduelles Intra.
De même qu'en Intra, cette orientation des images résiduelles Inter apporte surtout du gain dans les hauts débits. Ces gains peuvent aller jusqu'à 0.45 dB en PSNR et 4.0% de débit conservé en

plus de l'Intra sans le codage de l'information (cf fig. 5.32), et jusqu'à 0.25 dB et 2.2% de débit conservé en plus de l'Intra avec le codage de ces informations d'orientation (cf fig. 5.33).
En moyenne, nos pré- et post-traitements des images résiduelles Inter génèrent 0.16 dB de gain et permettent de conserver 1.7% de débit par rapport à notre méthode appliquée uniquement sur les images résiduelles Intra sans le codage des informations d'orientation. Lorsque ces informations sont codés, ces gains moyens deviennent de 0.06 dB en PSNR et 0.4% de débit conservé en plus de l'Intra.

	Maximum		Moyenne	
	PSNR	débit	PSNR	débit
Sans codage	0.45 dB	-4.0%	0.16 dB	-1.7%
Avec codage	0.25 dB	-2.2%	0.06 dB	-0.4%

TAB. 5.5 – Améliorations apportées par l'orientation Inter par rapport à l'orientation uniquement Intra avec et sans le codage des informations d'orientation

Cependant, les histogrammes de la figure 5.33 montrent qu'il existe toujours un coût élevé de codage des informations d'orientation qui dégrade alors les performances globales de notre méthode.
Il la rend plus coûteuse que la norme H.264/AVC dans les bas débits, soit pour des QP supérieur à 30 ($QPI > 30$ et $QPP > 31$) pour les séquences CIF.
Néanmoins, dans les hauts débits, soit pour des QP inférieur à 30 ($QPI < 30$ et $QPP < 31$) pour les séquences CIF, notre méthode d'orientation des images résiduelles par pré- et post-traitements permet toujours d'améliorer le codage H.264/AVC. Cette amélioration peut aller en moyenne jusqu'à 0.33 dB en PSNR et permet de conserver en moyenne jusqu'à 2.8% de débit. Pour la séquence Mobile&Calendar en CIF à 15 f.p.s., l'amélioration atteint 0.53 dB pour 3.9% de débit conservé.

Ce coût de codage des informations d'orientation peut alors être estimé en effectuant la différence entre les résultats avec (cf fig. 5.33) et sans ces informations (cf fig. 5.32). On obtient ainsi les coûts Intra, Inter et Intra/Inter associé. En Intra, le codage des informations d'orientation fait perdre en moyenne 0.37 dB et ajoute 6.3% de débit à notre méthode. Cependant, ces pertes sont au minimum de 0.18 dB avec 1.8% de débit en plus, elles sont atteintes dans les hauts débits.
En Inter, les pertes liées au codage des informations d'orientation sont en moyenne de 0.10 dB pour 1.3% de débit en plus, pertes pouvant se réduire à 0.06 dB pour 0.9% de débit en plus dans les bas débits.
Globalement, le codage des informations d'orientation coûte 0.47 dB et accroît le débit de 7.6% en moyenne. Dans les hauts débits, ces pertes atteignent leurs minimums de 0.36 dB pour 3.6% de débit en plus.

Coût des informations d'orientation	Minimum		Moyenne	
	PSNR	débit	PSNR	débit
Intra	-0.18 dB	1.8%	-0.37 dB	6.3%
Inter	-0.06 dB	0.9%	-0.10 dB	1.3%
Total	-0.36 dB	3.6%	-0.47 dB	7.6%

TAB. 5.6 – Pertes moyennes liées au codage des informations d'orientation

5.5 Modification du parcours des coefficients de la transformée

5.5.1 Singularités des contours résiduels dans les blocs de la DCT H.264/AVC

Dans cette section, nous cherchons à exploiter une caractéristique de répartition spatiale des coefficients de la transformée DCT de la norme H.264/AVC afin d'améliorer le parcours des coefficients quantifiés.

Comme on a pu le voir jusqu'ici, cette transformée DCT s'applique en lignes et en colonnes, et intervient avant la quantification et le codage entropique. Ce codage entropique n'est cependant pas réalisé directement sur les coefficients quantifiés, mais sur des plages de valeurs parcourues en zigzag afin d'exploiter le nombre important de coefficients haute fréquence nuls (cf section 1.2.4 fig. 1.10).

Néanmoins, dans certains cas, comme celui des blocs contenant des contours orientés, ce parcours zigzag n'est pas adapté, comme illustré sur l'exemple de la figure 5.34.

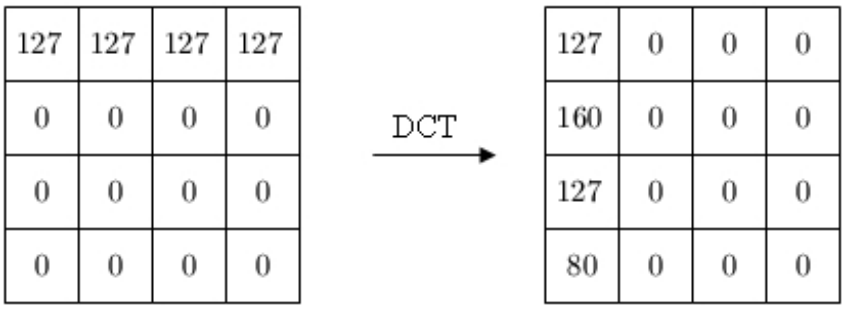


FIG. 5.34 – Exemple de bloc présentant un contour horizontal, avant et après transformation DCT

Dans ce cas, la transformée DCT H.264/AVC produit :
- un bloc de coefficients verticaux pour un contour horizontal dans le bloc.
- un bloc de coefficients horizontaux pour un contour vertical dans le bloc.

Le parcours zigzag n'est alors pas adapté pour ces blocs orientés. En effet, il produit, dans le cas de l'exemple de la figure précédente 5.34, la chaîne suivante :

$$\{127, 0, 160, 127, 0, 0, 0, 0, 0, 80, 0, 0, 0, 0, 0, 0\}$$

alors que le parcours optimal de ces coefficients produirait :

$$\{127, 160, 127, 80, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\}$$

Ce parcours optimal est, dans ce cas, un parcours vertical des coefficients quantifiés.

Il est donc préférable, dans le cas des blocs orientés, d'adopter un parcours des coefficients quantifiés différent, notamment :
- une analyse verticale des coefficients quantifiés pour un contour horizontal dans le bloc.
- une analyse horizontale des coefficients quantifiés pour un contour vertical dans le bloc.

5.5.2 Adaptation du parcours des coefficients après la DCT H.264/AVC

Dans le chapitre précédent, nous avons défini une méthode par pré- et post-traitements des blocs résiduels Intra et Inter H.264/AVC qui exploite l'orientation de ces blocs avant transformée DCT afin d'en améliorer les performances.

Cette méthode permet d'augmenter la décorrélation effectuée par la transformée DCT en redressant ces blocs résiduels soit vers l'horizontale soit vers la verticale.
Il est donc intéressant d'adopter, dans ce contexte, l'adaptation du parcours des coefficients de la DCT comme présenté précédemment.

Suivant le type de redressement

Les blocs carrés, 16×16, 8×8 et 4×4 qu'ils soient Intra ou Inter, et dont l'orientation est comprise entre -45^o et $+45^o$ inclus sont redressés selon la méthode du chapitre précédent 5 vers l'horizontale.
Et si leurs orientations sont comprises entre $+45^o$ et $+90^o$ ou entre -45^o et -90^o, alors ces blocs carrés sont redressés vers la verticale suivant le pré-traitement du chapitre 5.

D'après la caractéristique de la DCT présentée en section 5.5.1, il convient, après transformation DCT de ces blocs résiduels Intra ou Inter redressés, de parcourir les coefficients quantifiés :
- selon la verticale pour un redressement horizontal.
- selon l'horizontale pour un redressement vertical.
- en zigzag si aucun redressement n'est appliqué.

En effet, si aucune orientation n'est satisfaisante pour ces blocs ou s'il ne sont pas orientés (blocs horizontaux ou verticaux), on utilise alors l'état d'orientation 0 qui correspond au codage H.264/AVC classique. Il convient d'utiliser, dans ce cas, le parcours classique des coefficients quantifiés à savoir le zigzag.

Ces différents parcours des coefficients quantifiés en fonction des redressements appliqués pour chaque type de bloc qu'il soit Intra ou Inter peuvent être résumés suivant :

Taille de bloc	Orientation du bloc	Redressement effectué	Parcours des coefficients
4×4	$\pm 27^o$ et $\pm 45^o$	horizontal	vertical
	$\pm 63^o$	vertical	horizontal
	0	/	zigzag
8×8	$\pm 14^o$ et $\pm 45^o$	horizontal	vertical
	$\pm 63^o$ et $\pm 76^o$	vertical	horizontal
	0	/	zigzag
16×16	$\pm 7^o$ et $\pm 14^o$	horizontal	vertical
	$\pm 76^o$ et $\pm 83^o$	vertical	horizontal
	0	/	zigzag

TAB. 5.7 – Parcours de coefficients appliqués suivant le type de bloc et de redressement

Suivant le partitionnement Inter

La section précédente 5.5.2 présentait les parcours utilisés pour les blocs carrés, cependant, comme on a pu le voir en section 5.4, il existe dans le mode Inter des blocs rectangulaires : 16×8, 8×16, 8×4 et 4×8.

D'après la section 5.4.2, les partitions 16×8 et 8×4 horizontales sont redressées uniquement vers l'horizontale, et les partitions 8×16 et 4×8 verticales seulement vers la verticale.
Il convient donc, en s'appuyant sur la caractéristique de la DCT H.264/AVC présentée en 5.5.1, de parcourir les coefficients issus de ces partitions :

- selon la verticale pour les partitions horizontales 16×8 et 8×4 redressées.
- selon l'horizontale pour les partitions verticales 8×16 et 4×8 redressées.
- en zigzag si aucun redressement n'est appliqué.

Ces différents parcours des coefficients quantifiés en fonction du type de partitions Inter peuvent être résumés suivant :

Partitions Inter	Parcours des coefficients
16×8 et 8×4	vertical
8×16 et 4×8	horizontal
sans redressement	zigzag

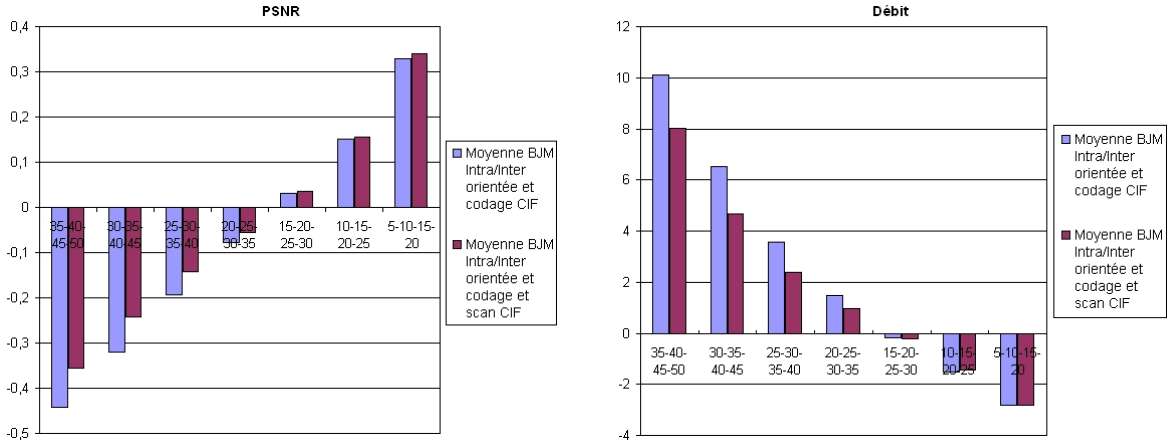
TAB. 5.8 – Parcours de coefficients appliqués suivant le type de partition

Résultats expérimentaux

Cette adaptation du parcours des coefficients quantifiés fonction du redressement appliqué ou de la partition traitée a été implémentée et associée à notre méthode d'orientation (cf chapitre 5) dans le codeur H.264/AVC (JM10.0).

Des tests ont alors été réalisés afin de comparer notre méthode d'orientation par pré- et post-traitements avec et sans cette adaptation du parcours des coefficients quantifiés vis à vis de la norme H.264/AVC. Ils sont obtenus en faisant varier le pas de quantification des images résiduelles Intra (QPI) et en fixant celui des images résiduelles Inter (QPP) à celui de l'Intra plus 1 (soit, QPP = QPI + 1) ("Standard Conformance Tests").

Ces expérimentations ont été menées sur un ensemble de séquences CIF à 15 f.p.s.. Les résultats sont présentés sur la figure 5.35 sous forme d'histogrammes moyens de gain en PSNR et en pourcentage de débit obtenus en utilisant la métrique de Bjöntegaard sur cet ensemble de séquences.



(a) Gain en PSNR

(b) Pourcentage de débit conservé

FIG. 5.35 – Moyenne des résultats Intra/Inter avec et sans l'adaptation du parcours des coefficients quantifiés

Ces histogrammes nous montrent que l'adaptation du parcours des coefficients quantifiés est efficace et notamment dans les bas débits, en améliorant notre méthode d'orientation.

En effet, les moyennes de ces histogrammes révèlent un gain de 0.04 dB en PSNR et une conservation de 0.8 % du débit par rapport à notre méthode sans cette adaptation. Ces moyennes confirment bien que cette adaptation du parcours des coefficients quantifiés permet d'améliorer notre méthode d'orientation par pré- et post-traitements.

Cependant, on peut remarquer que l'efficacité de cette adaptation de parcours des coefficients quantifiés est décroissante quand le débit augmente. Cela s'explique du fait que :

- Ce parcours des coefficients quantifiés permet de définir les événements $(last, run, level)$ à coder entropiquement (cf section 1.2.4) où le run correspond à la distance entre deux coefficients non nuls. Avec le parcours optimal des coefficients quantifiés, tous les run deviennent nuls, mais le même nombre d'événements reste à coder. Ce parcours optimal permet alors uniquement de conserver le débit de ces run annulés. Ce débit reste faible, il est donc à fortiori plus significatif dans les bas débits que dans les hauts débits.
- Notre sélection des orientations nécessaires pour le choix de parcours des coefficients quantifiés est basée sur un critère débit-distorsion. A bas débits, connaissant le coût de codage élevé de nos informations d'orientation, cette sélection est efficace, elle choisit d'utiliser les orientations seulement lorsqu'elles permettent effectivement de redresser les blocs vers l'horizontale ou la verticale, c'est-à-dire là où notre méthode est la plus performante. Par contre, dans les hauts débits, l'impact de nos informations d'orientation sur le débit total se réduit, la sélection utilisée devient alors plus souple et permet d'utiliser les orientations même si celles-ci ne redressent pas complètement les blocs vers l'horizontale ou la verticale. Pour ces blocs, notre adaptation du parcours des coefficients devient alors inefficace.

Remarque : en mode Inter, si après transformation et quantification il ne reste qu'un seul coefficient dans le bloc ou deux coefficients suffisamment éloignés dans le sens du parcours zigzag, alors ce bloc n'est pas codé, on réutilise sa version prédite et on ne transmet qu'un "skip". Lorsque l'on modifie le parcours des coefficients quantifiés, on peut être amené à coder quelques uns de ces blocs et, de fait, augmenter de beaucoup le débit et de peu la qualité.

5.6 Conclusion

Ce chapitre introduit une méthode d'orientation par pré- et post-traitements associée à un parcours adapté des coefficients quantifiés qui permet d'améliorer le codage H.264/AVC notamment dans les hauts débits.

La méthode d'orientation par pré- et post-traitements, présentée en sections 5.2, 5.3 et 5.4, effectue des pseudo-rotations des blocs prédits les redressant vers l'horizontale ou la verticale avant que leur soient appliqués la transformée DCT entière de H.264/AVC (4×4 ou 8×8).
Le pré-traitement réalisant ces pseudo-rotations consiste à transformer les blocs par des cisaillements assimilables à des permutations circulaires des pixels de ces blocs. Le post-traitement consiste alors à effectuer les permutations inverses après la DCT inverse lors de la reconstruction de ces blocs.
Nous avons alors défini différentes orientations pour chaque taille de blocs utilisés en Intra : 16×16, 8×8 et 4×4, et pour chaque taille de partitions Inter : 16×16, 16×8, 8×16, 8×8, 8×4, 4×8 et 4×4. Ces orientations sont ensuite sélectionnées pour chaque macrobloc en utilisant le critère débit-distorsion du codeur H.264/AVC.
Néanmoins, il faut transmettre au décodeur des informations d'orientation afin de lui indiquer comment décoder les blocs et partitions qu'il reçoit. Ces informations sont codées à l'aide d'un codeur arithmétique correctement initialisé et externe à H.264/AVC. Elles sont ensuite insérées dans l'en-tête du bloc ou de la partition correspondante.

Les résultats montrent que cette méthode d'orientation par pré- et post-traitements permet d'améliorer les performances du codeur H.264/AVC notamment dans les hauts débits.
En effet, les performances maximales sont atteintes sans le codage des informations d'orientation, et permettent de définir les limites maximales de la méthode. Elles montrent alors que cette méthode améliore le codage des images résiduelles H.264/AVC aussi bien Intra que Inter quel que soit le débit (cf Tab. 5.9).
Le codage des informations d'orientation a un coût élevé (cf Tab. 5.9) qui se ressent surtout dans les bas débits ($QP > 30$) rendant cette méthode d'orientation moins performante que la norme H.264/AVC. Cependant, dans de plus hauts débits ($QP < 30$), ces informations d'orientation deviennent moins importantes que le gain apporté par la méthode qui reste alors plus performante que la norme H.264/AVC (cf Tab. 5.9).

Performances	Maximales (sans le codage des infos)	Réelles (avec le codage des infos)	
		Moyennes	Maximales
Intra	0.45 dB -5.1% de débit	-0.15 dB +3.3% de débit	0.27 dB -1.6% de débit
Inter	0.16 dB -1.7% de débit	0.06 dB -0.4% de débit	0.25 dB -2.2% de débit
Totales	0.39 dB -5.1% de débit	-0.07 dB +2.4% de débit	0.33 dB -2.8% de débit

Tab. 5.9 – Performances moyennes de la méthode d'orientation des blocs et partitions selon les modes de codage H.264/AVC

Cependant, il serait possible d'améliorer le codage de ces informations d'orientation. Par exemple, on pourrait chercher à améliorer la prédiction de ces informations avec le mouvement et les images de référence Inter ou à utiliser un codeur adaptatif au contexte tel que CABAC nécessitant la définition de nombreux modèles statistiques et autres éléments de syntaxe. Mais on pourrait aussi chercher à se générer du débit pour ces informations, par exemple, en applicant cette méthode dans le domaine spatial et en supprimant la prédiction Intra (qui peut être redondante avec la méthode) ou en modifiant le critère débit-distortion.

La section 5.5 montre qu'une caractéristique de la DCT H.264/AVC nous a permis de définir des parcours adaptés des coefficients quantifiés pour notre méthode d'orientation afin d'en améliorer les performances.

Cette caractéristique de la DCT que nous avons exploitée ici est que la transformation d'un bloc contenant un contour vertical, respectivement horizontal, nous donne un bloc de coefficients horizontaux, respectivement verticaux.
La méthode d'orientation par pré- et post-traitements du chapitre précédent redressent les blocs résiduels avant transformée vers l'horizontale ou la verticale. On applique donc un parcours horizontal, respectivement vertical, des coefficients quantifiés lorsqu'ils sont issus d'un redressement vertical, respectivement horizontal et le zigzag quand aucun redressement n'est appliqué. Pour les blocs carrés, 16×16, 8×8 et 4×4, Intra et Inter, on choisit le parcours en fonction du redressement utilisé. Pour les partitions rectangulaires Inter, 16×8, 8×16, 8×4 et 4×4, le parcours est défini par l'orientation de ces partitions (rectangles horizontaux ou verticaux).

Les résultats de cette adaptation de parcours des coefficients quantifiés montrent qu'elle permet d'améliorer notre méthode d'orientation par pré- et post-traitements de presque 1% de débit (cf Tab. 5.10). De plus, cette amélioration qui touche surtout les bas débits, c'est-à-dire là où notre méthode est la moins performante, permet d'améliorer jusqu'à plus de 2% le débit (cf Tab. 5.10).

Performances réelles	Moyennes	Maximales
Totales	0.04 dB -0.8% de débit	0.08 dB -2.1% de débit

Tab. 5.10 – Performances moyennes de l'adaptation des parcours des coefficients quantifiés appliquée à la méthode d'orientation des blocs de la transformée

Les performances globales de notre méthode d'orientation par pré- et post-traitements accompagnée d'un parcours adapté des coefficients quantifiés, comparé à ceux du tableau 5.9 sans cette adaptation, deviennent alors :

Performances réelles	Moyennes	Maximales
Totales	-0.03 dB +1.6% de débit	0.34 dB -2.8% de débit

Tab. 5.11 – Performances moyennes de la méthode d'orientation suivie de l'adaptation du parcours des coefficients quantifiés par rapport à H.264/AVC

Quatrième partie

Conclusion et perspectives

Conclusion générale

Le codage vidéo et notamment le codage vidéo normalisé, s'appuie sur un schéma de codage hybride utilisant quatres principaux modules, présentés au chapitre 1 : la prédiction, la transformation, la quantification et le codage entropique.

L'étape de transformation s'applique au niveau bloc (8×8 dans la plupart des schémas et 8×8 ou 4×4 pour le codeur H.264/AVC) et permet mathématiquement de décorréler les informations de ces blocs. Elle représente donc une étape importante de la compression vidéo dont l'efficacité dépend du niveau de décorrélation réalisé.

Cette transformation peut être une DCT comme dans tous les schémas de codage vidéo normalisé, DCT réalisée en calculs flottants ou entiers pour la norme H.264/AVC. Ce peut aussi être, d'après le chapitre 2, des ondelettes de première génération continues ou discrètes, une DCT en étages lifting telles que la BinDCT et l'IntDCT, ou des transformées à recouvrement comme la LOT et la LBT.
Cependant, ces transformées souffrent toutes du même problème à savoir qu'elles ne permettent pas de bien représenter les contours des objets. D'autres transformées ont alors été proposées afin de palier ce problème.

Le chapitre 3 présente ces transformées qui utilisent avantageusement les structures géométriques contenues dans les images afin de représenter plus efficacement les contours des objets. Il en existe deux grandes catégories : une basée ondelettes et l'autre basée DCT.
Les ondelettes de seconde génération permettent cette représentation des contours. Parmi elles (cf section 3.2), les *ridgelets*, les *curvelets* et les bandelettes de première génération qui sont définies dans le cas continu ne s'adaptent pas bien ici au cas des images et des vidéos numériques (discrètes). Les *curvelets* et les *contourlets* sont, quant à elles, redondantes, donc moins facilement utilisables dans des schémas de compression. Par ailleurs, les bandelettes nécessitent une détection de contours et un ré-échantillonnage des données par interpolation. Enfin, les *directionlets* nécessitent une étape de rotation (par interpolation) non réversible.
Les DCT exploitant l'orientation (cf section 3.3) répondent aussi à ce problème de représentation des contours. Cependant, la BinDCT orientée nécessite, comme les bandelettes, une étape de ré-échantillonnage par interpolation. La SA-DCT implique de transmettre une carte de contours en plus des coefficients, et elle est basée, comme la DCT directionnelle, sur des DCT flottantes de taille variable dont les problèmes liés aux arrondis sont bien connus. Enfin, la DCT directionnelle modifie la forme des blocs traités, la quantification et le parcours des coefficients qui doivent alors être adaptés.

Nous avons cherché à exploiter les structures géométriques des images en s'affranchissant de ces problèmes. Pour cela, nous avons développé plusieurs méthodes, basées sur : les ondelettes ou les transformées à recouvrement dans le chapitre 4, et une méthode d'orientation par pré- et post-traitements au chapitre 5.

Dans un premier temps, ayant remarqué que les plus anciennes études sur la représentation des contours sont portées par les ondelettes, nous avons défini la DCT entière H.264/AVC sous forme d'un schéma lifting présenté en section 4.1. Cette mise en forme lifting permet de voir cette DCT entière comme étant proche d'une transformation en ondelettes. On peut alors lui

appliquer les propriétés des ondelettes de seconde génération pour améliorer sa représentation des contours.
Cependant, les expérimentations menées n'ont pas permis de montrer l'équivalence parfaite entre le schéma lifting proposé et la DCT entière H.264/AVC. En effet, une dérive apparaît dans les hauts débits à cause d'arrondis effectués à la suite d'une division par 5 inhérente au schéma lifting proposé. A bas débits, ces effets d'arrondis sont absorbés par la quantification, on obtient alors l'équivalence entre le schéma lifting et la DCT entière H.264/AVC.

La seconde approche, basée transformée à recouvrement, consiste à intégrer la LBT en pré- et post-traitements dans le codeur H.264/AVC, comme présenté dans la section 4.2. Mais, de par sa construction, le codeur H.264/AVC nous oblige à n'utiliser qu'une version causale de cette transformée LBT, diminuant, par ailleurs, sa possible efficacité.
Les excellentes performances de cette méthode sur les images naturelles ne se retrouvent finalement pas sur les images résiduelles Intra ou Inter H.264/AVC. En effet, les blocs d'images qui sont transformés dans ce codeur sont tous prédits soit spatialement soit temporellement. Il y a donc moins de corrélations à exploiter par une transformée à recouvrement entre des blocs résiduels qu'il peut en exister entre des blocs naturels.

Les images prédites Intra et Inter du codeur H.264/AVC présentent toujours des motifs réguliers et orientés. Nous avons donc cherché à les exploiter à l'aide d'une méthode d'orientation par pré- et post-traitements suivie d'un parcours adapté des coefficients quantifiés, présentée au chapitre 5.
Le pré-traitement permet de redresser les blocs vers l'horizontale ou la verticale par cisaillements, soit par permutations circulaires des pixels, améliorant la décorrélation de la DCT qui suit. Les coefficients quantifiés sont ensuite parcourus à la verticale ou à l'horizontale respectivement avec les redressements appliqués. Les orientations des blocs sont sélectionnées à l'aide d'un critère débit-distorsion et sont codées arithmétiquement avant d'être insérées dans le flux vidéo.
Les résultats montrent qu'en moyenne cette méthode d'orientation couplée à un parcours adapté des coefficients quantifiés ne permet pas d'améliorer le codage H.264/AVC. En effet, ces résultats montrent une perte moyenne de 0.03 dB et une augmentation de 1.6% du débit.
Cependant, dans les hauts débits ($QP < 30$ et même $QP < 35$ pour certaines séquences), cette méthode complète permet d'améliorer le codage H.264/AVC. Les performances de la méthode d'orientation complète peuvent alors atteindre un gain de 0.34 dB accompagné de 2.8% de gain en débit.

Perspectives de recherche

Suite à ces travaux, de nombreuses perspectives s'offrent à nous. Elles peuvent aussi bien être directement liées à nos travaux qu'être d'ordre plus général.

La section 4.1 propose un schéma sous forme lifting simulant la DCT H.264/AVC au moins dans les bas débits. Or, c'est pour ces débits que la méthode d'orientation par pré- et post-traitements est la moins performante même couplée à l'adaptation du parcours des coefficients quantifiés.

La DCT sous forme lifting à laquelle seront ajoutés des outils de seconde génération pourrait alors servir d'alternative à la méthode d'orientation dans les bas débits.

Comme on a pu le voir, le problème majeur de la méthode d'orientation par pré- et post-traitements est le coût des informations d'orientation associées aux traitements.
Une première idée est d'améliorer la prédiction des états d'orientation sélectionnés afin d'affiner les statistiques des symboles à coder. Pour cela, en Intra, on pourrait mettre en place des équivalences entre les états d'orientation des différentes tailles pour pouvoir utiliser le voisinage dans tous les cas (même s'il est de taille différente). Et, en Inter, on pourrait utiliser comme prédicteur la (ou les) orientation(s) du (ou des) bloc(s) de référence récupérée(s) à l'aide des informations de mouvement.
Ces informations d'orientation prédites doivent ensuite être codées entropiquement. Cette opération est ici réalisée avec un codeur arithmétique externe au codeur H.264/AVC. Les performances de celui-ci serait accrues par l'utilisation de contextes. Pour cela, nous pourrions utiliser CABAC, mais il faudrait, par ailleurs, lui adjoindre de nouveaux contextes (modèles statistiques) pour ces informations d'orientation. Il faudrait alors autant de contextes qu'il nous a fallu d'initialisations pour notre codeur arithmétique, à savoir 12 (4 en Intra et 8 en Inter).

Une autre solution serait de ne pas à avoir à transmettre ces informations. En utilisant un critère qui définirait une orientation pour le bloc courant à l'aide des informations de voisinage ou de référence, le décodeur n'aurait pas besoin d'informations supplémentaires pour effectuer son décodage. Cependant, un tel critère est difficilement définissable pour des images résiduelles où il n'existe presque plus de corrélation entre les blocs, comme on a pu le voir en section 4.2.

Plus généralement, on pourrait penser pré- et post-traiter non plus les blocs et partitions, mais directement des zones plus grandes dans les images. Ceci nécessiterait alors de connaître, au préalable, les images résiduelles complètes afin d'y appliquer une étape de segmentation avec la transmission d'une carte correspondante. Cela diminuerait fortement l'impact des informations d'orientation.

Les images Intra H.264/AVC subissent une prédiction spatiale directionnelle qui peut être redondante avec notre méthode d'orientation par pré- et post-traitements. Il serait envisageable de supprimer cette prédiction et d'appliquer notre méthode dans le domaine spatial (plutôt que résiduel). Le mode de prédiction Intra de chacun des blocs ne serait alors plus codé, mais remplacé par l'information d'orientation.
De plus, il serait possible de n'utiliser que notre méthode d'orientation, sans prédiction Intra ni transformation DCT. En effet, la section 5.5 montre que la transformation d'un bloc de contour ne diminue pas le nombre d'informations à coder entropiquement par la suite, elle n'est donc pas nécessaire. Et, l'APEC ("Adaptive Prediction Error Coding") [NM06] intégré au codeur KTA [kta07] (nouveau codeur développé par le groupe VCEG de l'ITU (cf Annexes A.1)) montre que la suppression de la transformation DCT dans certain cas permet d'améliorer la compression de ce codeur H.264/AVC.

La sélection des orientations utilisée est, comme on l'a vu en section 5.4.3, basée sur un critère débit-distorsion. Nous avons cependant remarqué que cette sélection est locale générant ainsi des améliorations locales, mais pouvant devenir des dégradations globales. En effet, cette sélection réalise une optimisation débit-distorsion du macrobloc courant qui sert ensuite de prédicteur.
On pourrait imaginer tourner cette optimisation plutôt vers la prédiction en cherchant le meilleur

codage du macrobloc au sens cette fois-ci débit-prédiction où le macrobloc codé/décodé serait le meilleur prédicteur pour ses successeurs. En effet, si un macrobloc est un bon prédicteur, alors on peut penser qu'il a une faible distorsion. Par contre, ce critère débit-prédiction ne serait alors plus local, mais deviendrait plus général et sans doute aussi plus complexe.

Enfin, il est évident, après ces travaux, que le codage vidéo sera amené à prendre en compte des techniques exploitant les orientations comme celle présentée ici.
En effet, on a vu qu'il existe, à l'heure actuelle, un très grands nombres de transformées qui utilisent déjà les orientations avantageusement telles que les ondelettes de seconde génération, les DCT orientées et la méthode développée dans ce manuscrit. Ces transformées sont toutes plus performantes que celles utilisées aujourd'hui dans les codeurs normalisés qui, s'ils veulent pouvoir évoluer, seront appelés à en faire usage.

De plus, ces techniques orientées qui sont déjà efficaces pour le codage vidéo peuvent s'avérer très utiles dans d'autres domaines que la compression. Elles permettent de mieux représenter les contours que les techniques classiques, et ainsi peuvent être utilisées pour de l'analyse ou de la description de contenu, ou pour la 3D.

Cinquième partie

Annexes

Annexe A

Le codage vidéo normalisé

A.1 Historique des normes vidéo

Les normes de codage vidéo sont issues de deux grands organismes de normalisation : l'ITU-T et l'ISO-IEC (cf fig. A.1).

ITU-T (International Telecommunications Union - Telecommunications Standards Sector)
Le SG16 (Study Group 16) de cet organisme a contribué à la normalisation de :

- H.261 [ITU90] "Video codec for audio-visual services at px64kbit/s" pour la visioconférence bas débit.
- H.262 [ITU94] équivalent à MPEG-2 pour la diffusion de télévision numérique.
- H.263 [ITU95], H.263+ [ITU98], H.263++ [ITU01] "Video coding for low bitrate communication" pour la visiophonie et la visioconférence "améliorée".
- H.264 [JVT05] "Advanced video coding for generic audio-visual services" en cours d'amélioration avec l'ISO (FRExt [JVT04], SVC [JVT06], MVC,...).

ISO-IEC (International Organisation for Standardization - International Electrotechnical Commission)
Cet organisme est constitué de plusieurs comités dont le JTC1 (Joint Technical Committee 1) qui est composé de plusieurs groupes d'étude (WG Work Group) dont le WG1 ou JPEG pour le codage d'images fixes et le WG11 ou MPEG (Moving Picture Experts Group) pour le codage d'images mobiles et d'audio. Le groupe de travail MPEG a contribué à la normalisation de :

- MPEG-1 [ISO93] "Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s" pour le stockage de vidéos sur CD-ROM.
- MPEG-2 [ISO94b] "Generic coding of moving pictures and associated audio information" pour la diffusion de télévision numérique.
- MPEG-4 [ISO00a] "Generic coding of audio-visual objects", norme générique pour le stockage et la diffusion de vidéos.
- MPEG-7 [ISO01a] "Multimedia content description interface" permet la description des contenus multimédia.
- MPEG-21 [ISO01b] "Multimedia framework" décrit comment s'assemblent les éléments de la chaîne qui va depuis la production jusqu'à la consommation de contenus multimédia.
- MPEG-4/AVC [JVT05] "Advanced video coding" en cours d'amélioration avec l'ITU (FRExt [JVT04], SVC [JVT06], MVC,...).

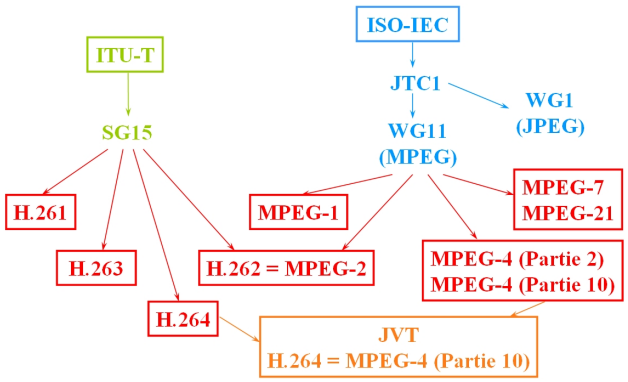


FIG. A.1 – Les organismes de normalisation et les normes

Le groupe d'études 16 de l'ITU-T fait des efforts de normalisation en codage vidéo depuis 1984, alors que le groupe MPEG existe depuis 1988. Les différentes normes présentées précédemment ont donc été créées depuis 1984 comme suit :

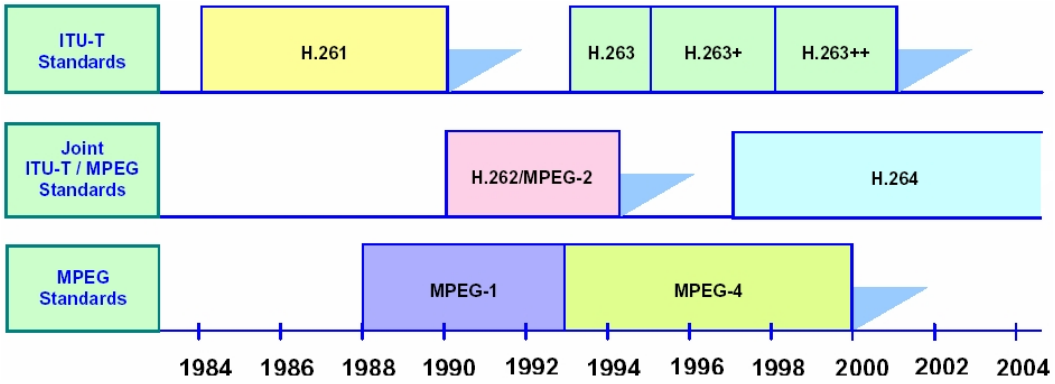


FIG. A.2 – Historique des normes ISO et ITU

Ces deux groupes ont formé fin 2001 le JVT (Joint Video Team) pour créer cette norme H.264 MPEG-4/AVC par effort de coopération plutôt que par concurrence. On voit ici que des travaux sont toujours en cours pour améliorer la version de base de la norme H.264 MPEG-4/AVC standardisée en mai 2003 en lui ajoutant des amendements tels que "FREXt" [JVT04] qui permet notamment d'utiliser une DCT entière 8×8 supplémentaire et de traiter les images hautes définitions, "SVC" [JVT06] [SMW06] [SMW07] ("Scalable Video Coding") qui est une version scalable (cf section A.2.3) de cette norme H.264/AVC, et "MVC" ("Multi-View Coding") pour le codage AVC de scènes multi-vues...

Toutes ces normes sont basées sur les mêmes principes de codage que l'on va maintenant présenter. La norme H.264 MPEG-4/AVC apporte de nombreuses innovations qui seront traitées ultérieurement.

A.2 Caractéristiques des différentes normes

Le codage vidéo repose sur quatre modules : l'estimation-compensation en mouvement, la transformation, la quantification et la codage entropique. Cependant, on a vu que le format des images d'entrées est à prendre en compte et qu'il reste des paramètres à définir pour ces quatre modules comme la taille de la fenêtre d'estimation de mouvement, sa précision et le pas de quantification.

Ces différents paramètres sont propres à chacune des normes présentées.

A.2.1 H.261

La norme H.261 [ITU90] "Video codec for audio-visual services at px64kbit/s" a été définie pour une application de visioconférence ayant un débit cible compris entre approximativement 64 kbit/s et 2048 kbit/s.

Le codeur opère sur une séquence d'images progressives à une fréquence de 30000/1001 soit approximativement 29.97 images par seconde, avec une tolérance de ± 50 images par minute.
Le format de ces images ne peut être que de deux types :

- QCIF en $4:2:0$ (soit 176×144 pixels pour Y et 88×72 pixels pour U et V).
- CIF en $4:2:0$ (soit 352×288 pixels pour Y et 176×144 pixels pour U et V).

Seul le mode $4:2:0$ est possible pour cette norme.

L'estimation-compensation en mouvement est optionnelle pour le codeur. Cependant, il peut y avoir un vecteur par macrobloc 16×16 dont les composantes sont entières (précision de 1 pixel) et comprises dans une fenêtre de $[-16;+15.5]$. L'unicité de ce vecteur impose que cette estimation ne peut être que de type P, il n'y a donc pas d'images B pour cette norme. De plus, ces vecteurs ne peuvent en aucun cas sortir de l'image.

La transformation utilisée est la DCT flottante sur des blocs 8×8 (cf (1.2)) :

$$\alpha(m) = \begin{cases} \frac{1}{\sqrt{32}} & \text{si m=0} \\ \frac{1}{4} & \text{sinon} \end{cases} \quad \text{et} \quad \beta(n) = \begin{cases} \frac{1}{\sqrt{32}} & \text{si n=0} \\ \frac{1}{4} & \text{sinon} \end{cases}$$

La quantification se fait avec le même pas de quantification pour tous les coefficients d'un macrobloc sauf pour le DC Intra. Pour ce coefficient, le pas de quantification est de 8 sans zone de mise à zéro, alors que pour les autres il y a 31 possibilités de pas de quantification compris entre 2 et 62 avec une zone de mise à zéro.

Le codage entropique s'effectue avec un codage à longueur variable VLC après un parcours en zigzag des coefficients. Cependant, les événements ne sont composés que de $(run, level)$, le *run* utilise 6 bits, le *level* 8 bits et un caractère EOB indique la fin du bloc.

Pour cette norme, le train binaire a une structure hiérarchique assez simple qui se résume pour une séquence à :

- image (pas de GOP)
- GOB (Group Of macroBlocks) (équivalent au slice de MPEG)

– macrobloc
– bloc

A.2.2 MPEG-1

La norme MPEG-1 [ISO93] [Sik97] "Coding of moving pictures and associated audio for digital storage media at up to about 1,5Mbit/s" a été définie pour le stockage de séquences vidéo sur CD-ROM ayant un débit cible d'environ 1.5 Mbit/s (1.15 Mbit/s pour la vidéo et 192 kbit/s pour l'audio Layer III). Ce débit cible doit, dans tous les cas, être inférieur à 1856 kbit/s.

Comme H.261, ce codeur ne traite que des images progressives dont la fréquence peut être de 23.976 (24000/1001), 24, 25, 29.97 (30000/1001) ou 30 images par seconde. Il y a des fréquences supplémentaires, et il en est de même pour la taille de ces images. En effet, les images doivent avoir moins de 576 lignes de 768 pixels, les formats possibles sont donc :
– QCIF en $4:2:0$ (soit 176×144 pixels pour Y et 88×72 pixels pour U et V).
– CIF en $4:2:0$ (soit 352×288 pixels pour Y et 176×144 pixels pour U et V).
– 4CIF en $4:2:0$ (soit 704×576 pixels pour Y et 352×288 pixels pour U et V).
– TV 4/3 en $4:2:0$ (soit 768×576 pixels pour Y et 384×288 pixels pour U et V).
De même que pour H.261, seul le mode $4:2:0$ est possible pour cette norme.

L'estimation-compensation pour la norme MPEG-1 est telle que décrite précédemment. En effet, il peut y avoir un vecteur par macrobloc 16×16 pour les images Inter P ou deux vecteurs pour les images Inter B. De plus, la précision pour cette estimation est de 1/2 pixel et les composantes des vecteurs doivent être comprises dans une fenêtre de $[-32;+32]$. La seule contrainte est qu'il ne peut pas y avoir plus de 2 images B successives (c'est-à-dire IBBPBBPBBP).

La transformation est toujours la DCT flottante sur des blocs 8×8 avec :

$$\alpha(m)\,(\text{respectivement}\beta(n)) = \begin{cases} \frac{1}{\sqrt{32}} & \text{si m=0 (respectivement si n=0)} \\ \frac{1}{4} & \text{sinon} \end{cases}$$

Tous les coefficients DCT sont quantifiés uniformément avec un pas de quantification Q. Cependant, un traitement particulier est ensuite appliqué aux coefficients DC. Chacun de ces coefficients est prédit à partir du dernier codé, car il y a généralement une corrélation forte entre les coefficients DC de deux blocs 8×8 adjacents.

Ces coefficients DCT quantifiés sont parcourus en zigzag et représentés en événements $(run, level)$. Ces événements sont ensuite codés par un codage à longueur variable VLC s'appuyant sur des tables.

Pour cette norme, le train binaire a une structure de séquence de la forme :
– GOP
– image
– slice
– macrobloc
– bloc

A.2.3 MPEG-2 et H.262

Les normes MPEG-2 [ISO94b] [Sik97] et H.262 [ITU94] "Generic coding of moving pictures and associated audio information" ont été définies pour la diffusion de télévision numérique ayant un débit cible compris entre 2 Mbit/s et 10 Mbit/s.

Les autres caractéristiques de ces normes sont multiples. En effet, ces normes se décomposent en "profiles" et "levels". Les "levels" (niveaux) définissent les données d'entrée (cf Tab.A.1(a)) alors que les "profiles" (profils) définissent les fonctionnalités du codeur (cf Tab.A.1(b)). Cependant, tous les "profiles" et "levels" ne sont pas compatibles. Seules quelques combinaisons sont possibles (cf Tab.A.2).
Quelque soit le profil, les fonctionnalités de MPEG-1 sont héritées, donc supportées, pour rester compatible avec cette norme.

La scalabilité permet de définir notre séquence dans une version de moins bonne qualité (ici qualité SNR ou taille des images) qu'on appelle couche de base, et de pouvoir repasser dans la version originale grâce à une ou plusieurs couches de rehaussement.
Pour la scalabilité SNR, toutes les images de la séquence sont uniformisées, c'est-à-dire que les variations de luminance sont atténuées, on introduit donc une sorte de bruit. Ceci permet de diminuer la quantité d'information à coder pour la couche de base. La couche de rehaussement est alors la différence entre cette couche de base et la séquence originale encodée (cf fig. A.3).
Pour la scalabilité spatiale, c'est le même principe, la couche de base est composée d'images réduites en résolution et la couche de rehaussement correspond à l'information nécessaire pour repasser en pleine résolution (cf fig. A.4).

FIG. A.3 – Exemple de scalabilité SNR sur 3 niveaux

FIG. A.4 – Exemple de scalabilité spatiale sur 3 niveaux

La scalabilité permet de s'adapter aux capacités du décodeur et aux réseaux de transmission. En effet, un décodeur peu puissant ou n'ayant pas accès à un débit important peut décoder uniquement la couche de base, il ne recevra que cette couche. Alors qu'un décodeur plus puissant ou pouvant recevoir un débit plus important recevra la couche de base et la couche de rehaussement, il décodera la séquence dans son format original.

(a) Bornes supérieures des paramètres de chaque niveau MPEG-2

Level	Paramètres
High	1920 lignes × 1152 pixels (HDTV) 60 images/s 80 Mbit/s
High 1440	1440 lignes × 1152 pixels 60 images/s 60 Mbit/s
Main	720 lignes × 576 pixels (TV) 30 images/s 30 Mbit/s
Low	352 lignes × 288 pixels (CIF) 30 images/s 4 Mbit/s

(b) Fonctionnalités supportés par chacun des profils de MPEG-2

Profile	Fonctionnalités
High	Supporte les fonctionnalités du "Spatial Scalable profile" avec : - 3 niveaux de scalabilité spatiale et SNR - le mode 4 : 2 : 2 pour améliorer la qualité
Spatial Scalable	Supporte les fonctionnalités du "SNR Scalable profile" plus : - la scalabilité spatiale sur 2 niveaux - le mode 4 : 0 : 0
SNR Scalable	Supporte les fonctionnalités du "Main profile" plus : - la scalabilité SNR sur 2 niveaux
Main	Inclut les fonctionnalités du "Simple profile" plus : - les images Inter interpolées B
Simple	Algorithme non scalable supportant : - l'accès aléatoire - les images entrelacées - le mode 4 : 2 : 0

TAB. A.1 – Récapitulatif des niveaux et profils MPEG-2

Pour ces normes, la transformation utilisée est toujours la DCT flottante 8×8, et le codage entropique est un codage à longueur variable VLC. Mais la quantification est matricielle, c'est-à-dire que chaque coefficient DCT d'un bloc 8×8 a un pas de quantification qui lui est propre, cela permet de mieux conserver les basses fréquences que les hautes (cf fig. A.5).

	Simple profile	Main profile	SNR Scalable profile	Spatial Scalable profile	High profile
High level		X			X
High 1440 level		X		X	X
Main level	X	X	X		X
Low level		X	X		

TAB. A.2 – Combinaisons possibles profil-niveau de MPEG-2

8	16	19	22	26	27	29	34
16	16	22	24	27	29	34	37
19	22	26	27	29	34	34	38
22	22	26	27	29	34	37	40
22	26	27	29	32	35	40	48
26	27	29	32	35	40	48	58
26	27	29	34	38	46	56	69
27	29	35	38	46	56	69	83

FIG. A.5 – Matrice de quantification utilisée par MPEG-2

A.2.4 H.263, H.263+ et H.263++

La norme H.263 [ITU95] "Video coding for low bit rate communication" a été définie pour des applications de visiophonie et de visioconférence "améliorée", ayant un débit cible compris entre 28.8 kbit/s et 640 kbit/s.

Comme H.261, ce codeur ne traite que des images progressives dont la fréquence doit être de 30000/1001 soit 29.97 images par seconde. Cependant, cette norme accepte des formats d'image supplémentaires qui sont :

- subQCIF en $4:2:0$ (soit 128×96 pixels pour Y et 64×48 pixels pour U et V).
- QCIF en $4:2:0$ (soit 176×144 pixels pour Y et 88×72 pixels pour U et V).
- CIF en $4:2:0$ (soit 352×288 pixels pour Y et 176×144 pixels pour U et V).
- 4CIF en $4:2:0$ (soit 704×576 pixels pour Y et 352×288 pixels pour U et V).
- 16CIF en $4:2:0$ (soit 1408×1152 pixels pour Y et 704×576 pixels pour U et V).

De même que pour H.261, seul le mode $4:2:0$ est possible pour cette norme.

Cette norme H.263 a été conçue sur la base de H.261. L'estimation-compensation en mouvement est effectuée de la même façon que celle de H.261, mais avec une précision au 1/2 pixel. Les mêmes méthodes de transformation, de quantification et de codage entropique (VLC avec des tables plus précises) ainsi que la structuration du train binaire sont identiques.

Cette norme possède en plus quatre modes optionnels [ITU95] [Rij96] (cités avec l'annexe de la recommendation ITU entre parenthèses) :

1. *Unrestricted Motion Vector mode (D) :* permet à l'estimation-compensation en mouvement d'utiliser des vecteurs sortant de l'image (en estimant les points extérieurs par symétrie)

et d'avoir une fenêtre de recherche plus grande [-31.5 ; +31.5].

2. *Syntax-based Arithmetic Coding mode (E) :* permet d'utiliser un codage arithmétique à la place du codage VLC.
3. *Advanced Prediction mode (F) :* permet d'utiliser 4 vecteurs mouvement par macrobloc (un pour chaque bloc 8×8) avec la même fenêtre [-16 ; +15.5] pour chacun des blocs ce qui donne pour un macrobloc une fenêtre de [-31.5 ; +31.5].
4. *PB-frames mode (G) :* permet un codage conjoint de 2 images, une en P l'autre en B.

H.263+ [ITU98] [CEGK98] apporte à H.263 un grand nombre de modes optionnels qui permettent d'améliorer la compression d'environ 15% par rapport à la version 1. Ces treize modes optionnels sont (cités avec l'annexe de la recommendation ITU entre parenthèses) :

1. *Unrestricted Motion Vector mode (D) :* permet à l'estimation-compensation en mouvement d'utiliser des vecteurs sortant de l'image (en estimant des points extérieurs par symétrie) et d'avoir une fenêtre de recherche plus grande allant jusqu'à [-256 ; +255.5]. De plus, lorsque ce mode est sélectionné, un nouveau codage VLC réversible (RVLC) est utilisé pour coder les informations de vecteurs de mouvement. Ce codage permet d'être plus robuste aux erreurs.
2. *Advanced Intra Coding mode (I) :* permet l'utilisation de trois prédictions spatiales différentes pour les blocs Intra. Cette prédiction peut être DC, le coefficient DC est prédit avec les DC des blocs voisins (au dessus et à gauche). Elle peut aussi être Vertical DC et AC, auquel cas la première ligne du bloc est prédite verticalement avec le bloc au dessus ou Horizontal DC et AC où la première colonne est prédite horizontalement avec le bloc de gauche. Ceci impose l'utilisation de trois types de parcours zigzag : le classique, l'alternatif vertical (pour les blocs Horizontal DC et AC) et l'alternatif horizontal (pour les blocs Vertical DC et AC), et de nouvelles tables VLC.
3. *Deblocking Filter mode (J) :* introduit un filtre adaptatif (différents pour les images I, P et B) dans la boucle de codage pour lisser les effets de blocs.
4. *Slice Structured mode (K) :* permet l'utilisation de slices plutôt que les GOB pour une meilleure synchronisation et une plus grande robustesse aux erreurs.
5. *Supplemental Enhancement Information mode (L) :* permet d'ajouter des informations au train binaire pour améliorer les possibilités d'affichage. Ces fonctionnalités sont le gel des images (freezing), l'extraction d'images fixes (snapshot) ou de sous-séquences (segmentation vidéo), le raffinement en qualité (une image peut raffiner une autre plutôt que d'être la suivante temporellement), et le verrouillage des chrominances (les chrominances ne sont pas mises à jour et on utilise une image de fond extérieure).
6. *Improved PB-Frames mode (M) :* permet l'utilisation de prédiction "forward", "backward" ou "bidirectionally" pour les blocs B (pas seulement "bidirectionally").
7. *Reference Picture Selection mode (N) :* permet d'utiliser une image de référence autre que la précédente pour éviter la propagation d'erreurs. Dans ce mode, la référence n'est plus la dernière image I ou P décodée, mais la dernière image I ou P bien décodée (sans erreur).
8. *Temporal, SNR, and Spatial Scalability mode (O) :* rend possible l'utilisation de la scalabilité temporelle, SNR et spatiale (cf fig. A.3 et fig. A.4).
9. *Reference Picture Resampling mode (P) :* permet d'utiliser une image de référence à une résolution différente de celle des images à coder. Cette image de référence n'est ré-échantillonnée à la résolution des images à coder que pour la prédiction.

10. *Reduced Resolution Update mode (Q) :* permet de coder les images à une résolution plus faible que l'originale, mais en utilisant des références à taille originale. Ce mode n'est pas intéressant pour des scènes avec beaucoup de mouvement.
11. *Independently Segmented Decoding mode (R) :* traite les frontières de GOB (ou de slices) comme les frontières de l'image en n'autorisant pas les dépendances à travers ces frontières.
12. *Alternative Inter VLC mode (S) :* permet d'utiliser les tables VLC Intra du mode Advanced Intra Coding pour les blocs Inter lorsque le codage à l'aide de ces tables est plus efficace.
13. *Modified Quantization mode (T) :* permet de modifier le pas de quantification d'un macrobloc à un autre (sans dépendance), d'utiliser un pas de quantification différent pour la luminance et les chrominances, et d'augmenter le nombre de coefficients DCT représentables.

De même que H.263+, H.263++ [ITU01] apporte quatre fonctionnalités supplémentaires à H.263 (citées avec l'annexe de la recommendation ITU entre parenthèses) :

1. *Enhanced Reference Picture Selection mode (U) :* permet de stocker plusieurs images de référence par utilisation d'un buffer pour rendre le codage plus efficace et être plus robuste aux erreurs. Ces images de référence peuvent être découpées en sous-images qui ne sont pas toutes utiles et donc pas conservées en mémoire. De plus, ce mode permet d'utiliser deux images de référence pour les images Inter B.
2. *Data-Partitioned Slice mode (V) :* réarrange le train binaire pour le rendre plus robuste aux erreurs. Les vecteurs de mouvement sont transmis en premier, suivis par les coefficients DCT. Un marqueur de synchronisation est introduit entre les deux. Avec ce mode, une erreur sur les mouvements d'un paquet n'influera pas sur les coefficients DCT, alors que sans ce mode les mouvements étaient entrelacés avec les coefficients DCT, d'où une erreur sur un paquet entraînait la perte du paquet.
3. *Additional Supplemental Enhancement Information specification (W) :* permet d'ajouter des informations complémentaires au train binaire comme du texte (copyright, légende, description,...), répétition des données d'en-tête, indication de trames entrelacés, et d'identification d'images de référence de réserve.
4. *Profiles And Levels definition (X) :* définit des profiles et des levels comme dans MPEG-2, les profiles déterminant les modes (de l'annexe D à W) qui sont utilisables (cf [ITU04]).

A.2.5 MPEG-4 partie 2

La norme MPEG-4 [ISO00a] [Sik97] "Generic coding of audio-visual objects" a été définie à l'origine pour des applications de très bas débit, mais elle est vite devenue une norme générique ayant un débit cible compris entre 5 kbit/s et 1 Gbit/s.

Cette norme est générique, elle est donc capable de traiter aussi bien les images progressives que les entrelacées, les images sub-QCIF que les images "studio" (4k $\times$ 4k), et les scènes réelles que les scènes de synthèses 2D ou 3D. Elle traite tous types d'images à tous les formats.

Pour cette norme, une nouvelle structure de données basée objets a été adoptée. La scène est découpée en Objets AudioVisuels (VO) qui sont codés indépendamment (cf fig. A.6).

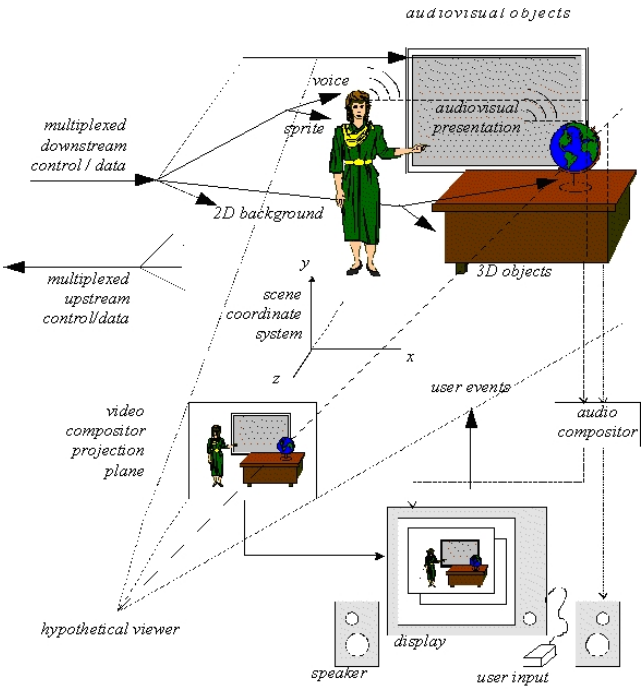


FIG. A.6 – Description d'une scène avec la norme MPEG-4 [ISO00a]

Pour chaque Objet Video, on code alors la forme, le mouvement et la texture avec les techniques classiques (comme dans MPEG-2).

Les formes sont codées à l'aide de masques binaires. Pour chaque objet (VO), un plan (VOP) est défini comme étant le plus petit rectangle contenant cet objet. Le masque consiste à dire si les pixels de ce plan (VOP) appartiennent ou non à l'objet. De plus, lorsque des BIFS (BInary Format for Scenes, descripteurs de scènes) sont utilisés, ils permettent de définir l'emplacement des objets les uns par rapport aux autres, leur orientation,...

L'estimation-compensation pour la norme MPEG-4 peut, comme dans le *Advanced Prediction mode* de H.263, être effectuée sur les macroblocs 16×16 ou sur les blocs 8×8, mais avec une précision pouvant aller jusqu'au 1/4 pixel. Cette estimation est basée blocs, mais elle tient compte des frontières d'objets et ne peut pas sortir de l'objet considéré.

Deux types de texture sont à coder, les textures dynamiques des objets en mouvement, et les textures statiques du fond.
La texture des blocs entièrement contenus dans un objet ainsi que celle des blocs frontières complétés par "padding" (les pixels hors de l'objet sont estimés à partir de ceux de l'objet) sont codées par une DCT adaptée aux contours (Shape-Adaptive DCT SA-DCT), quantification, et codage entropique VLC.
Les textures statiques sont décomposées en ondelettes et codées à l'aide d'arbre de zéros (zerotree).

Cette norme possède aussi plusieurs outils dont l'utilisation dépend de l'application visée, c'est-à-dire qu'on utilisera peu d'outils pour des applications bas débit et beaucoup d'outils pour les hauts débits. Ces différents outils sont :

1. *Content-Based Functionalities :*
   - "Content-based coding" permet le décodage et la reconstruction de n'importe quel objet vidéo indépendamment des autres.
   - "Random access" permet de faire pause, avance rapide et retour rapide.
   - "Extended manipulation" permet d'incruster du texte, des textures, des images ou des vidéos dans la séquence reconstruite comme, par exemple, une légende qui suit un objet en mouvement.
2. *Scalability of Textures, Images and Video :*
   - "Complexity scalability" permet à tout décodeur de décoder la séquence quelque soit sa complexité, l'encodeur doit alors générer plusieurs flux de complexités différentes, par exemple, le premier flux sera généré sans outil, les flux de rehaussement intégreront des outils.
   - "Spatial scalability" permet une scalabilité spatiale comme définie précédemment (cf fig. A.4), mais jusqu'à 11 niveaux.
   - "Temporal scalability" permet une scalabilité temporelle jusqu'à 3 niveaux. Pour cette scalabilité, la couche de base n'est pas composée de toutes les images de la séquence, les couches de rehaussement apportent ces images supprimées pour repasser à la séquence originale.
   - "Quality scalability" permet une scalabilité SNR comme définie précédemment (cf fig. A.3).
   - "Fine Grain scalability" est un cumul de toutes ces scalabilités dans les versions les plus fines, jusqu'à 11 niveaux pour chacune.
3. *Shape and Alpha Channel Coding :* permet de définir la transparence des objets.
4. *Robustness in Error Prone Environments :* permet une meilleure robustesse aux erreurs de transmission grâce à des nouvelles techniques d'error resilience (décrites ultérieurement avec le train binaire).
5. *Face and Body Animation :* permet d'animer des visages et des corps de synthèse.
6. *Coding of 2-D Meshes with Implicit Structure :* permet de coder des objets en 2D à l'aide d'une décomposition en meshes (structure autre que des blocs, généralement des triangles) et de modèles de déformation.
7. *Coding of 3-D Polygonal Meshes :* même chose que précédemment, mais pour des objets en 3D. Cet outil a été par la suite complétée par une extension : AFX (Animation Framework eXtension).

Un outil intéressant du "Content-based" est le "sprite". Il permet la transmission efficace des scènes de fond où les changements dans le contenu de ce fond sont principalement provoqués par un mouvement de caméra. Ainsi, un sprite statique est une grande image immobile (panorama statique de fond) qui est transmise au récepteur avec les paramètres de caméra de sorte que la partie appropriée de la scène puisse être retracée (ou déformée) (cf fig. A.7).

La correction d'erreurs est améliorée, dans cette norme, grâce à la structure du train binaire (cf fig. A.8), au partitionnement des données (comme dans H.263++) et à l'utilisation de code RVLC (Reversible VLC, décodable dans les deux sens). Une fois l'erreur détectée, on se resynchronise sur le marqueur suivant et les données sont décodées en sens inverse pour en récupérer un maximum. Si trop de données ont été perdues, alors elles sont remplacées par les informations des images suivantes ou précédentes.

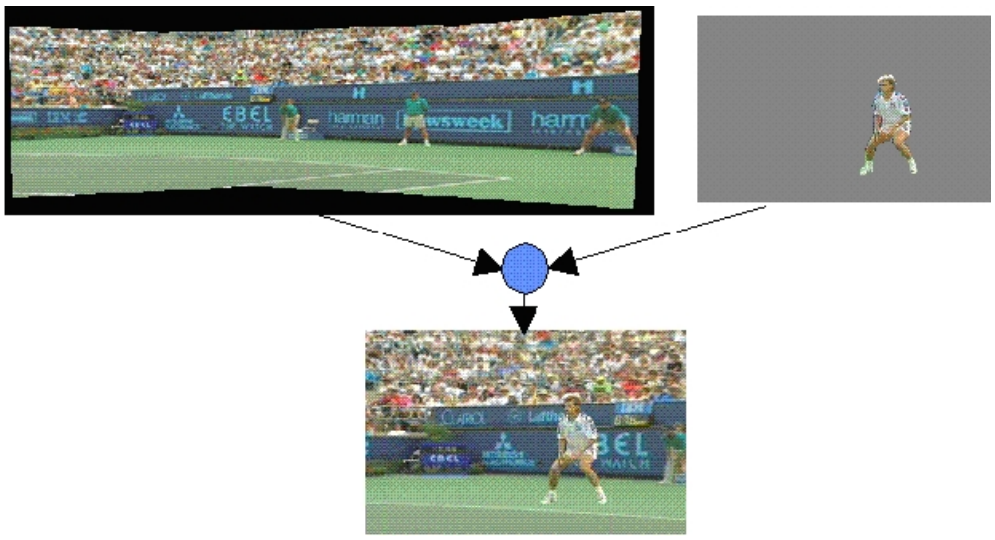


FIG. A.7 – Exemple de codage à l'aide de sprite dans la norme MPEG-4 [ISO00a]

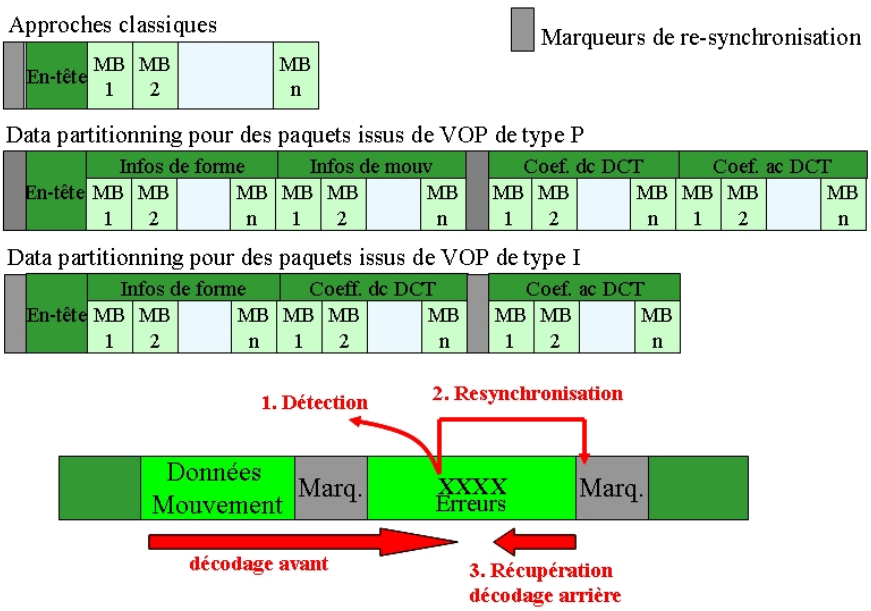


FIG. A.8 – Structure du train binaire et technique de correction d'erreurs de la norme MPEG-4

Cette norme possède donc un grand nombre d'outils, mais ceux-ci impliquent une forte augmentation de la complexité au codage et au décodage. En plus de ces outils, il y a des extensions qui ont été définies comme IPMP (Intellectual Property Management and Protection), AFX (Animation Framework eXtension),... Il existe une autre extension à cette norme MPEG-4 qui ne définit pas un outil, mais un nouveau schéma complet de codage vidéo : AVC (Advanced Video Coding), soit H.264 MPEG-4 AVC (partie 10) plus efficace en compression que MPEG-4 vidéo (partie 2), mais dépourvue du mode objet.

A.3 H.264 MPEG-4/AVC

Les groupes de travail de l'ITU-T (SG16) et de l'ISO (MPEG) ont approuvé le fusionnement de leurs équipes vidéo sous le nom de JVT (Joint Video Team) en décembre 2001. Le but de cette nouvelle entité était de standardiser un codec vidéo commun H.264/AVC (mai 2003). Elle recherche actuellement à en améliorer les performances avec l'amendement "FRExt" [JVT04] notamment et à adapter ce codec à toutes les situations avec "SVC" [JVT06] qui permet la scalabilité du flux vidéo et "MVC" qui permet le codage multi-vues.

A.3.1 Introduction de la norme H.264/AVC

La norme H.264/AVC [JVT05] n'apporte pas de rupture de technologie par rapport aux normes précédentes. Les différences se situent sur chacun des outils de codage (estimation du mouvement, transformation, quantification et codage entropique).

Comme la norme MPEG-2, cette norme utilise des "profiles" et "levels" qui permettent de mieux s'adapter à l'application visée en imposant des restrictions.
Les niveaux définissent entre autres les données d'entrée supportées (cf Tab.A.3), il y en a 13.

Niveau	Paramètres (p :progressive, i :interlaced)
1	QCIF @ 15fps (frames per second)
1.1	QCIF @ 30fps
1.2	CIF @ 15fps
2	CIF @ 30fps
2.1	HHR (Horizontal High Resolution) @ 15 ou 30fps
2.2	SDTV @ 15fps
3	SDTV : 720x480x20i - 720x576x25i 10Mbps(max)
3.1	1280x720x30p - SVGA (800x600)x50p
3.2	1280x720x60p
4	HDTV : 1920x1080x30i - 1280x720x60p - 2kx1kx30p 20Mbps(max)
4.1	HDTV : 1920x1080x30i - 1280x720x60p - 2kx1kx30p 50Mbps(max)
5	SHDTV (Super HDTV)/D-Cinema : 1920x1080x60p - 2.5kx2k
5.1	SHDTV/D-Cinema : 4kx2k

Tab. A.3 – Bornes supérieures des paramètres pour chaque niveau de H.264/AVC

Les profils, quant à eux, spécifient les outils utilisables (cf Tab.A.4), ils sont au nombre de 3.

Tous les outils et spécificités non cités dans ce tableau Tab.A.4 sont inclus dans tous les profils.

Cette norme [TR03] [WSBL03] [SWS03] [LF03] [Ric02b] s'inscrit toujours dans le même schéma de codage hybride (cf fig. 1.3 et 1.12), mais les techniques utilisées pour mettre en œuvre les principes généraux de ce schéma la rendent plus efficace que les précédentes normes (jusqu'à 50% par rapport à MPEG-2). En contrepartie, ces améliorations et les nombreux outils supplémentaires (cf Tab.A.4) font beaucoup croître la complexité de la norme (jusqu'à 5 fois).

Profil	FMO/ASO	B-slice	Weight pred.	CABAC	CAVLC	MBAFF	SP/SI
Baseline	X				X		
Main		X	X	X		X	
Extended	X	X	X		X		X

TAB. A.4 – Fonctionnalités supportés par chacun des profils de H.264/AVC

A.3.2 La prédiction Intra

Contrairement aux autres normes, cette norme apporte une prédiction Intra [Ric03b]. Si un macrobloc ou un bloc est à coder en Intra, une prédiction spatiale est effectuée. Pour un macrobloc 16×16, 4 modes sont disponibles. Et pour les blocs 4×4 (pas sur les blocs 8×8) 9 modes sont possibles.

Modes de prédiction 4×4

Cette prédiction se base sur les pixels au dessus et à gauche du bloc à coder. Cependant, ces pixels doivent avoir été encodés et décodés (mais pas filtrés) et appartenir au même "slice" pour pouvoir être utilisés.

Les 9 modes potentiels sont les 9 directions possibles pour la prédiction (cf fig. A.9).

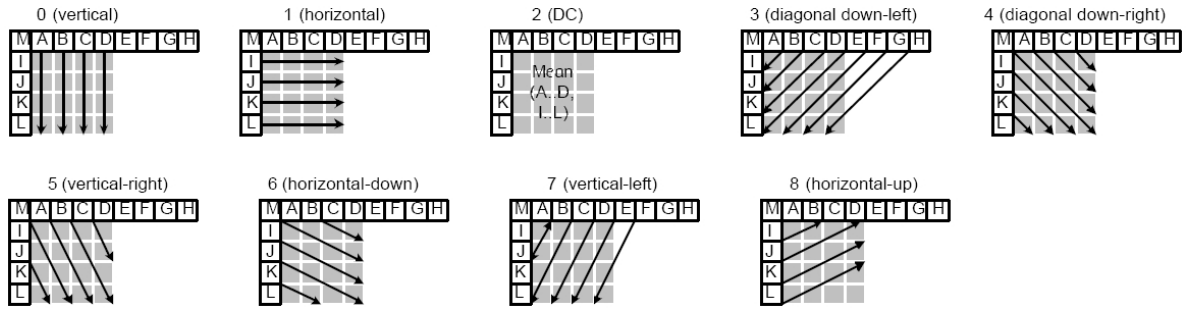


FIG. A.9 – Les 9 modes de prédiction 4×4 Intra [Ric03b]

Pour les modes 3 à 8, la prédiction est une moyenne pondérée des pixels A à M.

Il est possible que tous les pixels ne soient pas disponibles dans le même slice. Pour conserver un maximum de flexibilité, il est alors possible de modifier ces modes.
Le mode DC (0) s'adapte suivant les pixels disponibles.
Les autres modes (1 à 8) ne peuvent être utilisés que s'il manque juste E, F, G et H auquel cas ces valeurs sont recopiées de D.

Le codeur choisit pour chacun des blocs le mode qui minimise la différence entre la prédiction et le bloc, en s'appuyant sur les modes des blocs voisins.

Modes de prédiction 16×16

Il est aussi possible d'effectuer la prédiction Intra sur des macroblocs 16×16. 4 modes sont possibles (cf fig. A.10) :

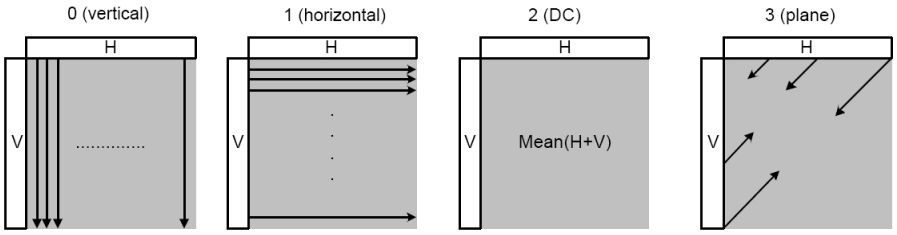


FIG. A.10 – Les 4 modes de prédiction 16×16 Intra [Ric03b]

A.3.3 L'estimation-compensation en mouvement

L'estimation-compensation en mouvement [Ric03a] reste basée macroblocs, mais elle a été beaucoup améliorée grâce à la précision au 1/4 pixel (comme MPEG-4), à l'utilisation de références multiples (comme H.263++) et à une grande variété de tailles et de formes de macroblocs.

En effet, chaque macrobloc 16×16 peut être divisé de 4 façons : 16×16, 16×8, 8×16 et 8×8, ce sont des partitions de macroblocs (cf fig. A.11). Si la partition choisie est 8×8, alors cette partition peut être de nouveau divisée de 4 façons : 8×8, 8×4, 4×8 et 4×4, ce sont les sous-partitions de macroblocs (cf fig. A.11).

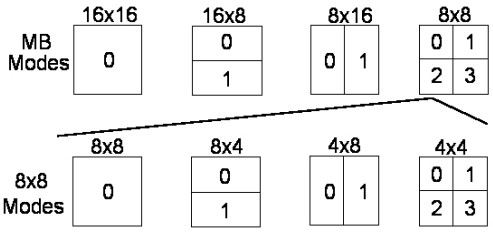


FIG. A.11 – Le partitionnement et sous-partitionnement des macroblocs [SWS03]

Chacune de ces partitions ou sous-partitions de macroblocs nécessite un vecteur de mouvement. Les partitions seront donc utilisées pour les zones homogènes, et les sous-partitions pour les zones de détails. La sélection du partitionnement des macroblocs se fait par optimisation du rapport débit-distorsion (minimum de débit pour une qualité donnée ou vice-versa).

Ces vecteurs de mouvement sont estimés au 1/4 pixel à partir de macroblocs (ou partitions) de référence de même taille. Les images de référence utilisées ici peuvent être multiples comme dans H.263++ (Annexe U), et ce jusqu'à 5 images de référence. On a donc aussi à transmettre l'indice de la référence. Pour obtenir cette précision, les demi-pixels sont d'abord interpolés à partir des voisins entiers à l'aide d'un filtre FIR d'ordre 6. Ensuite, lorsque tous les demi-pixels

ont été calculés, les quart de pixels sont interpolés bilinéairement à partir des pixels entiers et des demi. Ces vecteurs de mouvement peuvent sortir de l'image comme dans H.263 (Annexe D), les pixels hors de l'image étant recopiés par symétrie.

De plus, les vecteurs mouvement sont souvent très corrélés avec ceux du voisinage. Ils sont donc prédits (pour chaque composante) comme étant la médiane des vecteurs mouvement des partitions au dessus, de gauche et de la diagonale droite en haut. Seule la différence entre le vecteur réel et le vecteur prédit est transmise.

Pour les chrominances, le même travail est effectué. Puisque le format utilisé par la norme H.264/AVC est le 4 :2 :0, les chrominances sont deux fois plus petites que la luminance. Le même partitionnement que pour les luminances est conservé, mais divisé par deux (un partitionnement 8×16 de la luminance correspondra à un 4×8 pour les chrominances, un 8×4 à un 4×2,...). Les vecteurs mouvement des luminances sont divisés par deux et appliqués directement aux chrominances impliquant une précision au 1/8 pixel.

A.3.4 La transformation

Comme on a pu le voir précédemment, de nouvelles tailles de macroblocs ont été introduites dans la norme H.264/AVC. Ceci implique que la transformation a aussi due être modifiée [Ric03d] [Wie03] [MHKK03] pour pouvoir s'adapter à ces nouvelles tailles de macroblocs.

En effet, la transformation est appliquée sur des blocs 4×4 (unité de codage) et non plus sur des blocs 8×8 comme les normes précédentes. De plus, ces normes antérieures utilisaient la DCT flottante, c'est-à-dire la DCT calculée en réel avec un arrondi à l'entier le plus proche. La norme H.264/AVC utilise quant à elle une DCT entière, c'est-à-dire une approximation de la DCT mais calculée en entier. Cette norme utilise trois types de transformation dépendant des données à traiter.

La première transformation permet de traiter les coefficients résiduels des blocs 4×4, qu'ils soient Intra ou Inter. Cette transformation 4×4 est entière et sans multiplication, elle n'est réalisée qu'avec des additions et des décalages. Ceci implique qu'elle est peu complexe. Cette transformation est matricielle et s'écrit de la forme :

$$Y = H \,.\, X \,.\, H^T \qquad \text{avec} \quad H = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix} \tag{A.1}$$

La transformation inverse est obtenue simplement par :

$$X = H_{inv} \,.\, Y \,.\, H_{inv}^T \qquad \text{avec} \quad H_{inv} = \begin{bmatrix} 1 & 1 & 1 & 1/2 \\ 1 & 1/2 & -1 & -1 \\ 1 & -1/2 & -1 & 1 \\ 1 & -1 & 1 & -1/2 \end{bmatrix} \tag{A.2}$$

La seconde transformation s'applique aux coefficients DC de macroblocs non partitionnés et codés en Intra. Tous les sous-blocs 4×4 sont d'abord codés en utilisant la DCT décrite précédemment (cf A.1), puis les coefficients DC de tous ces blocs (W_{DC} bloc 4×4 contenant tous ces

DC) sont codés en utilisant une transformation de Hadamard définie comme suit :

$$Y_{DC} = H_{DC} \,.\, W_{DC} \,.\, H_{DC}^T \qquad \text{avec} \quad H_{DC} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \tag{A.3}$$

La transformation inverse est, dans ce cas, identique à la transformation directe.

La troisième transformation correspond aux coefficients DC des chrominances. Chaque bloc 4×4 de chrominance (Cr et Cb) associé à un bloc 16×16 de luminance est transformé en utilisant la première transformation (cf (A.1)). Et, de même que pour les coefficients DC des macroblocs non-partitionnés Intra, les coefficients DC des chrominances sont regroupés en blocs 2×2 puis transformés en utilisant :

$$Y_{chDC} = H_{chDC} \,.\, W_{chDC} \,.\, H_{chDC}^T \qquad \text{avec} \quad H_{chDC} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{A.4}$$

La transformation inverse est, dans ce cas aussi, identique à la transformation directe.

Et, de même que pour les normes précédentes, ces blocs transformés sont ensuite quantifiés pour réduire la quantité d'information qu'ils contiennent.

A.3.5 La quantification

H.264/AVC utilise une quantification scalaire [Ric03d] [MHKK03] comme les normes précédentes, mais elle diffère des précédentes par le fait qu'elle ne soit pas linéaire. En effet, le pas de quantification $Qstep$ est incrémenté de 12.5% à chaque pas. Il en existe 52 valeurs indexées par QP (cf Tab.A.5). Cet incrément de 12.5% se traduit aussi par $Qstep$ double chaque fois que QP est incrémenté de 6.

QP	0	1	2	3	4	5	...	10	...	18	...	24	...	51
Qstep	0.625	0.6875	0.8125	0.875	1	1.125		2		5		10		224

TAB. A.5 – Pas de quantification pour la norme H.264/AVC

La quantification directe

Le moyen le plus simple d'effectuer une quantification scalaire est d'utiliser la formule (1.4) définie dans la première partie, soit :

$$Z_{(i,j)} = \left\lfloor \frac{Y_{(i,j)}}{Qstep} \right\rfloor$$

Cependant, l'approximation de la DCT utilisée lors de la transformation nécessite d'introduire des facteurs de remise à l'échelle des coefficients. Ces facteurs PF dépendent de la localisation des coefficients dans le bloc, et s'organisent comme suit :

Localisation	PF
(0,0), (2,0), (0,2) ou (2,2)	1/4 (a^2 avec $a = 1/2$)
(1,1), (1,3), (3,1) ou (3,3)	1/10 ($\frac{b^2}{4}$ avec $b = \sqrt{2/5}$ approximation de $\sqrt{1/2}cos(\pi/8)$)
les autres	$1/4\sqrt{2/5}$ (ab/2)

TAB. A.6 – Facteurs PF de remise à l'échelle des coefficients transformés

La quantification est alors donnée par :

$$Z_{(i,j)} = \left\lfloor Y_{(i,j)} . \frac{PF}{Qstep} \right\rfloor$$

Cependant, pour éviter d'avoir à réaliser une division flottante, on utilisera plutôt :

$$Z_{(i,j)} = \left\lfloor Y_{(i,j)} . \frac{MF}{2^{qbits}} \right\rfloor \qquad \text{avec} \quad qbits = 15 + \lfloor QP/6 \rfloor \tag{A.5}$$

Pour l'implémentation, il est plus intéressant d'utiliser une version en arithmétique entière de l'équation (A.5) qui s'écrit :

$$\begin{cases} |Z_{(i,j)}| = (|Y_{(i,j)}|.MF + f) >> qbits \\ sign(Z_{(i,j)}) = sign(Y_{(i,j)}) \end{cases} \tag{A.6}$$

avec f qui vaut $2^{qbits}/3$ pour les blocs Intra et $2^{qbits}/6$ pour les blocs Inter. f est un paramètre permettant de contrôler la zone de mise à zéro (deadzone) (cf fig. A.12). Cette zone sera donc plus petite pour les images Intra conservant ainsi plus de coefficient non nuls pour une meilleure reconstruction (indépendante) que pour les images Inter.

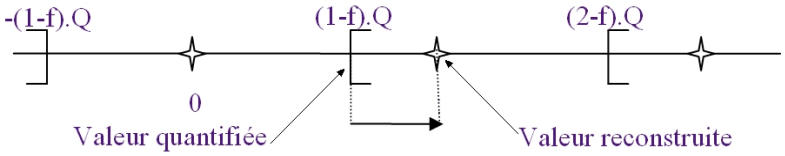


FIG. A.12 – Gestion de la zone de mise à zéro avec f

Pour $QP \leq 5$ les valeurs de MF peuvent être calculées à partir de PF, $Qstep$ et $qbits$, mais pour $QP > 5$, les valeurs de MF restent inchangées, seule la valeur de $qbits$ varie. Les valeurs de MF sont :

La quantification inverse

La quantification inverse sera donnée, dans sa version simple utilisée par les autres normes, par :

$$Y'_{(i,j)} = Z_{(i,j)} . Qstep$$

Qui devient, en incorporant la remise à l'échelle :

$$Y'_{(i,j)} = Z_{(i,j)} . Qstep . PF . 64$$

QP	Positions (0,0), (2,0), (0,2) ou (2,2)	Positions (1,1), (1,3), (3,1) ou (3,3)	Autres positions
0	13107	5243	8066
1	11916	4660	7490
2	10082	4194	6554
3	9362	3647	5825
4	8192	3355	5243
5	7282	2893	4559

TAB. A.7 – Facteurs de multiplication MF

Et, de même que précédemment, pour éviter les multiplications flottantes (Qstep non entier) on utilisera plutôt :

$$Y'_{(i,j)} = Z_{(i,j)}.V_{(i,j)}.2^{\lfloor QP/6 \rfloor} \tag{A.7}$$

La norme H.264/AVC ne définit pas les valeurs de pas de quantification $Qstep$, mais elle définit les valeurs du facteur de déquantification $V_{(i,j)}$ pour $QP \leq 5$, comme pour MF. Ces valeurs sont :

QP	Positions (0,0), (2,0), (0,2) ou (2,2)	Positions (1,1), (1,3), (3,1) ou (3,3)	Autres positions
0	10	16	13
1	11	18	14
2	13	20	16
3	14	23	18
4	16	25	20
5	18	29	23

TAB. A.8 – Facteurs de déquantification $V_{(i,j)}$

Ensuite, comme pour MF, pour $QP > 5$, les valeurs de $V_{(i,j)}$ restent inchangées et seules les valeurs de $\lfloor QP/6 \rfloor$ varient.

Les différentes quantifications

Comme on a pu le voir précédemment, il existe trois types de transformations qui dépendent du bloc traité et de son partitionnement.

- La première transformation s'applique sur des blocs 4×4 résiduels Intra ou Inter (cf (A.1)). Elle est utilisée dans la plupart des cas.
  La quantification utilisée est celle décrite précédemment avec (A.6) et pour l'inverse (A.7).

- La seconde transformation s'applique aux coefficients DC de luminance d'un macrobloc 16×16 codé en Intra16×16 (cf (A.3)).
  Après avoir divisé par 2 et arrondi les coefficients Y_{DC}, on les quantifie en utilisant une

version modifiée de (A.6) :

$$\begin{cases} |Z_{DC(i,j)}| = (|Y_{DC(i,j)}|.MF_{(0,0)} + 2f) >> (qbits + 1) \\ sign(Z_{DC(i,j)}) = sign(Y_{DC(i,j)}) \end{cases} \quad (A.8)$$

Dans ce cas, la quantification inverse est réalisée après la transformation inverse (A.3). Les coefficients transformés quantifiés $Z_{DC(i,j)}$ sont transformés en coefficients quantifiés $W'^{Q}_{DC(i,j)}$. La quantification inverse est alors aussi une version modifiée de (A.7), elle est donnée par :

$$W'_{DC(i,j)} = W'^{Q}_{DC(i,j)}.V_{(0,0)}.2^{\lfloor QP/6 \rfloor - 2} \quad \text{si} \quad QP \geq 12 \quad (A.9)$$
$$W'_{DC(i,j)} = [W'^{Q}_{DC(i,j)}.V_{(0,0)} + 2^{1-\lfloor QP/6 \rfloor}] >> (2 - \lfloor QP/6 \rfloor) \quad \text{si} \quad QP < 12$$

– La troisième transformation s'applique aux coefficients DC de chrominances (cf (A.4)). Ces coefficients Y_{chDC} sont quantifiés en utilisant une version modifiée de (A.6) :

$$|Z_{chDC(i,j)}| = (|Y_{chDC(i,j)}|.MF_{(0,0)} + 2f) >> (qbits + 1) \quad (A.10)$$
$$sign(Z_{chDC(i,j)}) = sign(Y_{chDC(i,j)})$$

Comme précédemment, la quantification inverse est réalisée après la transformation inverse (A.4). Les coefficients transformés quantifiés $Z_{chDC(i,j)}$ sont transformés en coefficients quantifiés $W'^{Q}_{chDC(i,j)}$. La quantification inverse est alors aussi une version modifiée de (A.7), elle est donnée par :

$$W'_{chDC(i,j)} = W'^{Q}_{chDC(i,j)}.V_{(0,0)}.2^{\lfloor QP/6 \rfloor - 1} \quad \text{si} \quad QP \geq 6 \quad (A.11)$$
$$W'_{chDC(i,j)} = [W'^{Q}_{chDC(i,j)}.V_{(0,0)}] >> 1 \quad \text{si} \quad QP < 6$$

Ces macroblocs transformés et quantifiés sont ensuite codés par codage entropique. La norme H.264/AVC définit de nouveaux codages entropiques : CAVLC (Context-based Adaptative VLC) et CABAC (Context-based Adaptative Binary Arithmetic Coding).

A.3.6 Le codage entropique

La norme H.264/AVC spécifie deux nouveaux types de codages entropiques. Ces codes sont, dans les deux cas, adaptatifs et contextuels, c'est-à-dire qu'ils utilisent le contexte (les blocs déjà codés) pour fournir des codes plus courts et mieux adaptés.

On a vu au début de cette partie que cette norme définit des profils et des niveaux (cf Tab.A.3 et Tab.A.4). Le type de codage utilisé sera donc défini par le profil (cf Tab.A.4) : CAVLC est utilisé en "Baseline Profile" et "Extend Profile", alors que CABAC est utilisé en "Main Profile".

Le codage VLC adaptatif contextuel

CAVLC (Context-based Adaptative Variable Length Coding) [Ric02c] est utilisé pour coder les résidus des blocs, les autres informations sont codées autrement, en utilisant un codage entropique Exp-Golomb.

Les codes Exp-Golomb (Exponential Golomb Codes) sont des codes à longueur variable construits de façon régulière selon : $[Mzeros][1][Info]$ où Info est l'information sur M bits.
Chacun de ces codes a une longueur de $(2M+1)$ et peut être construit à partir de l'index du code "code_num" :

- $M = \lfloor log_2(code\_num + 1) \rfloor$
- $Info = code\_num + 1 - 2^M$

Ils seront décodés par :

- lire le nombre M de zéros suivis d'un 1
- lire les M bits d'$Info$
- $code\_num = 2^M + Info - 1$

Les 9 premiers codes Exp-Golomb sont :

code_num	mot code
0	1
1	010
2	011
3	00100
4	00101
5	00110
6	00111
7	0001000
8	0001001

TAB. A.9 – Codes Exp-Golomb

Un paramètre v à coder sera associé à un "code_num" de trois façons différentes suivant quel est ce paramètre :

1. $ue(v)$ (Unsigned direct mapping) réalise $code\_num = v$, elle est utilisée pour le type de macrobloc, l'index de l'image de référence,...
2. $se(v)$ (Signed mapping) réalise $code\_num = 2|v|$ si $v < 0$ ou $code\_num = 2|v| - 1$ si $v \geq 0$, elle est utilisée pour les différences de vecteurs de mouvement, les delta QP,...
3. $me(v)$ (Mapped symbols) associe v à un code_num en suivant des tables prédéfinies dans le standard, elle est utilisée pour le CBP (coded_block_pattern qui indique quels blocs d'un macrobloc contiennent des coefficients).

Une fois ces différents paramètres encodés, il reste à coder les résidus des blocs en utilisant CAVLC. Ce codage s'applique aux blocs 4×4 (de luminance et 2×2 de chrominances) réarrangés en événements $(run, level)$ après un parcours en zigzag, et se déroule en 5 étapes.

1. Tout d'abord, le nombre de coefficients non nuls et le nombre de ± 1 terminant le bloc ("T1s") sont encodés sous le nom de "coeff_token". Le nombre de coefficients non nuls est compris entre 0 et 16, et le nombre "T1s" de ± 1 terminant le bloc doit être inférieur ou égal à 3.

Pour coder "coeff_token", il existe 4 tables : Num_VLC0, Num_VLC1, Num_VLC2 et Num_FLC (soit 3 tables VLC et 1 table de codes à longueur fixe). Le choix de la table dépend des blocs voisins déjà codés et de leur nombre de coefficients non nuls N_i.
Si le bloc U au dessus et le bloc L à gauche sont disponibles, on calcule le paramètre N tel que :

$N = (N_U + N_L)/2$.
Si seul le bloc U est disponible, on a : $N = N_U$, et si c'est le bloc L on a : $N = N_L$.
Sinon le paramètre N est mis à 0.
Ce paramètre N dépendant du voisinage (adaptatif contextuel) permet de sélectionner la table VLC :

N	Table VLC
0 ou 1	Num_VLC0
2 ou 3	Num_VLC1
4, 5, 6 ou 7	Num_VLC2
8 ou plus	Num_FLC

TAB. A.10 – Choix de la table VLC pour "coeff_token"

2. Le signe des "T1s" (± 1 terminant le bloc) sont codés par un seul bit : 0 ="+" et 1 ="-". Ces signes sont codés en ordre inverse, en commençant par la plus haute fréquence.

3. Ensuite, les "levels" des coefficients restants sont codés en ordre inverse (en commençant par la plus haute fréquence). Il existe pour cela 7 tables VLC : Level_VLC0 jusqu'à Level_VLC6. Le choix de la table s'effectue de la manière suivante :

- Initialisation à la table Level_VLC0 (sauf s'il y a plus de 10 coefficients non nuls et moins de 3 "T1s" auquel cas on initialise à Level_VLC1).
- Encodage du coefficient non nul de plus haute fréquence.
- Si l'amplitude du coefficient codé dépasse un certain seuil on passe à la table VLC supérieure.

Ces seuils sont définis par la norme tels que :

Table VLC	Seuil pour changer de table
Level_VLC0	0
Level_VLC1	3
Level_VLC2	6
Level_VLC3	12
Level_VLC4	24
Level_VLC5	48
Level_VLC6	N/A (plus haute table)

TAB. A.11 – Seuils d'incrément de table VLC pour les "levels"

4. On encode ensuite le nombre total de zéros précédent le coefficient de plus haute fréquence. Ce nombre est la somme de tous les "runs" du bloc et il est codé à l'aide d'un codage VLC.

5. Enfin, les "runs" sont codés en ordre inverse en commençant par la plus haute fréquence, mais avec deux exceptions :

- S'il n'y a pas plus de zéros à coder (le "run" = le nombre total de zéros), il n'est pas nécessaire de coder les "runs" restants.

– Il n'est pas nécessaire de coder le dernier "run" (de la plus basse fréquence).

Le choix du code VLC dépend du nombre de zéros qu'il reste à coder ($ZerosLeft$), s'il reste 2 zéros on a besoin de 2 bits, s'il en reste 6 on a besoin de 3 bits,...

On va maintenant présenter un exemple de codage CAVLC :

0	3	-1	0
0	-1	1	0
1	0	0	0
0	0	0	0

TAB. A.12 – Bloc 4×4 transformé quantifié à coder par CAVLC

Après le parcours zigzag, on a : $0, 3, 0, 1, -1, -1, 0, 1, 0, \ldots$

On peut alors calculer :

– Le nombre total de coefficients non nul $TotalCoeffs = 5$

– Le nombre total de zéros $TotalZeros = 3$

– Le nombre de "T1s" $= 3$ (ne peut pas être supérieur à 3)

L'encodage se déroulera alors comme suit :

Element	Valeur	Code
coeff_token	$TotalCoeffs = 5, T1s = 3$	0000100
signe T1(4)	+	0
signe T1(3)	-	1
signe T1(2)	-	1
Level(1)	+1 (Level_VLC0)	1
Level(0)	+3 (Level_VLC1)	0010
$TotalZeros$	3	111
run(4)	$ZerosLeft = 3, run = 1$	10
run(3)	$ZerosLeft = 2, run = 0$	1
run(2)	$ZerosLeft = 2, run = 0$	1
run(1)	$ZerosLeft = 2, run = 1$	01
run(0)	$ZerosLeft = 1, run = 1$	pas de code, dernier coeff

TAB. A.13 – Codage CAVLC du bloc 4×4

Le train binaire transmis pour ce bloc sera alors : 000010001110010111101101.

Le codage arithmétique adaptatif contextuel

CABAC (Context-based Adaptative Binary Arithmetic Coding) [Ric02a] [MSW03] [MSB+02] [MW03] combine un codage arithmétique binaire adaptatif avec une modélisation du contexte.

L'encodage d'un élément par CABAC se déroule en trois principales étapes (cf fig. A.13) :

1. La binarisation
2. La modélisation du contexte
3. Le codage arithmétique binaire

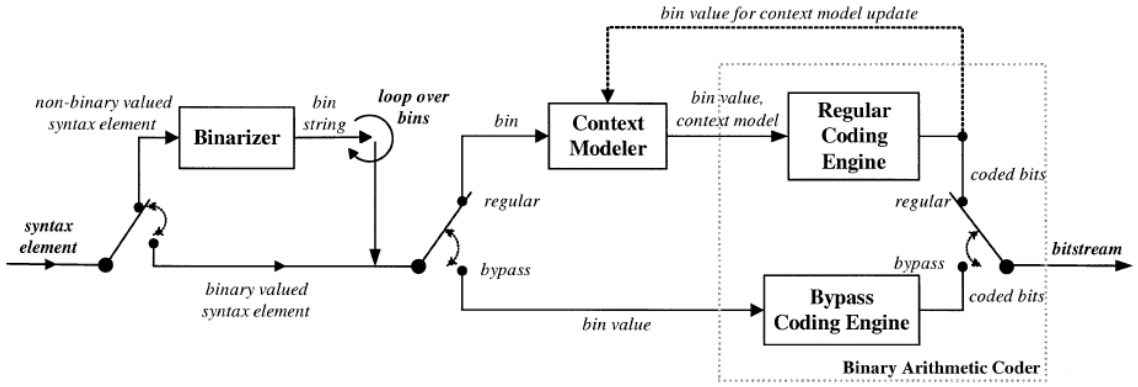


FIG. A.13 – Schéma général du codage CABAC [MSW03]

1. *La binarisation*

Elle permet de diminuer la taille de l'alphabet des éléments à coder en représentant chacun des éléments à coder par un mot binaire unique ("bin string"). Ce pré-traitement permet d'utiliser par la suite un codage arithmétique binaire adaptatif plutôt qu'un codage arithmétique m-aire adaptatif qui nécessite d'estimer beaucoup plus de probabilités, d'utiliser des multiplications et qui est beaucoup plus complexe.

Cette binarisation peut donc être vue comme une conversion en code à longueur variable, mais ce code est ensuite encore codé par codage arithmétique.

Cette binarisation peut être réalisée à l'aide de trois méthodes :

- le code unaire (U) ou unaire tronqué (TU) : pour tout entier non signé $x \geq 0$, le code unaire correspond à x bits à "1" avec un "0" pour terminer. Le code unaire tronqué s'applique à tout $0 \leq x \leq S$, pour $x < S$ il correspond au code unaire et pour $x = S$ le code ne contient que x bits à "1" (pas de "0").
- le code Exp-Golomb d'ordre k (EGk) : c'est une version modifiée des codes Exp-Golomb présentés précédemment. Ce code s'applique à un entier non signé x et se compose d'un préfixe et d'un suffixe. Le préfixe est un code unaire correspondant à la valeur $l(x) = \lfloor log_2(x/2^k + 1) \rfloor$. Le suffixe correspond à une représentation binaire de $x + 2^k(1 - 2^{l(x)})$ qui utilise $k + l(x)$ bits.
- le code à longueur fixe (FL) : pour $0 \leq x < S$, le code FL de x est donné par sa représentation binaire, mais limitée à $l_{FL} = \lceil log_2 S \rceil$ bits.

Ces quatre méthodes élémentaires sont ensuite concaténées pour donner les binarisations utilisées dans la norme. Ces méthodes dérivées sont au nombre de trois.

La première est la concaténation d'un préfixe FL sur 4 bits et d'un suffixe TU avec $S = 2$. Ce code permet de représenter le CBP (coded_block_pattern qui indique quels blocs d'un macrobloc contiennent des coefficients non nuls), le préfixe traduit la partie du CBP correspondant à la luminance et le suffixe la partie du CBP correspondant aux chrominances.

La seconde est la concaténation d'un préfixe TU et d'un suffixe EGk, elle est dénommée UEGk. Elle s'applique aux différences de vecteurs de mouvement mvd. Si $mvd = 0$ le code sera "0" autrement le préfixe est TU avec $S = 9$. Si $|mvd| \geq 9$, le suffixe est le code EG3 correspondant à la valeur $|mvd| - 9$ précédé du bit de signe ("1" pour - et "0" pour +). Sinon quand $0 < |mvd| < 9$ le suffixe n'est que le bit de signe.

La troisième est aussi un code UEGk, mais cette fois-ci elle s'applique aux valeurs absolues des coefficients transformés (abs_level-1 car on ne transmet pas les 0). Le préfixe est alors un code TU avec $S = 14$ et le suffixe un code EG0 (cf Tab.A.14).

abs_level	préfixe TU														suffixe EG0				
1	0																		
2	1	0																	
3	1	1	0																
4	1	1	1	0															
..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..
14	1	1	1	1	1	1	1	1	1	1	1	1	1	0					
15	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0				
16	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0		
17	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1		
18	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0
19	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1
..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..
bin	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19

TAB. A.14 – Exemple de binarisation : UEG0 des coefficients transformés

2. *La modélisation du contexte*

Une des propriétés les plus importantes du codage arithmétique est la possibilité d'utiliser une interface entre la modélisation et le codage. La modélisation permet d'assigner aux symboles un modèle de distribution de probabilité qui est ensuite utilisé par le codeur arithmétique pour produire une représentation de ces symboles en adéquation avec ce modèle de distribution. Il est donc important d'avoir des modèles statistiques bien adaptés aux données et de les garder à jour tout au long du codage.

La norme H.264/AVC définit 399 modèles statistiques représentés par leur indice de contexte γ, modèles qui dépendent de la donnée à coder et du type d'image auquel elle appartient (I, P ou B) (cf Tab.A.15).

Il faut maintenant déterminer une fonction permettant de calculer l'indice de contexte γ à utiliser pour coder le symbole. Les éléments de syntaxe présentés dans Tab.A.15 sont divisés en deux parties : la première $0 \leq \gamma \leq 72$ correspond aux données relatives aux macroblocs et au mode de prédiction, la seconde $73 \leq \gamma \leq 398$ correspond aux résidus des coefficients transformés. Il existe donc une fonction permettant de calculer γ pour chacune de ces parties.

Pour la première partie, on obtient γ selon :

$$\gamma = \Gamma_S + \chi_S \tag{A.12}$$

Élément de syntaxe	Type de slice		
	I	P	B
mb_type	0/3-10	14-20	27-35
mb_skip_flag		11-13	24-26
sub_mb_type		21-23	36-39
mvd (horizontal)		40-46	40-46
mvd (vertical)		47-53	47-53
ref_idx		54-59	54-59
mb_qp_delta	60-63	60-63	60-63
intra_chroma_pred_mode	64-67	64-67	64-67
prev_intra4×4_pred_mode_flag	68	68	68
rem_intra4×4_pred_mode	69	69	69
mb_field_decoding_flag	70-72	70-72	70-72
coded_block_pattern	73-84	73-84	73-84
coded_block_flag	85-104	85-104	85-104
significant_coeff_flag	105-165, 277-337	105-165, 277-337	105-165, 277-337
last_significant_coeff_flag	166-226, 338-398	166-226, 338-398	166-226, 338-398
coeff_abs_level_minus1	227-275	227-275	227-275
end_of_slice_flag	276	276	276

TAB. A.15 – Les éléments de syntaxe et leurs indices de contexte γ associés

où Γ_S est l'offset d'indice de contexte, il correspond à la borne inférieure de l'indice définie dans Tab.A.15, et χ_S est l'incrément d'indice de contexte de l'élément de syntaxe S, il est fonction du symbole et de l'indice du *bin* à coder.

Pour la seconde partie, on obtient γ en appliquant :

$$\gamma = \Gamma_S + \Delta_S(ctx\_cat) + \chi_S \tag{A.13}$$

où est utilisé en plus un offset Δ_S dépendant de la catégorie de contexte (ctx_cat). Ces catégories proviennent du type de transformée qui a été utilisée :

- Luma_Intra16_DC : catégorie 0
- Luma_Intra16_AC : catégorie 1
- Luma_4×4 : catégorie 2
- Chroma_DC : catégorie 3
- Chroma_AC : catégorie 4

Les offset Δ_S associés sont alors donnés par :

C'est donc χ_S qui permet de s'adapter et de garder à jour le modèle de contexte. Le calcul de cet incrément est basé sur le voisinage du bloc à coder (C avec A à gauche et B au dessus). Ces fonctions sont définies par la norme [JVT05] [MSW03] et un exemple va être présenté.

Pour *mvd*, différence de vecteurs de mouvement, et pour son premier *bin*, on a :

$$\chi_{mvd}(C) = \begin{cases} 0 & \text{si } e(A,B) < 3 \\ 1 & \text{si } 3 \leq e(A,B) \leq 32 \\ 2 & \text{si } e(A,B) > 32 \end{cases} \qquad \text{avec} \quad e(A,B) = |mvd(A)| + |mvd(B)|$$

Élément de syntaxe	Catégorie de contexte 0	1	2	3	4
coded_block_flag	0	4	8	12	16
significant_coeff_flag	0	15	29	44	47
last_significant_coeff_flag	0	15	29	44	47
coeff_abs_level_minus1	0	10	20	30	39

TAB. A.16 – Valeurs des offset Δ_S dépendant de la catégorie et de l'élément

On a vu précédemment que *mvd* est binarisé à l'aide d'un code UEG3 avec $S = 9$. Les *bin* 2 à 9 du préfixe utilisent les 4 autres modèles de contexte définis dans Tab.A.15 (*bin* 2 -> modèle 3, 3 -> 4, 4 -> 5, 5 -> 6, 6-9 -> 6). Et, les autres *bin* correspondant au signe et au EG3 sont codés en utilisant le "bypass coding mode" (cf fig. A.13).

3. *Le codage arithmétique binaire*

Chacun des modèles définis précédemment permettent de définir des probabilités d'apparition des *bin* ("0" ou "1"), ces probabilités sont : p_{LPS} (Least Probable Symbol) et p_{MPS} (Most Probable Symbol). Le codage arithmétique consiste à diviser récursivement un intervalle qui est défini par L sa borne inférieure et R sa longueur. La division de cet intervalle s'effectue de la façon suivante :

$$R_{LPS} = R \,.\, p_{LPS} \quad \text{et} \quad R_{MPS} = R - R_{LPS}$$

Pour s'affranchir de la multiplication, CABAC utilise une table construite à partir de Q_ρ et p_σ. Q_ρ est un ensemble de 4 approximations de R ($0 \leq \rho \leq 3$), et p_σ est un ensemble de 64 probabilités possibles pour p_{LPS} ($0 \leq \sigma \leq 63$).
Ces probabilités $p_\sigma \in [0.01875; 0.5]$ sont calculées à partir de l'équation récursive suivante :

$$p_\sigma = \alpha \,.\, p_{\sigma-1} \qquad \text{avec} \quad \alpha = \left(\frac{0.01875}{0.5}\right)^{1/63} \qquad \text{et} \quad p_0 = 0.5$$

Pour coder un *bin*, on calcule tout d'abord l'approximation $Q(R)$ de la longueur R de l'intervalle, cette approximation est définie par son indice ρ qui est calculé selon :

$$\rho = (R >> 6)\&3$$

On a maintenant ρ et σ (issu du modèle et mis à jour par le *bin* précédent), la table nous donne la valeur du produit $p_\sigma \,.\, Q_\rho$ correspondant à R_{LPS}. Ceci permet de calculer le nouvel intervalle $R = R - R_{LPS}$.

Si le *bin* est le MPS alors le sous-intervalle bas (le plus proche de 0) est sélectionné, la borne inférieure L de cet intervalle est inchangé et sa longueur R est celle qui a été calculée précédemment.
Sinon (LPS) c'est le sous-intervalle haut qui est sélectionné, la borne inférieure est mise à jour selon $L = L + R$ et sa longueur est affecté à $R = R_{LPS}$.

Dans les deux cas, le modèle de contexte est mis à jour en calculant la nouvelle valeur de σ. Si le *bin* est le MPS, alors il faut augmenter la probabilité p_{MPS} du modèle est réalisant :

$\sigma = min(\sigma + 1, 62)$.
Sinon c'est la probabilité p_{LPS} du modèle qu'il faut augmenter, cela est donné par une table ($TransIdxLPS[\sigma]$) définissant l'état σ atteint sachant qu'un LPS a été observé à l'état σ actuel (cf fig. A.14).

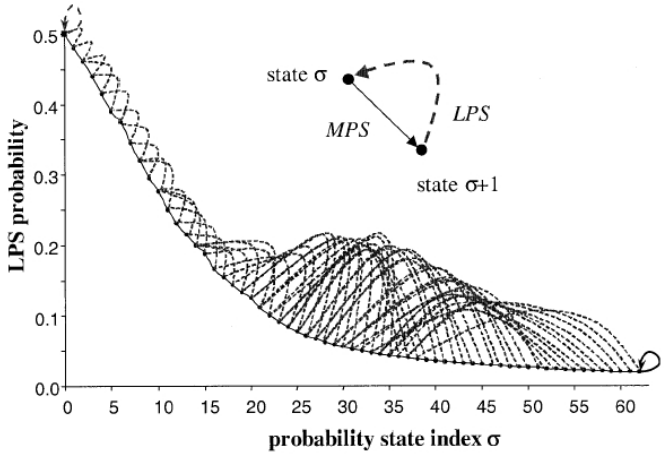


FIG. A.14 – Règles de transition pour la mise à jour des probabilités LPS pointillés) et MPS [MSW03]

Le codage alternatif "bypass" est utilisé pour améliorer la rapidité de l'encodage. Dans ce cas, les modèles de contexte ne sont pas pris en compte et il est considéré que les probabilités de MPS et LPS sont équiprobables. L'intervalle sera donc divisé par 2 à chaque *bin* codé (R est shifté de 1).

A.3.7 Le filtre réducteur d'effets de blocs

Une particularité du codage basé blocs est la production accidentelle de structures de blocs très visibles, considérée comme un des principaux artefacts de ces méthodes de compression. Les frontières des blocs sont reconstruites avec moins de précision que leurs pixels intérieurs.

Pour cela, H.264/AVC définit un filtre adaptatif dans la boucle de codage qui est appliqué à tous les macroblocs décodés après la transformée inverse du codeur ou du décodeur. Ce filtrage a deux avantages :

- les frontières de blocs sont lissés, améliorant la qualité des images décodées (particulièrement pour de forts taux de compression).
- les macroblocs filtrés sont utilisés pour l'estimation-compensation en mouvement, résultant en de plus faible résidus. après prédiction temporelle.

Ce filtrage est appliqué aux frontières verticales puis horizontales des blocs 4×4 des macroblocs (sauf pour les bords des images), en commençant par les luminances suivies des chrominances, comme illustré sur la figure A.15.

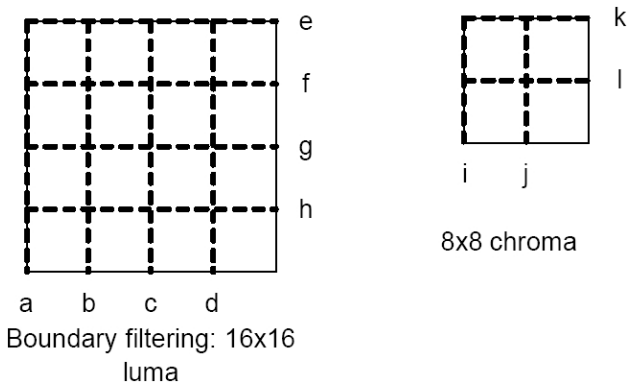
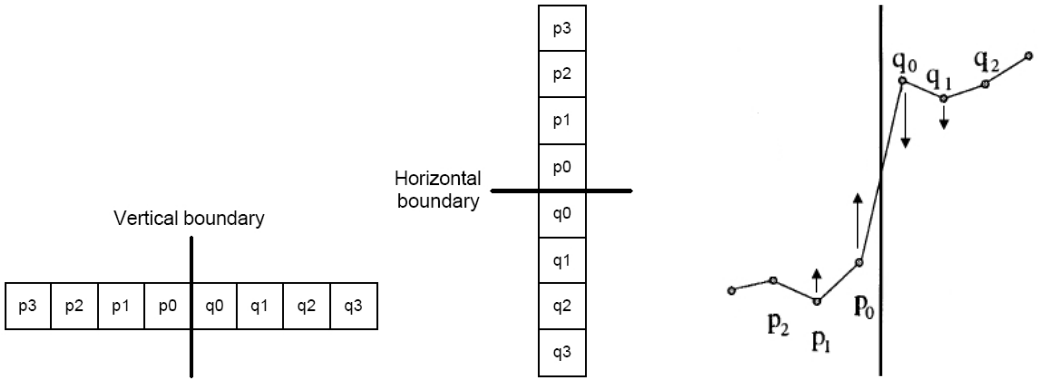


FIG. A.15 – Ordre de filtrage des frontières de blocs [Ric03c]

Chaque opération de filtrage affecte jusqu'à trois pixels de chaque côté de la frontière, comme illustré sur la figure A.16(b). La figure A.16(a) montre les quatre pixels des blocs adjacents p et q ($p0$,$p1$,$p2$,$p3$ et $q0$,$q1$,$q2$,$q3$) dont dépendent le choix du filtrage. En effet, la valeur de ces pixels associée à un seuil dépendant du pas de quantification permet d'autoriser ou non l'utilisation de ce filtre.



(a) Champ d'application du filtrage adaptatif (b) Action réalisé par ce filtrage

FIG. A.16 – Champ d'application et action du filtre réducteur d'effets de blocs [Ric03c]

Ce choix permet d'appliquer des filtrages plus ou moins important (touchant 1, 2 ou 3 pixels de chaque côté de la frontière). Il dépend alors de plusieurs critères :

- du mode de prédiction, Intra ou Inter.
- du type de frontière, frontière de macrobloc ou interne au macrobloc.
- de la présence ou non de coefficients dans chacun des blocs participants.
- de la valeur des vecteurs de mouvement.
- des images de référence utilisées et leur nombre.

Suivant l'importance du filtrage et du pas de quantification, ce filtrage peut donc aller de (a) aucun pixel n'est filtré jusqu'à (b) $p0$, $p1$, $p2$, $q0$, $q1$, $q2$ sont filtrés produisant les pixels $P0$, $P1$, $P2$, $Q0$, $Q1$ et $Q2$.

A.3.8 Les autres fonctionnalités de H.264/AVC

On a vu au début de ce chapitre que la norme H.264/AVC définissait des "profiles" comme MPEG-2 (cf Tab.A.4). Ces "profiles" spécifient les outils utilisables pour le codage, outils que l'on va maintenant présenter : B-slice, la prédiction pondérée et MBAFF.

De plus, cette norme utilise un certain nombre de mécanismes lui conférant une robustesse aux erreurs de transmission : FMO, ASO, SP/SI slices, Redundant slice, Data partitioning, et une bonne adaptation aux réseaux : "Parameter set", les NALU et les Access Units.

Les B-slices et la prédiction pondérée

Les B-slices

Les slices B [FWG98] [FG02] [FG03] correspondent à une généralisation des images B des normes précédentes, elles ne peuvent pas être utilisées dans le "Baseline profile".

Une différence majeure avec les standards antérieurs est qu'une image B peut ici avoir une autre image B en référence et pas seulement une I ou P. Et, comme on a pu le voir dans la partie estimation-compensation en mouvement, la prédiction Inter peut utiliser des références multiples.

La différence entre les P-slices et les B-slices est que les images B peuvent utiliser une moyenne pondérée de deux vecteurs de mouvement, un antérieur et un postérieur. Ces images B nécessitent donc de définir deux listes d'images de référence : $list0$ d'images antérieures et $list1$ d'images postérieures.

Dans les B-slices, la prédiction Inter peut être de quatre types : $list0$, $list1$, bi-prédiction et prédiction directe.
La prédiction $list0$ correspond à une prédiction arrière donc identique à une image P, la prédiction $list1$ correspond quant à elle à une prédiction avant.
La bi-prédiction fait une moyenne pondérée entre une prédiction $list0$ et une $list1$.
La prédiction directe est déduite des éléments de syntaxe déjà codés et peut être aussi bien $list0$, $list1$ que bi-prédiction. Si le mouvement peut être entièrement déduit des blocs déjà codés (mouvement global), il n'est pas nécessaire de transmettre d'information de mouvement et cette prédiction directe s'appelle alors le mode B-Skip.

Le partitionnement des macroblocs des images B est identiques à celui de images P présenté dans la partie estimation-compensation en mouvement. Il en est de même pour la prédiction des vecteurs de mouvement, ils sont prédits à partir du voisinage et seule la différence entre le vecteur réel et la prédiction est transmise. Il faut aussi transmettre l'indice des références utilisées dans chacune des deux listes.

La prédiction pondérée

H.264/AVC introduit aussi la prédiction pondérée ("Weighted Prediction") qui permet de mieux prédire les images dans les scènes de transitions (les fondus noirs : les luminances diminuent jusqu'à 0 et les chrominances jusqu'à 128 avant de repartir sur une nouvelle scène).

Cette prédiction pondérée peut aussi bien être appliquée à des P-slices qu'à des B-slices.
Pour une image B, on aura pour prédiction : $p = w1 \,.\, r1 + w2 \,.\, r2 + d$ où $r1$, $r2$ sont les références

utilisées avec $w1$, $w2$ leurs poids associés, et d est un offset.
Pour une image P, on aura la même chose mais sans la deuxième référence.

Donc si cette pondération n'est pas à utilisée, on prendra $(w1, w2, d) = (1/2, 1/2, 0)$ auquel cas on effectuera une moyenne des deux références (comme dans les standards précédents). Sinon il faut déterminer le triplet $(w1, w2, d)$, il existe pour cela deux méthodes. La première, explicite, utilise des coefficients prédéterminés que l'on transmet au décodeur, et la seconde, implicite, calcule ces coefficients à partir d'une distance temporelle entre les images utilisées en référence et l'image à coder.

Macrobloc Adaptative Frame/Field (MBAFF)

Des études ont montré que le codage en frame (mode progressif) est plus efficace sur les zones sans ou à faible mouvement, et que le codage en field (mode entrelacé) est plus efficace sur les zones à fort mouvement. H.264/AVC introduit donc ce mode MBAFF Macrobloc Adaptative Frame/Field qui lorsqu'il est actif permet de choisir le mode de codage.

Cette option MBAFF s'applique à une paire de macroblocs que l'on peut alors coder en frame ou en field. Dans le cas frame, on ne change rien, le codage est effectué comme si l'option MBAFF n'est pas utilisée. Par contre, dans le cas field, il faut modifier les macroblocs de telle sorte que le premier macrobloc contienne les lignes paires des deux macroblocs initiaux, il s'appelle alors le "top field", et que le second contienne toutes les lignes impaires, il s'appelle le "bottom field".

Lorsque cette option est utilisée, les macroblocs et leur voisinage ne sont pas forcément codés de la même manière (frame/field). C'est pourquoi les méthodes de :
- parcours zigzag
- prédiction des vecteurs de mouvement
- prédiction des modes de prédiction Intra
- prédiction spatiale Intra
- modélisation de contexte pour CABAC

sont modifiées pour être utilisables dans ce mode (par exemple, un macrobloc field ne peut pas utiliser son autre field pour la prédiction de mouvement).

La robustesse aux erreurs

Cette norme possède, de plus, plusieurs principes qui lui apportent une grande robustesse aux erreurs de transmission.

Flexible Macrobloc Ordering (FMO)

Lorsque cette option n'est pas utilisée, les images sont découpées en slices qui se succèdent, c'est-à-dire que les premiers macroblocs de l'image appartiennent au premier slice, les macroblocs suivants appartiennent au second slice, etc.

Lorsque l'option FMO [Wen03] est utilisée, des groupes de slices sont définis ainsi qu'une carte des groupes de slices. Chaque macrobloc doit alors posséder l'indice du groupe auquel il appartient pour pouvoir être replacé à la bonne place lors du décodage.

Ces groupes de slices peuvent être formés de plusieurs façons comme par exemple en slices entrelacés, en slices dispersés, en slices de premier plan et d'arrière plan, en damier,... (cf fig.

A.17)

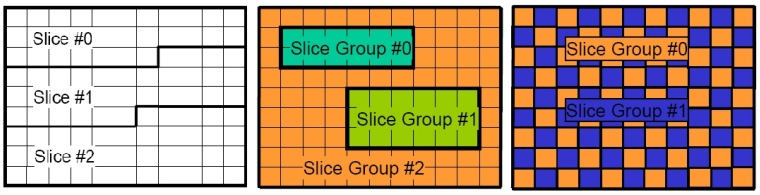


FIG. A.17 – Les slices sans FMO et deux exemples d'utilisation de FMO [WSBL03]

Cette option permet de transmettre les groupes de slices indépendamment évitant la perte de toute l'information si perte il y a.
L'option FMO peut être vue comme une méthode permettant de caractériser des régions d'intérêt.

Arbitrary Slice Ordering (ASO)

L'option ASO permet de transmettre et de recevoir les slices dans un ordre quelconque, les slices étant décodables indépendamment des autres. Elle s'applique notamment aux réseaux qui peuvent délivrer des paquets de données dépassés, c'est-à-dire des réseaux à mémoire qui peuvent transmettre un paquet en retard si celui-ci n'a pas déjà été transmis comme par exemple le réseau Internet.

Switched Slices SP/SI

La norme H.264/AVC introduit de nouveaux types d'images, les switched pictures SP et SI [KK03]. Ces images permettent notamment de pouvoir basculer d'un bitstream à un autre sans être obligé de passer par une image I, d'accéder aléatoirement dans la séquence et de faire des avances et retours rapides.

Si on considère deux bitstreams à différent débit binaire (Bitstream1 et Bitstream2) de la même séquence (cf fig. A.18), les images de chacun ne sont pas codées de la même manière : le partitionnement des macroblocs est différent, les références sont différentes,... Si on veut changer de bitstream au niveau n, et obtenir la séquence $\ldots P_{1,n-2}, P_{1,n-1}, P_{2,n}, P_{2,n+1} \ldots$, l'image $P_{2,n}$ ne sera pas bien décodée, car ces références passées ne sont pas disponibles. Elle introduira alors des artefacts visuels qui se propageront jusqu'à la prochaine image I à cause de la prédiction temporelle des vecteurs de mouvement.
On introduit donc dans les bitstreams des images SP $S_{1,n}$ et $S_{2,n}$ dites primaires. Chaque SP-slice primaire construit un SP-slice secondaire (ici $S_{12,n}$ pour passer du bitstream1 au 2) qui une fois décodé donne exactement le même résultat que le SP-slice primaire. Cependant, le SP-slice primaire utilise le bitstream auquel il appartient pour être décodé, alors que le secondaire utilise l'autre bitstream.
Dans l'exemple présenté (cf fig. A.18), $S_{12,n}$ sera décodé comme $S_{2,n}$, mais en utilisant $P_{1,n-1}$ en référence.

Dans la discussion précédente, on a considéré que les deux bitstreams représentaient la même séquence à différents débits, cependant il est aussi possible que ces bitstreams soient issus de différentes séquences comme par exemple la même scène filmée par plusieurs caméras. On parle alors de concaténation de bitstreams et non plus de basculement, le basculement étant un cas

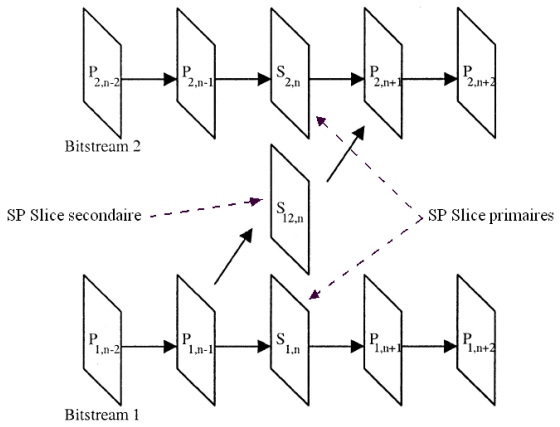


FIG. A.18 – Les SP-slices pour basculer d'un bitstream à un autre [KK03]

particulier de la concaténation.
Dans ce cas, les SP-slice secondaires ne peuvent pas être prédits à partir du second bitstream qui est différent (séquence différente). Pour cela, on utilise un SI-slice qui utilise une prédiction spatiale pour l'image secondaire plutôt qu'une prédiction temporelle (SP-slice). Cette image secondaire $SI_{12,n}$ sera donc décodée exactement comme l'image SP $S_{2,n}$ (cf fig. A.19 a)).
Ces SI-slice permettent donc de concaténer deux bitstreams différents et d'accéder aléatoirement à la séquence.

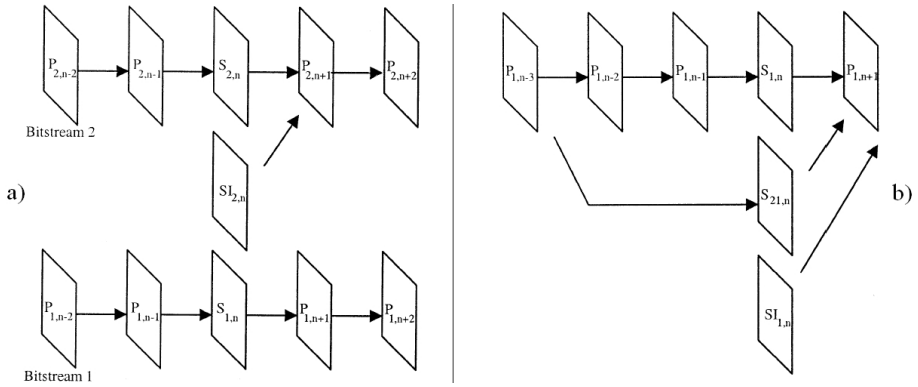


FIG. A.19 – a) SI-slices pour la concaténation ; b) SP-slices pour la récupération de données [KK03]

Ces images peuvent aussi servir pour la récupération de données. En effet, si une image est perdue lors de la transmission, le client perd une référence et ne peut plus décoder la suite sans introduire d'erreurs de prédiction. Il le signale donc au codeur qui lui transmet alors l'image secondaire $S_{21,n}$ de la prochaine image SP (cf fig. A.19 b)). Cette image secondaire utilise comme référence pour sa prédiction la dernière image correctement reçue par le client. Ce client peut alors décoder cette image correctement et repartir sur de bonnes références pour la suite.
Le codeur peut aussi décider d'envoyer une image SI $SI_{1,n}$ (cf fig. A.19 b)) à la place de l'image secondaire de la prochaine image SP si, par exemple, il y a eu trop d'images perdues et que la

référence utilisable est trop lointaine.

Les SP- et SI-slices permettent donc de concaténer deux bitstreams issus de la même séquence ou pas, et de récupérer des données perdues lors de la transmission et d'éviter la propagation d'erreurs.

Redundant slice

Cette option permet d'introduire une ou plusieurs représentations d'une image, d'une partie d'une image (slice) ou d'un macrobloc dans le même bitstream.

Cette représentation redondante peut être codée avec des paramètres différents. Par exemple, la représentation primaire peut être codée en utilisant un pas de quantification faible (bonne qualité), et la représentation secondaire avec un pas de quantification élevé (moins bonne qualité, mais peu de bits).

Le décodeur n'utilise que la représentation primaire si celle-ci est disponible, et ne tient pas compte de la représentation secondaire. Cependant, si la représentation primaire a été perdue lors de la transmission, alors la représentation secondaire est décodée.

Data partitioning

Le partitionnement des données est une technique déjà largement utilisée : H.263++ Annexe V et MPEG-4.

Cela consiste à réorganiser le train binaire par type de données plutôt que macrobloc par macrobloc (cf fig. A.8). Les données importantes telles que les données de mouvement ou les coefficients DC sont alors transmises avec une priorité plus forte que les autres comme les coefficients AC.

L'adaptation aux réseaux

Tout ce qui a été présenté jusqu'ici pour H.264/AVC est contenu dans une partie de la norme appelée la couche de codage vidéo VCL (Video Coding Layer). Cette norme définit en plus une autre couche appelée la couche d'abstraction réseau NAL (Network Abstraction Layer) [Wen03]. Cette nouvelle couche permet d'adapter la couche VCL à différents types de réseaux par souci de portabilité de cette norme sur tous types de réseaux (IP, MMS, H.32x pour wireless, MPEG-2 Systems pour le broadcasting,...)

Parameter Set

Ces ensembles de paramètres contiennent toutes les informations qui ne changent pas ou très peu pendant les phases de codage et de décodage. Leur utilisation permet d'éviter d'avoir à répéter certaines informations et donc de diminuer le débit. Il en existe deux types :

- Sequence Parameter Sets (SPS) qui contiennent toutes les informations relatives à une séquence d'images (le nombre d'images, leur format, leur fréquence,...)
- Picture Parameter Sets (PPS) qui contiennent toutes les informations relatives à tous les slices d'une image unique (type de prédiction, CABAC ou pas,...)

Chacun de ces ensembles de paramètres est identifié par un indice. L'indice du PPS est spécifié

dans chaque en-tête de slice (plutôt que les informations contenus dans l'ensemble de paramètres) et chaque PPS contient l'indice du SPS auquel il appartient.

Cependant, pour que l'utilisation de Parameter Sets soit efficace, il faut qu'ils soient transmis sans perte au décodeur. Il existe pour cela deux méthodes :

- La transmission "in-band" : tous les PS sont transmis en premier au décodeur par le même canal que le VCL, mais une bonne transmission doit être assurée. Ils sont donc tous transmis plusieurs fois pour augmenter la probabilité qu'au moins une copie arrive entière.
- La transmission "out-band" : les PS sont transmis par un autre canal, plus sûr que celui du VCL assurant ainsi, grâce à un protocole de contrôle, une transmission sans perte (cf fig. A.20).

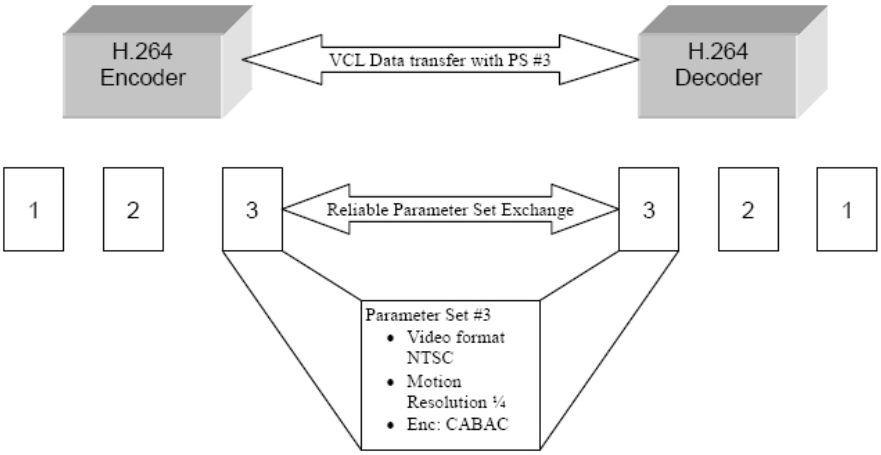


FIG. A.20 – Transmission "out-band" des Parameter Sets [TR03]

Les NAL Units et les Access Units

La couche d'abstraction réseau NAL est composée de NAL Units (NALU), eux-mêmes référencés par des Access Units.

Les NALU sont des paquets contenant un nombre entier d'octets, le premier étant une en-tête spécifiant les types de données qui suivent (dans les autres octets).

Ces données peuvent être entrelacées avec des octets préventifs permettant d'éviter la création de codes préfixes à l'intérieur des NALU. Ces codes préfixes sont utilisés dans certains protocoles réseaux (H.320, MPEG-2 System) pour pouvoir repérer les frontières des NALU. Ils sont insérés entre les NALU, d'où la nécessité d'éviter d'en créer dans les NALU.

Ces NALU peuvent être de deux types : les VCL NALU et les non-VCL NALU. Les VCL NALU contiennent les informations relatives aux images codées, et les non-VCL NALU des informations additionnelles telles que les Parameter Sets et des informations supplémentaires d'amélioration (des informations non nécessaires au décodage mais qui peuvent en améliorer la qualité).

Les Access Units permettent de référencer un ensemble de NALU, cet ensemble définissant une image décodable de la séquence.

Chaque Access Unit est défini selon la figure A.21.

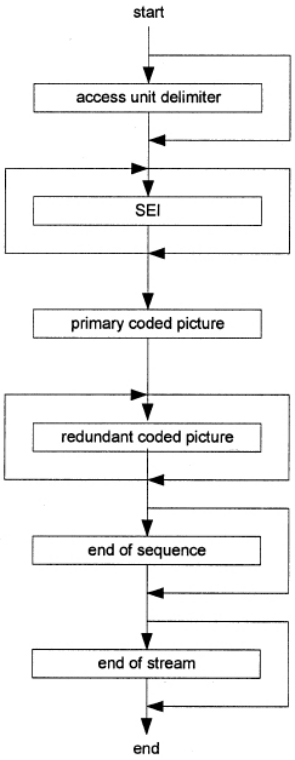


FIG. A.21 – Structure d'un Access Unit [JVT05] [WSBL03]

L'"access unit delimiter" est un NALU utilisé pour localiser le début de l'Access Unit, il peut être suivi d'informations supplémentaires d'amélioration SEI (Supplemental Enhancement Information). Chacune de ces informations SEI sont contenues dans un NALU qui comme pour l'"access unit delimiter" sont des non-VCL NALU.
La "primary coded picture" est un ensemble de VCL NALU décrivant la totalité d'une image décodable. Cette première représentation de l'image codée peut être suivie par d'autres VCL NALU, la "redundant coded picture" contenant des représentations redondantes de cette image ou de partie de cette image. Ces informations sont alors disponibles au niveau du décodeur qui peut les utiliser si des informations de la "primary coded picture" sont perdues.

Lorsque l'image représentée par cet Access Unit est la dernière d'une séquence (décodable indépendamment), un non-VCL NALU est ajouté : le "end of sequence".
De plus, si cette image est la dernière du bitstream, un autre non-VCL NALU est ajouté pour le signaler : le "end of stream".

Une séquence vidéo codée complète est donc représentée par un ensemble d'Access Units qui utilisent tous le même Sequence Parameter Set. Ces Access Units sont transmis séquentiellement dans le NAL stream (cf fig. A.22). Chaque séquence vidéo peut donc être décodée indépendamment des autres si on fournit les bonnes informations de Parameter Sets.
De plus, au début de chaque séquence, un IDR Access Unit (Instantaneous Decoding Refresh) est transmis. Il contient la première image Intra de la séquence qui sera l'unique référence utilisable pour décoder le début de cette nouvelle séquence.

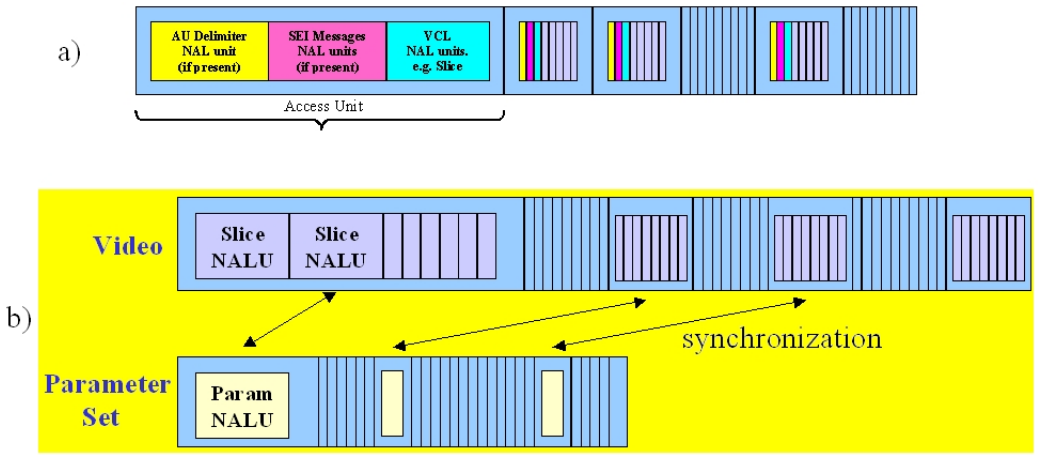


FIG. A.22 – Structure du NAL stream avec Parameter Set a) "Out-band" et b) "In-band"

A.3.9 Evaluation des performances de H.264/AVC par rapport à ces prédécesseurs

De nombreuses études ont été menées afin de montrer les performances de cette norme H.264 MPEG-4/AVC par rapport aux normes antérieures.

Nous allons ici présenter les résultats obtenus dans [WSJ+03] en utilisant :
- MPEG-2 en Main Level et Main Profile (ML@MP).
- H.263 avec le High Level Profile (HLP défini dans l'annexe X de H.263++) en utilisant 5 images de référence.
- MPEG-4 ASP (Advanced Simple Profile) avec une compensation en mouvement global et au 1/4 pixel.
- H.264 MPEG-4/AVC Main Profile avec CABAC et 5 images de référence mais pas les images B (pour être comparable à MPEG-2).

Toutes les séquences générées dans ces tests sont de la forme : IBBPBBP... (une image I unique et 2 images B insérées entre chaque image P). L'estimation du mouvement est effectuée dans une fenêtre de ±32 pixels pour toutes ces normes.

Les résultats (cf fig. A.23) peuvent être résumés sous la forme :

	Débit conservé par rapport à		
Codeur	MPEG-4 ASP	H.263 HLP	MPEG-2 ML@MP
H.264 MPEG-4/AVC	37.44%	47.58%	63.57%
MPEG-4 ASP	-	16.65%	42.95%
H.263 HLP	-	-	30.61%

TAB. A.17 – Moyennes des débits sauvés suivant les normes [WSJ+03]

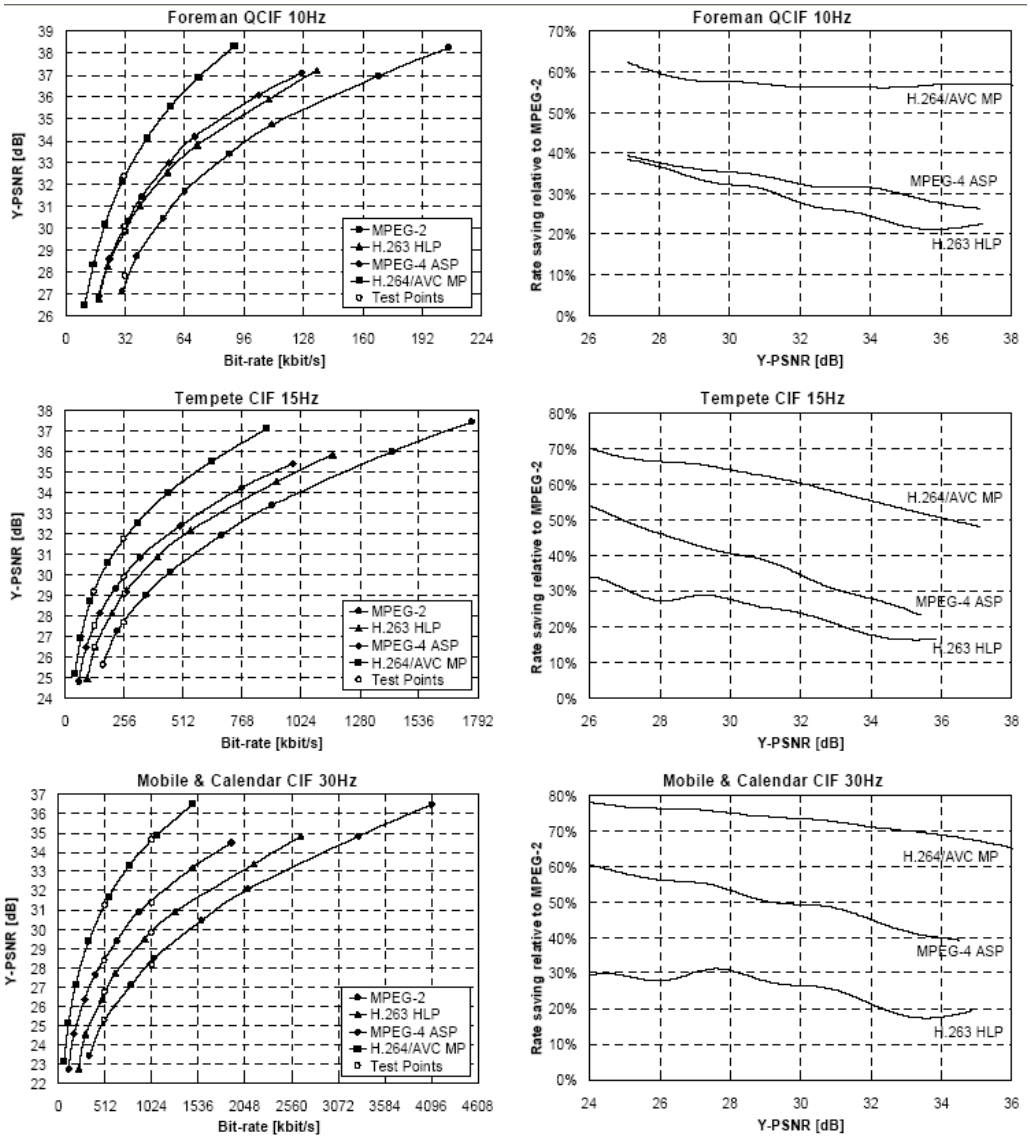


FIG. A.23 – Rapport signal à bruit PSNR en fonction du débit pour différentes normes [WSJ+03]

Bibliographie de l'auteur

[Bre06] A. Robert, I. Amonou. *« Procédé et dispositif de codage et de décodage d'une image, notamment d'une séquence d'images vidéo »*, brevet n° 06 51260.

[Bre07] A. Robert, I. Amonou. *« Adaptation du parcours des coefficients de la DCT dans un codeur hybride »*, brevet n° 07 06665.

[RAP06a] A. Robert, I. Amonou, B. Pesquet-Popescu. *« Improving DCT-based coders through Block Oriented Transforms »*. IEEE Advanced Concepts for Intelligent Vision Systems (ACIVS'06), Lecture Notes in Computer Science, vol.4179, p.375-383, septembre 2006. Anvers, Belgique.

[RAP06b] A. Robert, I. Amonou, B. Pesquet-Popescu. *« Improving Intra mode coding in H.264/AVC through Block Oriented Transforms »*. IEEE 8th International Workshop on Multimedia Signal Processing (MMSP'06), octobre 2006. Victoria, Canada.

[RAP06c] A. Robert, I. Amonou, B. Pesquet-Popescu. *« Amélioration de Codeurs DCT par Orientation des Blocs de la Transformée »*. COmpression et REprésentation des Signaux Audiovisuels (CORESA'06), novembre 2006. Caen, France.

[RAP07a] A. Robert, I. Amonou, B. Pesquet-Popescu. *« Amélioration du codage H.264 par Orientation des Blocs de la Transformée »*. COmpression et REprésentation des Signaux Audiovisuels (CORESA'07), novembre 2007. Montpelier, France.

[RAP07b] A. Robert, I. Amonou, B. Pesquet-Popescu. *« Amélioration du codage Intra H.264 par orientation des blocs de la transformée »*. GDR-ISIS Thème B - Image et Vision, mai 2007. Paris, France.

[RAP08a] A. Robert, I. Amonou, B. Pesquet-Popescu. *« Improving H.264 video coding through block oriented transforms »*. IEEE International Conference on Multimedia, & Expo (ICME'08), juin 2008. Hannover, Allemagne. Accepté.

Bibliographie

[Abh02] G.C.K. Abhayaratne. « *Lossless and Nearly Lossless Digital Video Coding* ». PhD thesis, University of Bath, United Kingdom, 2002.

[Abh03] G.C.K. Abhayaratne. « *N-Point Discrete Cosine Transforms that Map Integers to Integers for Lossless Image and Video Coding* ». Picture Coding Symposium (PCS'03), pages 417–422, avril 2003. Saint-Malo, France.

[ANR74] N. Ahmed, T. Natarajan and K.R. Rao. « *Discrete Cosine Transfom* ». IEEE Transactions on Computers, vol.32, n°1, p.90–93, janvier 1974.

[BG05] A.N. Belbachir and P.M. Goebel. « *The Contourlet Transform for Image Compression* ». Physics in Signal and Image Processing (PSIP'05), janvier 2005. Toulouse, France.

[Bjö01] G. Bjöntegaard. « *Calculation of Average PSNR Differences between RD-curves* ». VCEG Contribution VCEG-M33, avril 2001. Austin, Texas.

[BS92] R.H. Bamberger and J.T. Smith. « *A Filter Bank for the Directional Decomposition of Images : Theory and Design* ». IEEE Transactions on Signal Processing, vol.40, n°4, p.882–893, avril 1992.

[BVDE92] F.A.M.L. Bruekers and A.W.M. Van Den Enden. « *New Networks for Perfect Inversion and Perfect Reconstruction* ». IEEE Journal on Selected Areas in Communications, vol.10, n°1, p.130–137, janvier 1992.

[Cam04] N. Cammas. « *Codage Vidéo Scalable par Maillages et Ondelettes t+2D* ». PhD thesis, Université de Rennes 1, France, 2004.

[Can98] E.J. Candès. « *Ridgelets : Theory and Applications* ». PhD thesis, Standford University, United Kingdom, 1998. Department of Statistics.

[CD99] E.J. Candès and D.L. Donoho. « *Ridgelets : a key to higher-dimensional intermittency ?* ». Philosophical Transactions : Mathematical, Physical and Engineering Sciences, vol.357, n°1760, p.2495–2509, septembre 1999.

[CD00a] E. Candès and D. Donoho. « *Curvelets, Multiresolution Representation, and Scaling Laws* ». In *Wavelet Applications in Signal and Image Processing VIII*. Proc. SPIE 4119, 2000.

[CD00b] E.J. Candès and D.L. Donoho. « *Curvelets - A Surprisingly Effective Nonadaptive Representation For Objects with Edges* ». In *Curves and Surfaces*. L. L. Schumaker et al., 2000. Vanderbilt University Press, Nashville, Tennessee.

[CD02] E.J. Candès and D.L. Donoho. « *New Tight Frames of Curvelets and Optimal Representations of Objects with C^2 Singularities* ». Communications on Pure and Applied Mathematics, vol.57, n°2, p.219–266, novembre 2002.

[CDDY06] E.J. Candès, L. Demanet, D.L. Donoho and L. Ying. « *Fast Discrete Curvelet Transforms* ». SIAM Journal of Multiscale Modeling and Simulation, vol.5, n°3, p.861–899, septembre 2006.

[CDF92] A. Cohen, I. Daubechies and J.-C. Feauveau. « *Biorthogonal Bases of Compactly Supported Wavelets* ». Communications on Pure and Applied Mathematics, vol.45, n°5, p.485–560, mai 1992.

[CDSY98] R.C. Calderbank, I. Daubechies, W. Sweldens and B.-L. Yeo. « *Wavelet Transforms that Map Integers to Integers* ». Applied and Computational Harmonic Analysis, vol.5, n°3, p.332–369, juillet 1998.

[CEGK98] G. Coté, B. Erol, M. Gallant and F. Kossentini. « *H.263+ : Video Coding for Low Bit Rates* ». IEEE Transactions on Circuits and Systems for Video Technology, vol.8, n°7, p.849–866, novembre 1998.

[CON00] Y.J. Chen, S. Oruintura and T. Nguyen. « *Video Compression Using Integer DCT* ». IEEE International Conference on Image Processing (ICIP'00), vol.2, p.844–847, septembre 2000. Vancouver, Canada.

[COT+02] Y.J. Chen, S. Oraintara, T.D. Tran, K. Amaratunga and T.Q. Nguyen. « *Multiplierless Approximation of Transforms With Adder Constraint* ». IEEE Signal Processing Letters, vol.9, n°11, p.344–347, novembre 2002.

[CSF77] W.H. Chen, H.C. Smith and S.C. Fralick. « *A Fast Computational Algorithm for the Discrete Cosine Transform* ». IEEE Transactions on Communications, vol.25, n°9, p.1004–1009, septembre 1977.

[CXL01] L.Z. Cheng, H. Xu and Y. Luo. « *Integer Discrete Cosine Transform and its Fast Algorithm* ». IEEE Electronics Letters, vol.37, n°1, p.64–65, janvier 2001.

[CZL02] L.Z. Cheng, G.J. Zhong and J.S. Luo. « *New Family of Lapped Biorthogonal Transform via Lifting Steps* ». IEEE Proceedings on Vision, Image and Signal Processing, vol.149, n°2, p.91–96, avril 2002.

[Dau90] I. Daubechies. « *The Wavelet Transform, Time-Frequency Localization and Signal Analysis* ». IEEE Transactions on Information Theory, vol.36, n°5, p.961–1005, septembre 1990.

[DS98] I. Daubechies and W. Sweldens. « *Factoring Wavelet Transforms into Lifting Steps* ». Journal of Fourier Analysis and Applications, vol.4, n°3, p.247–269, 1998.

[DT03] W. Dai and T.D. Tran. « *Regularity-Constrained Pre- and Post-Filtering for Block DCT-Based Systems* ». IEEE Transactions on Signal Processing, vol.51, n°10, p.2568–2581, octobre 2003.

[DV00] M.N. Do and M. Vetterli. « *Orthonormal Finite Ridgelet Transform for Image Compression* ». IEEE International Conference on Image Processing (ICIP'00), vol.2, p.367–370, septembre 2000. Vancouver, Canada.

[DV01a] M.N. Do and M. Vetterli. « *Contourlets* », pages 1–27. Beyond Wavelets. G. V. Welland, 2001.

[DV01b] M.N. Do and M. Vetterli. « *Pyramidal Directional Filter Banks and Curvelets* ». IEEE International Conference on Image Processing (ICIP'01), vol.3, p.158–161, octobre 2001. Thessaloniki, Grèce.

[DV03] M.N. Do and M. Vetterli. « *The Finite Ridgelet Transform for Image Representation* ». IEEE Transactions on Image Processing, vol.12, n°1, p.16–28, janvier 2003.

[DV05] M.N. Do and M. Vetterli. « *The Contourlet Transform : An Efficient Directional Multiresolution Image Representation* ». IEEE Transactions on Image Processing, vol.14, n°12, p.2091–2106, décembre 2005.

[FG02] M. Flierl and B. Girod. « *AVC/H.264 Generalized B Pictures* ». Workshop and Exhibition on MPEG-4, juin 2002. San Jose, Californie.

[FG03] M. Flierl and B. Girod. « *Generalized B Pictures and the Draft H.264/AVC Video-Compression Standard* ». IEEE Transactions on Circuits and Systems for Video Technology, vol.13, n°7, p.587–597, juillet 2003.

[FKE07] A. Foi, V. Katkovnik and K. Egiazarian. « *Pointwise Shape-Adaptive DCT for High-Quality Denoising an Deblocking of Grayscale and Color Images* ». IEEE Transactions on Image Processing, vol.16, n°5, p.1395–1411, mai 2007.

[FWG98] M. Flierl, T. Wiegand and B. Girod. « *A Locally Optimal Design Algorithm for Block-Based Multi-Hypothesis Motion-Compensated Prediction* ». IEEE Data Compression Conference (DCC'98), pages 239–248, avril 1998. Snowbird, USA.

[FZ07a] J. Fu and B. Zeng. « *A Comparative Study of Compensation Techniques in Directional DCT's* ». IEEE International Symposium on Circuits and Systems (ISCAS'07), pages 521–524, mai 2007. New Orléans, USA.

[FZ07b] J. Fu and B. Zeng. « *Directional Discrete Cosine Transforms : A Theoretical Analysis* ». IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'07), vol.1, p.1105–1108, avril 2007. Honolulu, USA.

[GM84] A. Grossmann and J. Morlet. « *Decomposition of Hardy Functions into Square Integrable Wavelets of Constant Shape* ». SIAM Journal on Mathematics Analysis, vol.15, n°4, p.723–726, 1984.

[HM03] D.Y. Huang and R. Ma. « *Integer Fast Modified Cosine Transform* ». IEEE International Conference on Multimedia and Expo (ICME'03), vol.2, p.729–732, juillet 2003. Baltimore, Maryland.

[HW00] S.-T. Hsiang and J.W. Woods. « *Embedded Image Coding using Zeroblocks of Subband/Wavelet Coefficients and Context Modeling* ». IEEE International Symposium on Circuits and Systems (ISCS'00), vol.3, p.662–665, mai 2000. Genève, Suisse.

[ISO93] ISO/IEC 11172-2 Video. « *Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s* ». MPEG-1, 1993.

[ISO94a] ISO/IEC 10918. « *Digital Compression and Coding of Continuous-Tone Still Images* ». JPEG, 1994.

[ISO94b] ISO/IEC 13818-2 Video. « *Generic coding of moving pictures and associated audio information* ». MPEG-2, 1994.

[ISO00a] ISO/IEC 14496-2 Video. « *Generic coding of audio-visual objects* ». MPEG-4, 2000.

[ISO00b] ISO/IEC 15444. « *JPEG2000 Image Coding System* ». JPEG2000, 2000.

[ISO01a] ISO/IEC 15938 version 1. « *Multimedia content description interface* ». MPEG-7, 2001.

[ISO01b] ISO/IEC 21000. « *Multimedia framework* ». MPEG-21, 2001.

[ISO05] ISO/IEC 14496-5. « *Reference Software* ». MPEG-4, 2005.

[ITN02] M. Ikehara, T.D. Tran and T.Q. Nguyen. « *A Family of Lapped Regular Transforms With Integer Coefficients* ». IEEE Transactions on Signal Processing, vol.50, n°4, p.834–841, avril 2002.

[ITU90] ITU-T Recommendation H.261. « *Video codec for audio-visual services at px64kbit/s* », 1990.

[ITU94] ITU-T Recommendation H.262. « *Generic coding of moving pictures and associated audio information* », 1994.

[ITU95] ITU-T Recommendation H.263. « *Video coding for low bit rate communication* », 1995.

[ITU98] ITU-T Recommendation H.263 version 2. « *Video coding for low bit rate communication* », 1998.

[ITU01] ITU-T Recommendation H.263 version 3. « *Video coding for low bit rate communication* », 2001.

[ITU04] ITU-T Recommendation H.263 Annexe X. « *Video coding for low bit rate communication - Annex X : Profiles and levels definition* », mars 2004.

[jm06] « Joint Model 10 », 2006. http :\\ iphome.hhi.de\ suehring\ tml\ index.htm.

[JS94] B. Jawerth and W. Sweldens. « *An Overview of Wavelet Based Multiresolution Analysis* ». SIAM Review, vol.36, n°3, p.377–412, 1994.

[JVT04] JVT - ISO/IEC 14496-10 AVC - ITU-T Recommendation H.264 Amendment 1. « *Advanced Video Coding Amendment 1 : Fidelity Range Extensions* ». Draft Text of H.264/AVC Fidelity Range Extensions Amendment, juillet 2004.

[JVT05] JVT - ISO/IEC 14496-10 AVC - ITU-T Recommendation H.264. « *Advanced video coding for generic audio-visual services* ». ITU-T Recommendation and Final International Standard, JVT-G050, 2005.

[JVT06] JVT - ISO/IEC 14496-10 AVC - ITU-T Recommendation H.264 Amendment 3. « *Advanced Video Coding Amendment 3 : Scalable Video Coding* ». Final Draft International Standard of H.264/AVC Scalable Video Coding Amendment, juillet 2006.

[KK03] M. Karczewicz and R. Kurceren. « *The SP- and SI-Frames Design for H.264/AVC* ». IEEE Transactions on Circuits and Systems for Video Technology, vol.13, n°7, p.637–644, juillet 2003.

[KS98] P. Kauff and K. Schuur. « *Shape-Adaptive DCT with Block-based DC Separation and ΔDC Correction* ». IEEE Transactions on Circuits and Systems for Video Technology, vol.8, p.237–242, juin 1998.

[kta07] « VCEG - KTA reference model », 2007. http :\\ www.tnt.unihannover.de\ vatis\ kta\ jm11.0kta1.2.zip.

[LF03] F. Loras and J. Fournier. « *H.264/MPEG-4 AVC, un nouveau standard de compression vidéo* ». Conference on COmpression et REprésentation des Signaux Audiovisuels (CORESA'03), janvier 2003. Lyon, France.

[LPM00] E. Le Pennec and S. Mallat. « *Image Compression with Geometrical Wavelets* ». IEEE International Conference on Image Processing (ICIP'00), vol.1, p.661–664, septembre 2000. Vancouver, Canada.

[LPM01] E. Le Pennec and S. Mallat. « *Représentation d'Image par Bandelettes et Application à la Compression* ». 18e Colloque du Groupe de Recherche et d'Etudes du Traitement du Signal et des Images (GRETSI'01), pages 56–59, septembre 2001. Toulouse, France.

[LPM03] E. Le Pennec and S. Mallat. « *Bandelettes et Représentation Géométrique des Images* ». 19e Colloque du Groupe de Recherche et d'Etudes du Traitement du Signal et des Images (GRETSI'03), pages 376–379, septembre 2003. Paris, France.

[LPM04] E. Le Pennec and S. Mallat. « *Bandelet Image Approximation and Compression* ». SIAM Journal of Multiscale Modeling and Simulation, vol.4, n°3, p.992–1039, novembre 2004.

[LPM05] E. Le Pennec and S. Mallat. « *Sparse Geometric Image Representations With Bandelets* ». IEEE Transactions on Image Processing, vol.14, n°4, p.423–438, avril 2005.

[LT00] J. Liang and T.D. Tran. « *Approximating the DCT with the Lifting Scheme : Systematic Design and Applications* ». 34th IEEE Asilomar Conference on Signals, Systems and Computers, vol.1, p.192–196, novembre 2000. Pacific Grove, Californie.

[LT01] J. Liang and T.D. Tran. « *Fast Multiplierless Approximations of the DCT With the Lifting Scheme* ». IEEE Transactions on Signals Processing, vol.49, n°12, p.3032–3044, décembre 2001.

[LTT01] J. Liang, C. Tu and T.D. Tran. « *Fast Lapped Transforms via Time Domain Pre- and Post-Processing* ». IEEE International Conference on Information, Communications, and Signal Processing (ICICS'01), octobre 2001. Singapour.

[Mal89a] S. Mallat. « *Multiresolution Approximations and Wavelet Orthogonal Bases of $L^2(\mathbb{R})$* ». Transaction of the American Mathematical Society, vol.315, p.69–87, septembre 1989.

[Mal89b] S. Mallat. « *A Theory for Multiresolution Signal Decomposition : The Wavelet Representation* ». IEEE Transactions on Pattern Analysis and Machine Intelligence, vol.2, n°7, p.674–693, juillet 1989.

[Mal89c] Henrique S. Malvar. « *The LOT : Transform Coding Without Blocking Effects* ». IEEE Transactions on Acoustics, Speech, and Signal Processing, vol.37, n°4, p.553–559, avril 1989.

[Mal90] Henrique S. Malvar. « *Lapped Transforms for Efficient Transform/Subband Coding* ». IEEE Transactions on Acoustics, Speech, and Signal Processing, vol.38, n°6, p.969–978, juin 1990.

[Mal92] Henrique S. Malvar. « *Signal Processing With Lapped Transforms* ». Norwood MA Artech House, 1992.

[Mal98] Henrique S. Malvar. « *Biorthogonal and Nonuniform Lapped Transforms for Transform Coding with Reduced Blocking and Ringing Artifacts* ». IEEE Transactions on Signal Processing, vol.46, n°4, p.1043–1053, avril 1998.

[Mal00a] S. Mallat. « *Une Exploration des Signaux en Ondelettes* ». Ellipses diffusion. Editions de l'Ecole Polytechnique, 2000.

[Mal00b] Henrique S. Malvar. « *Fast Progressive Image Coding without Wavelets* ». IEEE Data Compression Conference (DCC'00), pages 243–252, mars 2000. Snowbird, Utah.

[Mey90] Y. Meyer. « *Ondelettes et Opérateurs* ». Tome 1. Hermann, Paris 1990.

[MHKK03] H.S. Malvar, A. Hallapuro, M. Karczewicz and L. Kerofsky. « *Low-Complexity Transform and Quantization in H.264/AVC* ». IEEE Transactions on Circuits and Systems for Video Technology, vol.13, n°7, p.598–603, juillet 2003.

[MSB+02] D. Marpe, H. Schwarz, G. Blättermann, G. Heising and T. Wiegand. « *Context-based Adaptive Binary Arithmetic Coding in JVT / H.264* ». IEEE International Conference on Image Processing (ICIP'02), septembre 2002. Rochester, New York.

[MSW03] D. Marpe, H. Schwarz and T. Wiegand. « *Context-based Adaptive Binary Arithmetic Coding in the H.264/AVC Video Compression Standard* ». IEEE Transactions on Circuits and Systems for Video Technology, vol.13, n°7, p.620–636, juillet 2003.

[MW03] D. Marpe and T. Wiegand. « *A Highly Efficient Multiplication-Free Binary Arithmetic Coder and its Application in Video Coding* ». IEEE International Conference on Image Processing (ICIP'03), septembre 2003. Barcelone, Espagne.

[NM06] M. Narroschke and H.G. Musmann. « *Adaptive Prediction Error Coding in Spatial and Frequency Domain with a fixed Scan in the Spatial Domain* ». ITU-T Q.6/SG16, VCEG-AD07, octobre 2006. Hangzhou, Chine.

[PM05a] G. Peyré and S. Mallat. « *Discrete Bandelets with Geometric Orthogonal Filters* ». IEEE International Conference on Image Processing (ICIP'05), vol.1, p.65–68, septembre 2005. Gênes, Italie.

[PM05b] G. Peyré and S. Mallat. « *Surface Compression with Geometric Bandelets* ». ACM Transactions on Graphics, vol.24, n°3, p.601–608, juillet 2005.

[PT03a] G. Plonka and M. Tasche. « *Integer DCT-II by lifting steps* », volume 145of*International Series of Numerical Mathematics (W. Haussmann, K. Jetter, M. Reimer, J. Stöckler (edition))*, pages 235–252. Birkhäuser, 2003.

[PT03b] G. Plonka and M. Tasche. « *Invertible Integer DCT Algorithms* ». Applied and Computational Harmonic Analysis, vol.15, n°1, p.70–88, juillet 2003.

[Ric02a] Iain E.G. Richardson. « *Context-based Adaptive Binary Arithmetic Coding (CABAC)* ». H.264 / MPEG-4 Part 10 White Paper, octobre 2002.

[Ric02b] Iain E.G. Richardson. « *Overview of H.264* ». H.264 / MPEG-4 Part 10 White Paper, octobre 2002.

[Ric02c] Iain E.G. Richardson. « *Variable-Length Coding* ». H.264 / MPEG-4 Part 10 White Paper, octobre 2002.

[Ric03a] Iain E.G. Richardson. « *Prediction of Inter Macroblocks in P-slices* ». H.264 / MPEG-4 Part 10 White Paper, avril 2003.

[Ric03b] Iain E.G. Richardson. « *Prediction of Intra Macroblocks* ». H.264 / MPEG-4 Part 10 White Paper, avril 2003.

[Ric03c] Iain E.G. Richardson. « *Reconstruction Filter* ». H.264 / MPEG-4 Part 10 White Paper, avril 2003.

[Ric03d] Iain E.G. Richardson. « *Transform and Quantization* ». H.264 / MPEG-4 Part 10 White Paper, mars 2003.

[Rij96] Karel Rijkse. « *H.263 : Video Coding for Low-Bit-Rate Communication* ». IEEE Communications magazine, pages 42–45, décembre 1996.

[RSC01] M. Rabbani and D. Santa Cruz. « *The JPEG2000 Still-Image Compression Standard* ». IEEE International Conference on Image Processing (ICIP'01), octobre 2001. Thessaloniki, Grèce.

[RY90] K.R. Rao and P. Yip. « *Discrete Cosine Transform : Algorithms, Advantages, Applications* ». Academic Press Professional, Inc., 1990. San Diego, USA.

[SCD02] J.L. Starck, E.J. Candès and D.L. Donoho. « *The Curvelet Transform for Image Denoising* ». IEEE Transactions on Image Processing, vol.11, n°6, p.670–684, juin 2002.

[SFY06] V.P. Shah, J.E. Fowler and N.H. Younan. « *Tarp Filtering of Block-Transform Coefficients for Embedded Image Coding* ». IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'06), vol.2, mai 2006. Toulouse, France.

[Sik97] Thomas Sikora. « *MPEG digital video-coding standards* ». IEEE Signal Processing Magazine, vol.14, n°5, p.82–100, septembre 1997.

[SM95] T. Sikora and B. Makai. « *Shape-Adaptive DCT for Generic Coding of Video* ». IEEE Transactions on Circuits and Systems for Video Technology, vol.5, n°1, p.59–62, février 1995.

[SMW04] H. Schwarz, D. Marpe and T. Wiegand. « *MCTF and Scalability Extension of H.264/AVC* ». Picture Coding Symposium (PCS'04), décembre 2004. San Francisco, USA.

[SMW06] H. Schwarz, D. Marpe and T. Wiegand. « *Overview of the Scalable H.264/MPEG4-AVC Extension* ». IEEE International Conference on Image Processing (ICIP'06), pages 161–164, octobre 2006. Atlanta, GA, USA.

[SMW07] H. Schwarz, D. Marpe and T. Wiegand. « *Overview of the Scalable Video Coding Extension of the H.264/AVC Standard* ». IEEE Transactions on Circuits and Systems for Video Technology, vol.17, n°9, p.1103–1120, septembre 2007.

[SS96] W. Sweldens and P. Schröder. « *Building Your Own Wavelets at Home* ». In *Wavelets in Computer Graphics*, pages 15–87. ACM SIGGRAPH Course notes, 1996.

[Str99] G. Strang. « *The Discrete Cosine Transform* ». SIAM Review, vol.41, n°1, p.135–147, 1999.

[Swe96] W. Sweldens. « *The Lifting Scheme : a Custom-Design Construction of Biorthogonal Wavelets* ». Applied and Computational Harmonic Analysis, vol.3, n°2, p.186–200, 1996.

[Swe97a] W. Sweldens. « *The Lifting Scheme : a Construction of Second Generation Wavelets* ». SIAM Journal on Mathematics Analysis, vol.29, n°2, p.511–546, 1997.

[Swe97b] W. Sweldens. « *Wavelets and the Lifting Scheme : a 5 Minutes Tour* ». Journal of Applied Mathematics and Mechanics (ZAMM), vol.76, n°2, p.41–44, 1997.

[SWS03] R. Schäfer, T. Wiegand and H. Schwarz. « *The Emerging H.264/AVC Standard* ». EBU Technical Review, janvier 2003.

[TLT03] T.D. Tran, J. Liang and C. Tu. « *Lapped Transform via Time-Domain Pre- and Post-Filtering* ». IEEE Transactions on Signal Processing, vol.54, n°6, p.1557–1571, juin 2003.

[TR03] A. Tamhankar and K.R. Rao. « *An Overview of H.264 / MPEG-4 Part 10* ». 4th EURASIP Conferences, juillet 2003. Zagreb, Croatie.

[Tra99] Trac D. Tran. « *A Fast Multiplierless Block Transform for Image and Video Compression* ». IEEE International Conference on Image Processing (ICIP'99), vol.3, p.822–826, octobre 1999. Kobe, Japon.

[Tra00a] Trac D. Tran. « *The BinDCT : Fast Multiplierless Approximation of the DCT* ». IEEE Signals Processing Letters, vol.7, n°6, p.141–144, juin 2000.

[Tra00b] Trac D. Tran. « *The LiftLT : Fast-Lapped Transforms via Lifting Steps* ». IEEE Transactions on Signal Processing Letters, vol.7, n°6, p.145–148, juin 2000.

[TT01] T.D. Tran and C. Tu. « *Lapped Transform Based Video Coding* ». Proceedings of SPIE Applications of Digital Image Processing XXIV, vol.4472, p.319–333, août 2001. San Diego, USA.

[TZ94] D. Taubman and A. Zakhor. « *Orientation Adaptive Subband Coding of Images* ». IEEE Transactions on Image Processing, vol.3, n°4, p.421–437, juillet 1994.

[UTY95] M. Unser, P. Thévenaz and L.P. Yaroslavsky. « *Convolution-Based Interpolation for Fast, High-Quality Rotation of Images* ». IEEE Transactions on Image Processing, vol.4, n°10, p.1371–1381, octobre 1995.

[VBLV07] V. Velisavljević, B. Beferull-Lozano and M. Vetterli. « *Space-Frequency Quantization for Image Compression with Directionlets* ». IEEE Transactions on Image Processing, vol.16, n°7, p.1761–1773, juillet 2007.

[VBLVD05] V. Velisavljević, B. Beferull-Lozano, M. Vetterli and P.L. Dragotti. « *Approximation Power of Directionlets* ». IEEE International Conference on Image Processing (ICIP'05), vol.1, p.741–744, septembre 2005. Gênes, Italie.

[VBLVD06] V. Velisavljević, B. Beferull-Lozano, M. Vetterli and P.L. Dragotti. « *Directionlets : Anisotropic Multi-Directional Representation with Separable Filtering* ». IEEE Transactions on Image Processing, vol.15, n°7, p.1916–1933, juillet 2006.

[VBLVD07] V. Velisavljević, B. Beferull-Lozano, M. Vetterli and P.L. Dragotti. « *Image Representation and Compression using Directionlets* ». SPIE Photonics & Optics *(papier invité)*, août 2007. San Diego, Californie.

[Vei00] Daniel Veiner. « *EE368b Final Report on Performance of the Lapped Orthogonal Transform* ». http :\\www.stanford.edu\class\ee368b\Projects\dveiner\index.html, novembre 2000.

[Wen03] Stephan Wenger. « *H.264/AVC over IP* ». IEEE Transactions on Circuits and Systems for Video Technology, vol.13, n°7, p.645–656, juillet 2003.

[WHS01] C. Wei, P. Hao and Q. Shi. « *Integer DCT-based Image Coding* ». Picture Coding Symposium (PCS'01), pages 175–178, avril 2001. Seoul, Korea.

[Wie03] Mathias Wien. « *Variable Block-Size Transforms for H.264/AVC* ». IEEE Transactions on Circuits and Systems for Video Technology, vol.13, n°7, p.604–613, juillet 2003.

[WSBL03] T. Wiegand, G.J. Sullivan, G. Bjontegaard and A. Luthra. « *Overview of the H.264/AVC Video Coding Standard* ». IEEE Transactions on Circuits and Systems for Video Technology, vol.13, n°7, p.560–576, juillet 2003.

[WSJ+03] T. Wiegand, H. Schwarz, A. Joch, F. Kossentini and G.J. Sullivan. « *Rate-Constrained Coder Control and Comparison of Video Coding Standards* ». IEEE Transactions on Circuits and Systems for Video Technology, vol.13, n°7, p.688–703, juillet 2003.

[WZVS06] D. Wang, L. Zhang, A. Vincent and F. Speranza. « *Curved Wavelet Transform for Image Coding* ». IEEE Transactions on Image Processing, vol.15, n°8, p.2413–2421, août 2006.

[XXW07] H. Xu, J. Xu and F. Wu. « *Lifting-Based Directional DCT-like Transform for Image Coding* ». IEEE Transactions on Circuits and Systems for Video Technology, vol.17, n°10, p.1325–1335, octobre 2007.

[ZCBK01] Y. Zeng, L. Cheng, G. Bi and A.C. Kot. « *Integer DCTs and Fast Algorithms* ». IEEE Transactions on Signal Processing, vol.49, n°11, p.2774–2782, novembre 2001.

[ZCC01] G. Zhong, L. Cheng and H. Chen. « *Integer Lapped Biothogonal Transform* ». IEEE International Conference on Image Processing (ICIP'01), vol.2, p.471–474, octobre 2001. Thessaloniki, Grèce.

[ZF06] B. Zeng and J. Fu. « *Directional Discrete Cosine Transforms for Image Coding* ». IEEE International Conference on Multimedia and Expo (ICME'06), pages 721–724, juillet 2006. Toronto, Canada.

Abstract

Hybrid video coding scheme

This thesis deals with improving state of the art video coders like H.264 MPEG-4/AVC by using structural informations contained in the coded images.

In particular, it addresses the transform step used in all image and video coders. The transform, allowing to mathematically decorrelate the information in order to decrease its entropy coding cost, is a key point of video compression. The basis functions of most transforms are built in the horizontal and vertical directions. This results, in the presence of a contour or edge, in high frequency components that are not desirable for an efficient compression and that bring unpleasant artifacts (like ringing) when heavily quantized at low bitrates. As most natural images, either predicted or not shows strong geometrical structures, few of them (DCT, wavelets, lapped, DCT by lifting scheme...) permit to represent efficiently the geometrical structures of images.
State of the art of transforms exploiting geometrical structures are historically linked to second generation wavelets like contourlets, bandelets or directionlets. However, more recently, recent researches focused on the block-based DCT, trying to catch the orientation to effectively represent these geometrical structures.

The aim of our study is to improve the coding stage of H.264/AVC residual images coming from spatial (Intra) or temporal (Inter) predictions by using geometrical structures present in the blocks.

The first approach of this thesis work tries to find a unified approach between these works on DCT and wavelets.
To this end, we have defined a lifting scheme that realizes the operations of the DCT of H.264/AVC. This scheme allows to see this DCT as a wavelet transform. Second generation tools can be applied on it in order to improve its representation of geometrical structures.
A version of lapped transform in pre- and post-processing has been introduce in H.264/AVC.

The second approach of our researches has been to define a new method that can be seen as a pre-processing stage prior to the effective transform stage associated with an adapted scan of the quantized coefficients.
The pre-processing stage of this method realizes pseudo-rotations, straightening blocks of the image to horizontal or vertical axes. The rotations are realized by shearing that can be viewed as circular shifts of the pixels, thus avoiding the defaults inherent to matrix based rotations. This method presents interesting compression results. However the high coding cost of the orientation information selected through a rate-distortion criterion deteriorates the overall performance of the encoder in low bitrates. This method remains more efficient than H.264/AVC in high bitrates ($QP < 30$).
Quantized coefficients coming from the oriented transform are then scanned according to vertical, horizontal or zigzag patterns depending on the applied pseudo-rotation or the partition type. This adapted scan allows to improve the overall efficiency of the method a few rate and so ameliorate our global oriented method that becomes more efficient than H.264/AVC in medium bitrates ($QP < 35$).

Keywords : Orientation, transforms, contours, coefficients scan, H.264/AVC, wavelets.

Résumé

Cette thèse s'intéresse à améliorer les codeurs vidéo actuels tels que H.264 MPEG-4/AVC en utilisant avantageusement des informations structurelles contenues dans les images codées.

Dans ce contexte, on observe que tous codeurs vidéo utilisent une étape de transformation permettant de décorréler mathématiquement les informations traitées afin d'en diminuer le coût de codage entropique. D'autre part, on remarque que toutes les images traitées qu'elles soient prédites ou non, possèdent des structures géométriques très marquées.
Une étude des transformées existantes et possibles pour ces codeurs vidéo montre que peu d'entre elles (DCT, en ondelettes, à recouvrement, DCT sous forme lifting...) permettent de représenter efficacement ces structures géométriques des images.
L'état de l'art de ces transformées exploitant les structures géométriques est porté historiquement par les ondelettes de seconde génération comme les *contourlets*, les bandelettes ou les *directionlets*. Mais, plusieurs études plus récentes utilisent des approches DCT, basées blocs, avec des orientations afin de mieux représenter ces structures géométriques.

L'objectif de notre étude est d'améliorer le codage des images résiduelles H.264/AVC, issues de prédictions spatiales (Intra) ou temporelles (Inter), en utilisant leurs structures géométriques.

Une première approche de ce travail de thèse nous a conduits à analyser et exploiter des méthodes connues de l'état de l'art.
Pour cela, nous avons défini un schéma sous forme lifting réalisant les opérations de la DCT H.264/AVC. Ce schéma permet de voir cette DCT comme une transformée en ondelettes et donc de disposer d'une approche commune. On peut alors lui appliquer des outils de seconde génération afin qu'elle représente au mieux les structures géométriques des images.
Et, une version de transformée à recouvrement en pré- et post-traitements a été utilisée dans le codeur H.264/AVC.

Une seconde approche de nos recherches a été de définir une méthode d'orientation par pré- et post-traitements associée à un parcours adapté des coefficients quantifiés produits.
Le pré-traitement de cette méthode d'orientation réalise des pseudo-rotations permettant de redresser les blocs des images vers l'horizontale ou la verticale. Cette opération est réalisée par cisaillements, soit par permutations circulaires des pixels, améliorant la décorrélation de la DCT qui suit sans présenter les défauts inhérents aux approches de l'état de l'art. Cette méthode, insérée dans un codeur H.264/AVC, présente de bonnes performances de codage. Cependant, le coût des informations d'orientation, sélectionné selon un critère débit-distorsion, est élevé dégradant ces performances dans les bas débits, la méthode restant plus efficace que H.264/AVC dans les hauts débits ($QP < 30$).
Les coefficients quantifiés issus de la méthode d'orientation précédente sont ensuite parcourus à la verticale, à l'horizontale ou en zigzag suivant les redressements appliqués ou le type de partitions. Cette adaptation de parcours permet de légèrement conserver du débit améliorant ainsi notre méthode globale qui devient plus efficaces que H.264/AVC dans les moyens débits ($QP < 35$).

Mots clefs : Orientation, transformées, contours, parcours des coefficients, H.264/AVC, ondelettes.