Amélioration de codeurs DCT par orientation des blocs de la transformée

ROBERT Antoine[1] AMONOU Isabelle[1] PESQUET-POPESCU Béatrice[2]

[1] France Telecom R&D
4, rue du Clos Courtel
35512 Cesson-Sévigné Cedex
{a.robert, isabelle.amonou}@francetelecom.com

[2] TSI - ENST Paris
46, rue Barrault
75634 Paris Cedex 13
pesquet@tsi.enst.fr

Résumé

Cet article décrit un pré-traitement pour des codeurs de type DCT, et plus généralement pour des codeurs d'images ou de vidéo basés blocs utilisant avantageusement l'orientation de ces blocs. Contrairement à la plupart des solutions proposées jusqu'alors, ce n'est pas ici la transformée qui s'adapte au signal, mais le signal qui est traité pour s'adapter à la transformée. Les blocs sont orientés grâce à des permutations circulaires appliquées au niveau pixel. Avant de réaliser ces permutations, l'orientation de chaque bloc est évaluée à l'aide d'une sélection basée sur un critère débit-distorsion. Ce pré-traitement introduit dans un codeur AVC [1] et appliqué aux images résiduelles intra permet d'en améliorer les performances.

Mots clefs

Transformation directionnelle, orientation, permutations circulaires, codeurs DCT, H.264-MPEG4/AVC.

1 Introduction

La compression d'images et de séquences vidéo est motivée par la recherche de représentations compactes pour les données en utilisant des transformées. Les premières transformées introduites étaient séparables et simples comme la DCT et les ondelettes de première génération. Par simple, on entend que ces transformations ne sont pas optimales pour représenter les données images de manière compacte et qu'elles sont souvent redondantes, mais elles sont toutefois généralement rapides et peu complexes. Cette sous-optimalité est partiellement due au fait que ces transformées ne sont pas adaptées aux données possédant des discontinuités situées le long de courbes régulières. Afin de profiter des structures géométriques des images ou des séquences vidéo, plusieurs auteurs ont proposé de nouvelles transformations telles que les curvelets, les contourlets, les bandelettes ou les directionlets, et d'autres ont cherché à améliorer les transformations existantes en tenant compte de ces structures géométriques.

Les curvelets, introduites par Candès et Donoho [2], permettent grâce à leur grand degré de directionnalité d'obtenir une approximation optimale des images lisses possédant des contours C^2. Cette transformée nécessite une rotation et correspond à un partitionnement 2D des fréquences basé sur les coordonnées polaires, ce qui est équivalent à un banc de filtres directionnels. Elles ont été définies initialement pour le cas continu et ne sont pas trivialement transférables au cas discret. Pour outrepasser ce problème, Do et Vetterli ont proposé les contourlets [3] qui ont les mêmes caractéristiques géométriques que les curvelets, mais définies directement dans le cas discret. Cette transformation effectue une analyse directionnelle d'un signal 2D en utilisant un banc pyramidal de filtres directionnels. Pour cela, le signal subit d'abord une décomposition Laplacienne redondante et pyramidale avant que chacune des sous-bandes obtenues ne soit traitée par le banc de filtres directionnels. Ces méthodes d'analyse impliquent que les curvelets et les contourlets sont redondantes, de plus ces transformées ne sont pas efficaces à haut débit.

Mallat a proposé les bandelettes [4] et plus récemment les bandelettes de seconde génération [5] qui sont toutes deux des transformées adaptatives. Dans le cas de seconde génération, une transformée géométrique orthogonale est appliquée aux coefficients en ondelettes, c'est à dire qu'après avoir décomposé les données grâce à un banc de filtres d'ondelettes orthogonales, les coefficients sont traités à l'aide de filtres directionnels orthogonaux. Chacune des directions géométriques correspond à une transformée différente (un filtre directionnel différent). Ceci nécessite alors une détection des contours et une déformation des données afin de pouvoir appliquer la bonne transformée sur le bon treillis.

Plus récemment, Velisavljević et Vetterli ont introduit les directionlets [6]. Elles travaillent à échantillonnage critique en appliquant un filtrage séparable non seulement à l'horizontale et à la verticale mais aussi suivant des sous-ensembles de "co-lignes" numériques. Ces co-lignes numériques représentent toutes les directions définies sur un treillis entier. Le filtrage le long de ces co-lignes est réalisé en effectuant une rotation définie par la pente de ces co-lignes avant d'appliquer un filtrage horizontal. Cependant, ces directionlets ne sont pas des transformées basées blocs.

D'autres méthodes utilisent des rotations complètes de l'image ou d'une partie de l'image, mais elles nécessitent généralement une interpolation. Unser et al. [7] ont conçu des algorithmes rapides de rotation d'images permettant de conserver la qualité initiale de cette image. Ces rotations

sont décomposées en trois translations s'appuyant sur une interpolation basée sur une convolution. Le principal désavantage de ces méthodes rotationnelles est que les informations contenues dans les coins des blocs ou de l'image sont généralement perdues.

Peu de méthodes présentées précédemment utilisent des transformées basées blocs qui sont pourtant les plus courantes dans les standards actuels (e.g. MPEGx, H.26x, JPEG, ...). Ceci vient du fait que les blocs introduisent de nouvelles discontinuités à cause de leurs bords. Notre but est donc de construire une rotation basée bloc qui conserve la forme (rectangulaire) de ce bloc afin de pouvoir y appliquer une transformée basée bloc.

Cet article est organisé comme suit : dans la Section 2 nous introduisons notre pré-traitement et décrivons comment il s'applique aux blocs. Dans la Section 3, nous présentons la méthode de sélection d'orientation avant de décrire le codage de ces informations dans la Section 4. Enfin, quelques résultats numériques de notre pré-traitement appliqué aux images résiduelles intra de AVC sont présentés en Section 5 avant de conclure et de donner quelques perspectives dans la Section 6.

2 Orientation des blocs de la transformée

Toutes les transformées présentées précédemment essaient de s'adapter au signal, mais chacune d'entre elles nécessite pour cela des opérations non entières en contradiction avec la reconstruction parfaite. Les contourlets et les curvelets sont redondantes, les bandelettes nécessitent une détection de contours et une déformation des données, et les rotations une interpolation. La méthode que nous proposons ici est un pré-traitement des images ou des séquences vidéo qui tient compte des structures géométriques des données sans déformation ni interpolation du signal.

La plus connue des transformées basées bloc est la DCT (Discrete Cosine Transform) qui est utilisée dans la plupart des standards images et vidéo tels que JPEG [8], MPEGx, H.26x comme H.264-MPEG4/AVC [1]. Cette transformée s'applique à des blocs de coefficients qui peuvent avoir une taille variable : soit 8x8 dans le cas de la DCT flottante de JPEG, soit 4x4 et 8x8 pour les DCT entières de AVC. De plus, les coefficients de ces blocs peuvent être de différentes natures : soit directement issus des images (données brutes) comme dans le cas JPEG, soit issus de prédiction comme dans le cas AVC. Dans tous ces cas, les blocs de coefficients à traiter présentent des motifs réguliers orientés comme représenté sur la Figure 1.

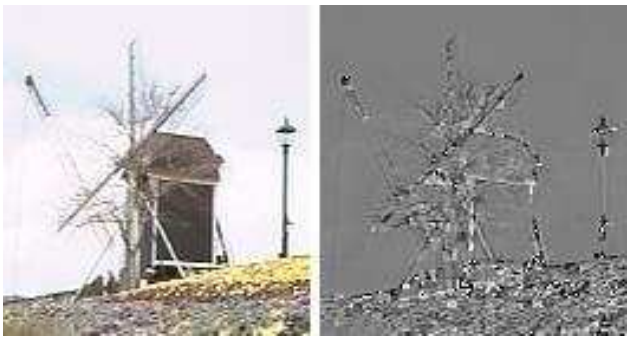

Figure 1 – *Extrait de l'image Flower (CIF) et son résidu de prédiction intra AVC*

Les images intra de AVC subissent une prédiction spatiale : chaque bloc 4x4 ou 8x8 est prédit à partir de ces voisins et dans une des 9 directions possibles [1], le bloc résiduel à coder est alors la différence entre le bloc original et sa prédiction. Pour les macroblocs 16x16, le même procédé est appliqué, mais il n'existe que 4 directions [1].

Notre pré-traitement cherche à exploiter l'orientation de ces blocs sans utiliser de rotation qui nécessite une interpolation, mais en effectuant des permutations circulaires au niveau pixel. Il effectue un cisaillement ("shear") simulant ainsi une pseudo-rotation des blocs. Ceci permet alors de redresser les blocs vers l'horizontale ou la verticale.

2.1 Blocs 4x4

Dans le cas des blocs 4x4, nous avons défini cinq états différents qui correspondent à sept orientations prédéfinies du bloc et qui sont associés à des permutations.

A l'état 0, aucune opération n'est à effectuer parce que soit les blocs sont non-orientés (si leur direction est horizontale ou verticale), soit ils n'ont pas de directions acceptables : ils sont orientés suivant un angle trop éloigné (plus de $\pm 3^o$) des angles prédéfinis pour pouvoir correspondre à une des permutations.

Les autres états spécifient les blocs dont la direction est proche (moins de $\pm 3^o$) des angles définis par le Tableau 1.

État	Angle	État	Angle
1	$+27^o$	2	-27^o
3	$+45^o$	4	-45^o

Tableau 1 – *Les états du cas 4x4*

Pour chacun de ces états, une permutation circulaire est appliquée au niveau pixel afin de simuler une rotation par cisaillement. Ces permutations circulaires nous permettent de nous affranchir d'une étape d'interpolation inhérente à tout processus réel de rotation (matricielle). De plus, par ces simples réarrangements de pixels nous simulons une rotation sans créer de trous dans les coins des blocs.

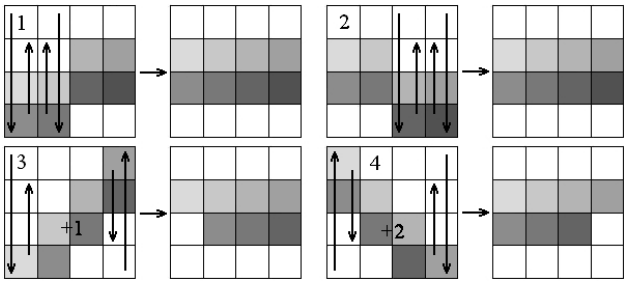

Figure 2 – *Permutations circulaires du cas 4x4*

Dans l'état 1 (cf Figure 2, 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. 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 circulaire 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 Figure 2, au milieu).

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

2.2 Blocs 8x8

Comme dans le cas 4x4, nous définissons ici 9 états d'orientation pour les blocs 8x8 [9].

Comme précédemment, 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.

Les autres états correspondent aux blocs dont la direction est proche des angles : $\pm 14^o$, $\pm 27^o$, $\pm 37^o$ et $\pm 45^o$.

Chacune de ces orientations est associée à un réarrangement des pixels appliqué aux blocs par des permutations circulaires comme dans le cas 4x4 (cf Figure 3).

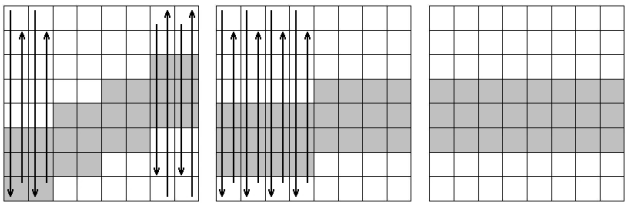

Figure 3 – *Exemple de permutations circulaires du cas 8x8 : l'état 3 (angle de $+27^o$)*

2.3 Blocs 16x16

Comme dans les cas 4x4 et 8x8, on définit ici 17 états d'orientation pour les blocs 16x16 ou macroblocs.

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 prédéfinies.

Les autres états correspondent aux blocs dont la direction est proche des angles : $\pm 7^o$, $\pm 14^o$, $\pm 20^o$, $\pm 27^o$, $\pm 32^o$, $\pm 37^o$, $\pm 41^o$ et $\pm 45^o$.

Chacune de ces directions est associée à un réarrangement des pixels appliqué aux blocs 16x16 grâce à des permutations circulaires comparables à celles des cas 4x4 et 8x8 (cf Figure 4).

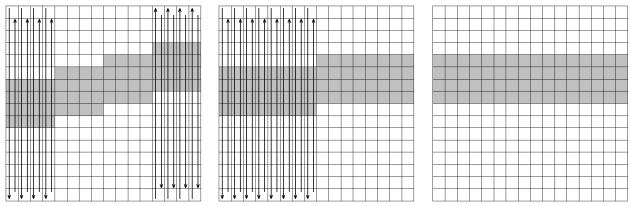

Figure 4 – *Exemple de permutations circulaires du cas 16x16 : l'état 3 (angle de $+14^o$)*

3 Sélection de l'orientation

Avant d'appliquer notre pseudo-rotation, il faut sélectionner la bonne orientation pour chacun des blocs de l'image ou de la séquence vidéo.

Nous nous basons sur l'optimisation débit-distorsion (RDO) de AVC [10]. Cette optimisation consiste à utiliser toutes les combinaisons de modes disponibles et à coder les macroblocs avec celle qui donne les meilleures performances : la plus faible distorsion pour un débit donné ou le meilleur débit pour une distorsion donnée.

Le coût de codage d'un macrobloc dépend donc de deux variables : le débit et la distorsion. Le débit est, dans tous les cas, la somme des débits des blocs le composant ($16\times$(4x4), $4\times$(8x8), ou $1\times$(16x16)). Et la distorsion est toujours la distorsion globale du macrobloc quels que soient les blocs le composant, elle est donnée par l'erreur quadratique du macrobloc reconstruit :

$$D = \sum_{m=0}^{15} \sum_{n=0}^{15} \left(i_{MB}(m,n) - \hat{i}_{MB}(m,n) \right)^2 \qquad (1)$$

où $i_{MB}(m,n)$ est le pixel (m,n) du macrobloc original et $\hat{i}_{MB}(m,n)$ celui du macrobloc reconstruit.

Pour chaque macrobloc, toutes les tailles de blocs disponibles sont testées avec tous les modes de prédiction disponibles. Par rapport à un codeur classique, notre prétraitement ne fait qu'ajouter des combinaisons à tester. Par exemple, dans un codeur vidéo AVC et pour une image intra, cela revient à :

– dans le cas 16x16, au lieu de tester une seule fois chacun des quatres modes de prédiction [1], nous les testons 17 fois avec nos orientations 16x16 (cf 2.3).

– dans le cas 8x8 et pour chaque bloc 8x8, nous testons 9 fois chacun des 9 modes de prédiction [1] avec nos orientations 8x8 (cf 2.2).
– dans le cas 4x4, pour chaque bloc 4x4 du macrobloc, nous testons les 9 modes de prédiction [1] 5 fois avec nos cinq orientations candidates (cf 2.1).

Donc pour le codage d'un macrobloc, tous les cas sont testés et la meilleure combinaison est conservée dans chacun des cas. Ces trois solutions sont ensuite comparées à l'aide de la même méthode d'optimisation débit-distorsion afin d'en isoler la meilleure combinaison, combinaison utilisée par la suite pour le codage réel du macrobloc.

Cette méthode de sélection des orientations est efficace, mais complexes en termes d'évaluations de débit-distorsion : par exemple, dans un codeur vidéo AVC et pour une image intra, nous testons $17\times4+4\times(9\times9)+16\times(5\times9) = 1112$ modes par macrobloc (contre 184 pour AVC). Cette complexité, quasiment multipliée par 7, ne touche cependant que la sélection débit-distorsion des modes de prédiction intra qui ne représente qu'une partie du codage AVC. La complexité globale de l'algorithme AVC n'en est que très peu affectée.

Pour réduire la complexité de cette première méthode, nous nous attacherons à développer une autre méthode calculant la direction de chaque bloc afin d'appliquer directement la meilleure permutation.

4 Codage et décodage

4.1 Ensemble du macrobloc

Après avoir sélectionné le meilleur mode (le meilleur mode étant une combinaison de taille de blocs, de permutations et le cas échéant de modes de prédiction), le codage du macrobloc est effectué par le codeur image ou vidéo utilisé. Chaque macrobloc est transformé par DCT, quantifié et codé entropiquement, et ce quelque soit le codeur DCT utilisé. Par exemple, dans le cas de AVC, après la prédiction intra, les résidus (orientés ou non) sont transformés par la DCT entière 4x4 ou 8x8 de AVC, quantifiés et codés avec CABAC [11] : le Context-based Adaptive Binary Arithmetic Coding de AVC parvient à de bonne performance en compression grâce à (a) une sélection de modèles de probabilité pour chacun des éléments de syntaxe en fonction du contexte de cet élément, (b) une adaptation de l'estimation des probabilités basée sur des statistiques locales et (c) l'utilisation d'un codage arithmétique (il n'est utilisable qu'en profils Main et High (défini dans FRExt [9]) sinon CAVLC [1] [12] le remplace dans les autres profils). Les macroblocs orientés sont traités de la même manière que les non-orientés.

Avec notre pré-traitement, les blocs sont redressés vers l'horizontale ou la verticale avant la transformation, ceci implique que la transformée DCT est plus efficace sur ces données et donc améliore les performances générales en débit-distorsion.

Le décodage est aussi effectué par le codeur image ou vidéo utilisé. Les macroblocs sont décodés entropiquement (avec CABAC pour AVC), déquantifiés, et subissent une transformation DCT inverse (DCT entière 4x4 ou 8x8 pour AVC). Les macroblocs qui ont été orientés avant codage doivent être réorientés après le décodage en utilisant les permutations inverses de celles définies précédemment en section 2. Pour cela, nous devons transmettre les informations de permutations nécessaire à cette réorientation.

4.2 Information de permutation

L'information de permutation correspondant à l'état d'orientation du meilleur mode de codage est écrite, quelque soit la taille de bloc utilisée et même si elle correspond à l'état 0, dans l'entête du bloc. Dans le cas 16x16, cette information est écrite dans l'entête du macrobloc (pour le cas AVC, après le mode de prédiction intra). Dans les cas 8x8 et 4x4, elle est écrite dans l'entête de chacun des blocs (pour le cas AVC, entre le mode de prédiction intra pour les luminances et le mode de prédiction intra pour les chrominances).

Pour le 16x16, l'information de permutation est codée en utilisant CABAC et le même contexte que celui défini pour les modes de prédiction intra 16x16 dans AVC. Pour les autres cas, cette information est prédite en fonction de son voisinage (comme pour les modes de prédiction intra 4x4 et 8x8 de AVC) avant d'être codée avec CABAC et les contextes des modes de prédiction intra 4x4 et 8x8 respectivement. Les états d'orientation des blocs adjacents au dessus (A) et à gauche (B) (cf Figure 5) sont comparés s'ils sont disponibles sinon ils sont considérés nuls. Celui possédant la plus grande valeur définit l'état d'orientation le plus probable pour le bloc à coder (C). Si l'orientation du bloc (C) est égale à l'état d'orientation le plus probable, on ne transmet qu'un drapeau sinon cette orientation est complètement codée.

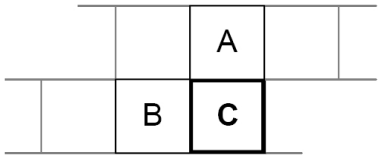

Figure 5 – *Blocs adjacents 4x4 ou 8x8 du bloc à coder*

Le décodage est effectué classiquement, le décodeur image ou vidéo utilisé lit toutes les informations y compris celle de permutations. Les macroblocs sont ensuite décodés, et si l'orientation des blocs n'est pas nulle ils sont réorientés. Dans le cas de AVC, les macroblocs résiduels sont décodés, puis réorientés (si besoin), et enfin ajoutés à leurs prédictions pour obtenir les macroblocs reconstruits.

5 Résultats expérimentaux

Le pré-traitement proposé a été implémenté dans le JM10 [13] fourni par le JVT, et uniquement pour les luminances des images résiduelles intra. Tous les tests ont été réalisés en profil High et à niveau 4.0 permettant ainsi l'utilisation

de CABAC et de FRExt [9] avec la transformée DCT 8x8. Les séquences sont générées en faisant varier les valeurs de QP des slices intra sur toute la plage disponible (0-51 et fixée à 28 pour les inter), elles sont alors composées de vingt images I.
Nous présentons ici deux séries de résultats : la première sans le codage de l'information de permutation, et la seconde avec cette information encodée.

5.1 Sans information de permutation

Les résultats sans le codage de l'information de permutation pour la séquence Container en CIF à 15Hz sont présentés sur la Figure 6 et ceux pour Mobile&Calendar en CIF à 15Hz sur la Figure 7.

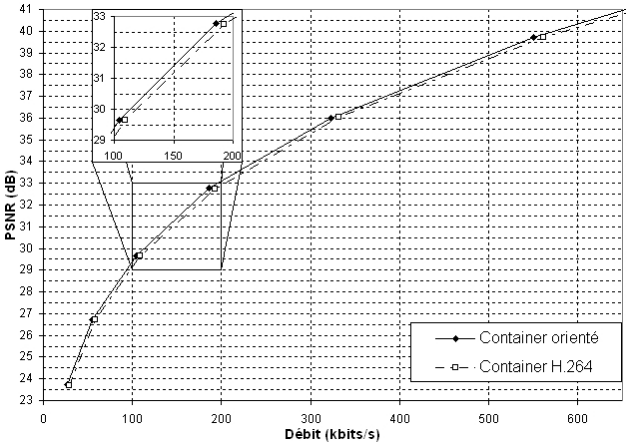

Figure 6 – *Résultats pour Container (CIF)*

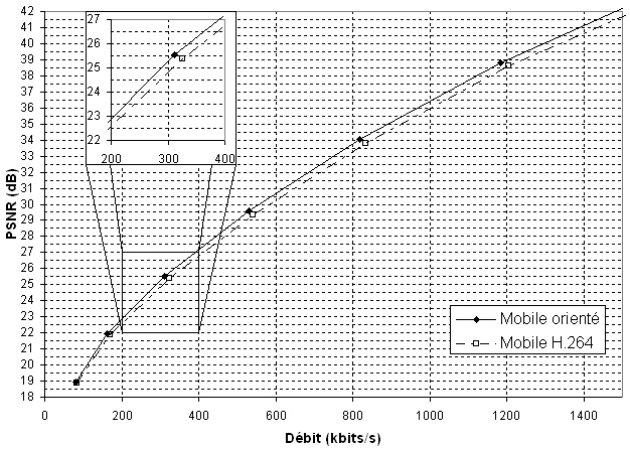

Figure 7 – *Résultats pour Mobile&Calendar (CIF)*

On peut voir sur ces figures que notre pré-traitement améliore le codage AVC à tous les débits. De plus, ces tests ont montré que toutes les possibilités de notre pré-traitement ont été exploitées : les images de la séquence Container sont composées de 396 macroblocs. A 550kbits/s, 179 de ces macroblocs ont été codés en intra 4x4, 188 en intra 8x8 et 29 en intra 16x16. Et dans tous les cas, les orientations ont pu être utilisées. L'amélioration de PSNR est à partir de 150kbits/s de près de 0.25dB pour Container et de près de 0.3dB pour Mobile&Calendar, et jusqu'à plus de 1dB dans les deux cas à très haut débit (à partir de 2500-3000kbits/s). Des résultats similaires ont été obtenus avec un grand nombre de séquences telles que Akiyo, Foreman, Bus, Tempete comme indiqué dans le Tableau 2. Par exemple, la séquence Tempete génère un gain de 0.32dB par rapport à AVC à 200kbits/s, et un gain supérieur à 0.5dB au delà de 3100kbits/s soit pour un QP inférieur à 9.

Séquence CIF	$d = 200kbits/s$ $\Delta_{PSNR} =$	$\Delta_{PSNR} > +0.5dB$ $d >$	$QP <$
Akiyo	+0.12dB	1200kbits/s	11
Bus	+0.18dB	2700kbits/s	9
Flower	+0.29dB	3000kbits/s	10
Football	+0.16dB	2600kbits/s	6
Foreman	+0.18dB	1900kbits/s	8
Tempete	+0.32dB	3100kbits/s	9
Séquence QCIF	$d = 80kbits/s$ $\Delta_{PSNR} =$	$\Delta_{PSNR} > +0.5dB$ $d >$	$QP <$
Carphone	+0.21dB	500kbits/s	9
Foreman	+0.19dB	600kbits/s	7

Tableau 2 – *Les résultats sur d'autres séquences*

5.2 Avec information de permutation

Dans un second temps, nous codons l'information de permutation selon la méthode présentée précédemment dans la section 4.2. Les courbes de débit-distorsion correspondantes pour les séquences Container et Mobile&Calendar sont présentées sur les Figures 8 et 9 respectivement.

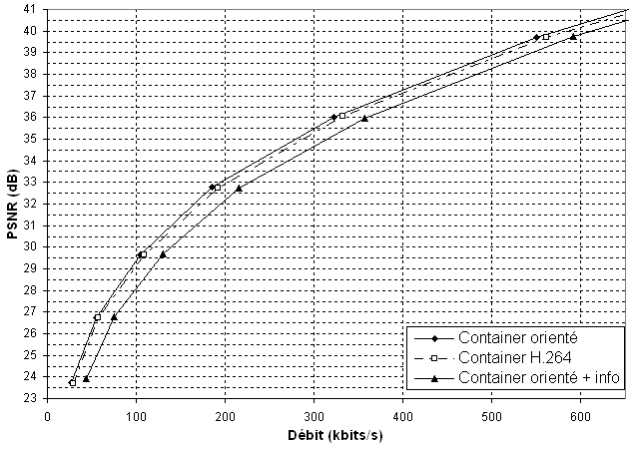

Figure 8 – *Résultats pour Container avec l'information de permutation*

Ces figures montrent que le débit nécessaire pour encoder l'information de permutation est supérieur au gain apporté

par notre pré-traitement jusqu'à un certain (haut) débit (ici près de 2000kbits/s). Pour la séquence Mobile&Calendar à 200kbits/s, l'information de permutation nous fait perdre 0.7dB contre un gain de 0.3dB sans cette information, soit une perte de 0.4dB par rapport à AVC. L'encodage simple de l'information que nous réalisons ici n'est pas efficace et nécessitera une amélioration future : en améliorant la prédiction du mode d'orientation le plus probable et en définissant de nouveaux contextes et éléments de syntaxe pour ces orientations et pour chaque taille de blocs (4x4, 8x8 et 16x16).

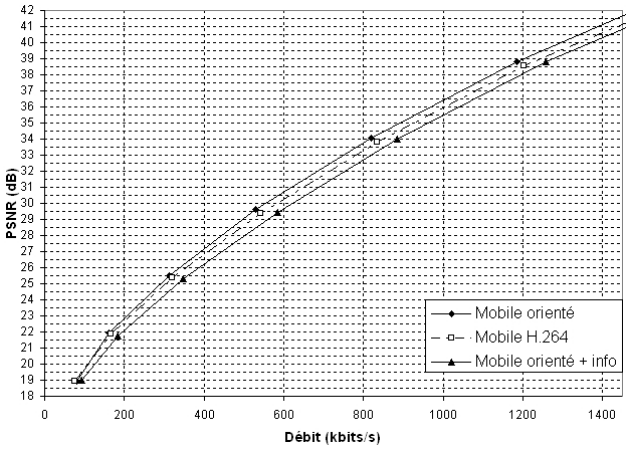

Figure 9 – *Résultats pour Mobile&Calendar avec l'information de permutation*

6 Conclusions et perspectives

Nous avons introduit dans cet article un pré-traitement pour des codeurs basés blocs qui tient compte de l'orientation de ces blocs pour en améliorer les performances. Prendre en considération l'orientation avant l'encodage nous permet d'adapter les blocs à la transformée sans la modifier. Ces blocs ou macroblocs sont orientés en appliquant de simples permutations circulaires au niveau pixel évitant ainsi les problèmes inhérents aux méthodes classiques de rotation avec interpolation. Le codeur doit ensuite encoder et transmettre l'information de permutation en utilisant CABAC. Le codage que nous adoptons ici pour cette information est très proche de celui utilisé pour les modes de prédiction intra de AVC. Ce pré-traitement semble prometteur puisqu'il n'est appliqué ici qu'aux images résiduelles intra d'un codeur AVC, images qui ne représentent qu'une faible proportion des images et des blocs d'une séquence vidéo.
Nos futurs travaux s'attacheront à améliorer ce pré-traitement. En particulier, nous travaillerons à réduire l'impact de l'information additionnelle d'orientation qui doit être transmise au décodeur, et de réduire la complexité de l'algorithme. Nous avons aussi l'intention d'étendre notre méthode au codage des chrominances et, dans le cas de la vidéo, aux images inter avec des blocs hybrides (16x8, 8x16, 8x4 et 4x8).

Références

[1] JVT - ISO/IEC 14496-10 AVC - ITU-T Recommendation H.264. *Advanced video coding for generic audio-visual services*. Draft ITU-T Recommendation and Final Draft International Standard, JVT-G050r1, 2003.

[2] E. Candès et D. Donoho. Curvelets, multiresolution representation, and scaling laws. Dans *Wavelet Applications in Signal and Image Processing VIII*. Proc. SPIE 4119, 2000.

[3] M.N. Do et M. Vetterli. The contourlet transform : An efficient directional multiresolution image representation. *IEEE Transactions on Image Processing*, 14(12) :2091–2106, décembre 2005.

[4] E. Le Pennec et S. Mallat. Sparse geometric image representations with bandelets. *IEEE Transactions on Image Processing*, 14(4) :423–438, avril 2005.

[5] G. Peyré et S. Mallat. Discrete bandelets with geometric orthogonal filters. *IEEE International Conference on Image Processing (ICIP'05)*, 1 :65–68, septembre 2005.

[6] V. Velisavljević, B. Beferull-Lozano, M. Vetterli, et P.L. Dragotti. Directionlets : Anisotropic multidirectional representation with separable filtering. *IEEE Transactions on Image Processing*, septembre 2005.

[7] M. Unser, P. Thévenaz, et L.P. Yaroslavsky. Convolution-based interpolation for fast, high-quality rotation of images. *IEEE Transactions on Image Processing*, 4(10) :1371–1381, octobre 1995.

[8] ISO/IEC 10918. *Digital Compression and Coding of Continuous-Tone Still Images*. JPEG, 1994.

[9] 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.

[10] T. Wiegand, H. Schwarz, A. Joch, F. Kossentini, et G.J. Sullivan. Rate-constrained coder control and comparison of video coding standards. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(7) :688–703, juillet 2003.

[11] D. Marpe, H. Schwarz, et 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*, 13(7) :620–636, juillet 2003.

[12] T. Wiegand, G.J. Sullivan, G. Bjontegaard, et A. Luthra. Overview of the H.264/AVC video coding standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(7) :560–576, juillet 2003.

[13] Joint model 10, 2006. http :\\ iphome.hhi.de\ suehring\ tml\ index.htm.