Amélioration du codage H.264 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}@orange-ftgroup.com

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

Résumé

Cet article introduit un pré-traitement pour le codeur H.264/MPEG4-AVC qui utilise avantageusement l'orientation des blocs de la transformée. 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 afin de s'adapter à la transformée. Les blocs résiduels sont redressés, avant transformation, vers l'horizontale ou la verticale à l'aide de permutations circulaires appliquées au niveau pixel permettant alors de simuler des pseudo-rotations dans ces blocs. Elles sont définies pour toutes les tailles de partitionnement des blocs (16x16 à 4x4) et pour les deux transformées entières du codeur AVC (8x8 et 4x4). La taille et les orientations de chaque bloc sont sélectionnées à l'aide d'un critère débit-distorsion. Ce pré-traitement, introduit dans un codeur AVC [1] et appliqué aux images résiduelles permet d'en améliorer les performances.

Mots clefs

Transformation directionnelle, orientation, permutation circulaire, H.264-MPEG4/AVC.

1 Introduction

H.264/AVC [1] est le nouveau standard vidéo international de l'ITU-T (comme Recommendation H.264) et de l'ISO/IEC (ie Standard International 14496-10 c'est à dire MPEG-4 part 10) Advanced Video Coding (AVC). Il améliore considérablement les standards précédents du domaine, notamment MPEG-2, en termes d'efficacité en compression permettant ainsi de cibler une plus large gamme d'applications : diffusion vidéo sur tout type de réseaux (DSL, terrestre, mobile...), vidéo à la demande, services de streaming, applications conversationnelles...

Bien qu'il 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 débloqueur dans la boucle... [1]. En outre, ce codeur a été amélioré en incluant, par exemple, une taille de transformée 4x4, un codage hiérarchique des blocs ou le codage arithmétique adaptatif contextuel [2].

Cependant, il reste toujours une possibilité d'amélioration. En effet, nous avons observé que les blocs après prédiction (spatiale ou temporelle) possèdent toujours une orientation. Afin d'exploiter cette observation, nous proposons de pré-traiter les blocs avant la transformation DCT d'AVC : nous choisissons la meilleure permutation à appliquer au bloc parmi un ensemble de permutations correspondant aux orientations des blocs. Les résultats expérimentaux montrent une possible amélioration du codage AVC quand notre étape de pré-traitement est appliquée.

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 des informations dans la Section 4. Enfin, quelques résultats numériques de notre pré-traitement appliqué aux images résiduelles AVC sont présentés en Section 5 avant de conclure et de donner quelques perspectives.

2 Orientation des blocs de la transformée

La transformée DCT (Discrete Cosine Transform) d'AVC est la plus connue des transformées basées bloc, elle est utilisée dans la plupart des standards de codage vidéo : MPEGx et H.26x comme H.264-MPEG4/AVC [1]. Cette transformée s'applique à des blocs de coefficients qui peuvent avoir une taille variable : 4x4 ou 8x8 pour les DCT entières d'AVC. De plus, les coefficients de ces blocs sont issus de prédiction spatiale ou temporelle.

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, elle reste donc réversible (sans perte). Pour cela, notre pré-traitement effectue des permutations circulaires au niveau pixel, il réalise 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 En mode intra

Les images intra d'AVC subissent une prédiction spatiale : chaque bloc 4x4 ou 8x8 est prédit à partir de ses voisins et dans une des 9 directions possibles [1] (cf Figure 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].

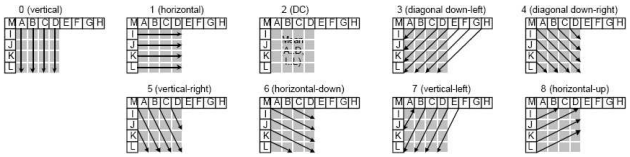

Figure 1 – *Les 9 prédictions spatiales intra 4x4 et 8x8 d'AVC*

Les blocs de coefficients résiduels à traiter présentent alors encore des motifs réguliers orientés comme représenté sur la Figure 2.

Figure 2 – *Image résiduelle intra AVC après prédiction spatiale*

Blocs 4x4. Dans le cas des blocs 4x4, nous avons défini sept 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 (leur direction est horizontale ou verticale), soit ils n'ont pas de direction acceptable : leurs orientations sont trop éloignées (plus de $\pm 3^o$) des angles prédéfinis pour être associé à un des autres états.
Les autres états spécifient les blocs dont la direction est proche (moins de $\pm 3^o$) des angles : $\pm 27^o$, $\pm 45^o$ et $\pm 63^o$. Pour chacun de ces états, une permutation circulaire est appliquée au niveau pixel afin de simuler une rotation par cisaillement. Ces permutations 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.
Dans l'état 1 (cf Figure 3, 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 Figure 3).

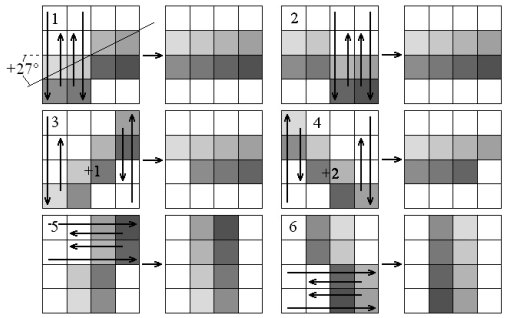

Figure 3 – *Permutations circulaires du cas 4x4*

Cette figure 3 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.

Blocs 8x8. Comme dans le cas 4x4, on peut définir 15 états d'orientation pour les blocs 8x8 et la DCT 8x8 [3]. Cependant, afin de limiter le coût de codage de l'information d'orientation, nous n'avons défini ici que 9 états d'orientation (les plus utilisés statistiquement) : 0, $\pm 14^o$, $\pm 45^o$, $\pm 63^o$ et $\pm 76^o$ (cf Figure 4).

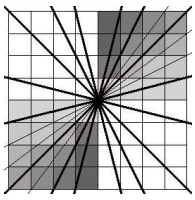

Figure 4 – *Orientations définies dans le cas 8x8 (en gras)*

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. Et dans chacun des autres états, les orientations sont associées à un réarrangement des pixels appliqué aux blocs par des permutations circulaires comme dans le cas 4x4.

Blocs 16x16. Comme dans les cas 4x4 et 8x8, on peut définir 31 états d'orientation pour les blocs 16x16 ou macroblocs, mais seulement 9 sont utilisés ici (les plus utilisés statistiquement) par souci de limitation du coût de codage de ces informations : 0, $\pm 7^o$, $\pm 14^o$, $\pm 76^o$ et $\pm 83^o$.
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. Et dans chacun des autres états, ces directions sont associées à un réarrangement des pixels appliqué aux macroblocs grâce à des permutations circulaires comparables à celles des cas 4x4 et 8x8.

2.2 En mode inter

En inter, les images AVC subissent une prédiction temporelle réalisée par une estimation compensation en mouvement. Cette prédiction s'applique au macrobloc et a la possibilité de le découper en différentes partitions comme illustré en Figure 5. Elle peut utiliser une ou plusieurs images de référence qui peuvent être : backward (image de référence passée), forward (image de référence future) ou les deux.

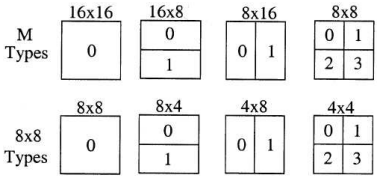

Figure 5 – *Les différentes partitions possibles d'un macrobloc inter*

Les blocs de coefficients résiduels présentent alors encore des motifs réguliers orientés comme représenté sur la Figure 6.

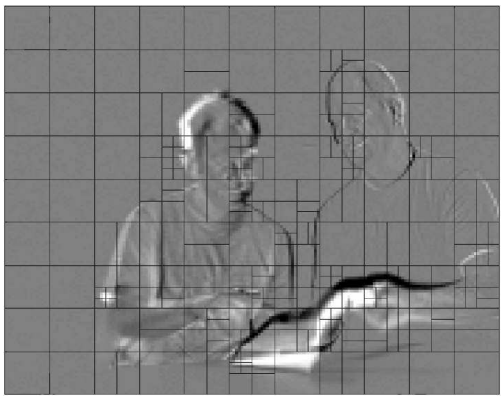

Figure 6 – *Image résiduelle inter AVC avec sa découpe après prédiction temporelle*

Pour les blocs de tailles 4x4, 8x8 et 16x16 en inter, on applique les mêmes permutations circulaires que celles définies pour le cas intra. Par contre, il faut aussi, dans ce mode de codage, définir nos permutations circulaires pour toutes les autres tailles de partitions.
Pour les blocs de tailles 16x8 et 8x16, on redéfinit les mêmes permutations circulaires que dans le cas 16x16, mais pour une taille rectangulaire de bloc. La même opération est effectué avec les permutations du cas 8x8 pour être appliquées aux cas 8x4 et 4x8.

3 Sélection de l'orientation

Afin d'améliorer la transformation DCT grâce aux pseudo-rotations, il faut sélectionner la bonne orientation pour chacun des blocs de la séquence vidéo. Cette sélection est basée sur l'optimisation débit-distorsion d'AVC [4] qui consiste à tester tous modes disponibles et à coder les macroblocs avec le plus performant : 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 et dépend de deux variables : le débit et la distorsion. Le débit d'un macrobloc est la somme des débits des blocs le composant, ils sont eux-mêmes la combinaison des débits des coefficients et de l'en-tête contenant l'information d'orientation. La distorsion est toujours la distorsion globale du macrobloc quel que soit son partitionnement, elle est donnée par l'erreur quadratique du macrobloc reconstruit.
Pour le codage d'un macrobloc, chaque taille de blocs est testée avec tous les modes de prédiction et d'orientation disponibles. Ces solutions sont comparées à l'aide de l'optimisation débit-distorsion afin d'en isoler la meilleure combinaison (taille de bloc, prédiction(s), orientation(s)), combinaison utilisée par la suite pour le codage réel du macrobloc. Par rapport à un codeur classique, notre prétraitement 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 que les sélections des modes de prédiction en intra et du partitionnement en inter qui ne représentent qu'une partie du codage AVC.

4 Codage et décodage

4.1 Ensemble du macrobloc

Après avoir sélectionné la meilleure combinaison (taille de blocs, permutations et le cas échéant modes de prédiction), le codage du macrobloc est effectué par le codeur vidéo. Chaque macrobloc résiduel est, après prédiction et orientation (si besoin), transformé avec la DCT entière 4x4 ou 8x8, quantifié et codé entropiquement à l'aide de CABAC (Context-based Adaptive Binary Arithmetic Coding) [2]. Les macroblocs résiduels 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 DCT qui est plus efficace sur ces données en ligne-colonne et donc améliore les performances générales en débit-distorsion.
Le décodage est aussi effectué par le codeur vidéo. Les macroblocs sont décodés entropiquement avec CABAC, déquantifiés, et subissent une transformation DCT inverse entière 4x4 ou 8x8. Les macroblocs qui ont été orientés avant codage doivent être réorientés après le décodage en utilisant les permutations inverses de celles définies 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 dans l'en-tête du bloc quelle que soit sa taille même si elle correspond à l'état 0. Dans le cas 16x16, cette information est écrite dans l'en-tête du macrobloc après le mode de prédiction. Dans les autres cas, elle est écrite dans l'en-tête de chacun des blocs entre le mode de prédiction et les coefficients de luminance.
Ces informations sont toutes codées en utilisant un codeur arithmétique externe qui nous permet d'obtenir le code à insérer dans le bitstream ainsi que le débit associé.
Dans le cas 16x16, si l'état d'orientation est 0 on ne transmet qu'un drapeau sinon on code complètement cette orientation. Dans les autres cas, cette information est prédite en fonction de son voisinage (comme les modes de prédiction intra d'AVC) avant d'être codée. Les états d'orientation des blocs adjacents au-dessus et à gauche 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. Si l'orientation du bloc à coder est égale à cette orientation la plus probable, on ne transmet qu'un drapeau sinon cette orientation est complètement codée.
Le décodage est effectué classiquement, le décodeur AVC lit toutes les informations y compris celle de permutation. Les macroblocs résiduels sont décodés, ensuite si l'orientation des blocs n'est pas nulle ils sont réorientés, et enfin ajoutés à leurs prédictions pour obtenir les macroblocs reconstruits.

5 Résultats expérimentaux

Le pré-traitement proposé a été implémenté dans le JM10 [5] fourni par le JVT, et uniquement pour les luminances des images résiduelles. Tous les tests ont été réalisés en profil High et à niveau 4.0 permettant ainsi l'utilisation de FREXt [3] avec la transformée DCT 8x8.
Nous présentons deux séries de résultats : une uniquement sur le codage intra avec ou sans le codage de l'information de permutation, et la seconde en intégrant l'inter.

5.1 Résultats intra

Les tests de cette section ont été réalisés en faisant varier les valeurs de QP des slices intra (QPI) sur toute la plage disponible (de 0 à 51) pour une séquence composée de 20 images intra. Un premier test a été effectué sans le codage de l'information de permutation permettant de voir le gain que peut apporter notre méthode, et un second test avec cette information afin de bien identifier la perte liée à ce codage.
Les résultats pour la séquence Flower en CIF à 15Hz sont présentés sur la Figure 7 et ceux pour Soccer en SD à 60Hz sur la Figure 8.
D'après ces figures, notre pré-traitement sans le codage de l'information améliore le codage AVC à tous les débits. Cette amélioration de PSNR est de 0.28dB pour Flower à partir de 150kbits/s (avec 73% des macroblocs contenant au moins un bloc 4x4 orienté) et pour Soccer à partir de 500kbits/s, et jusqu'à plus de 1dB dans les deux cas à haut débit.
Le codage des informations de permutations entraîne une perte en débit pour notre méthode, perte supérieure au gain apporté par notre pré-traitement jusqu'à un certain (haut) débit. Aux débits cités précédemment, on a une perte de 0.18dB pour Flower et de 0.35dB pour Soccer par rapport à AVC, le codage a fait perdre respectivement 0.46dB pour Flower et 0.50dB pour Soccer. L'encodage simple de l'information que nous réalisons ici n'est pas assez efficace et nécessitera une amélioration future : en modifiant la prédiction de l'orientation la plus probable et en utilisant un codage contextuel tel que CABAC impliquant la définition de nouveaux contextes et éléments de syntaxe pour ces orientations et pour chaque taille de blocs.

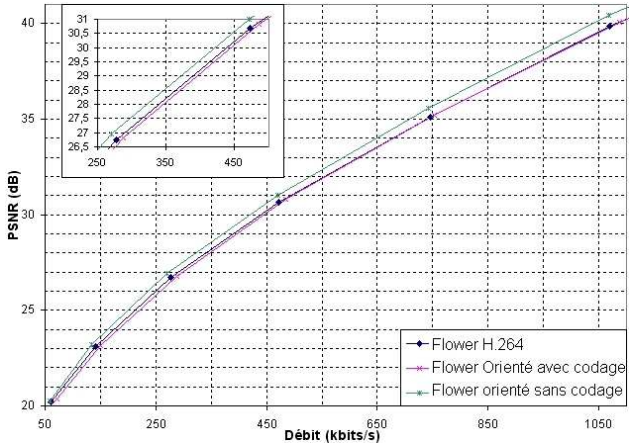

Figure 7 – *Résultats intra pour Flower (CIF 15Hz)*

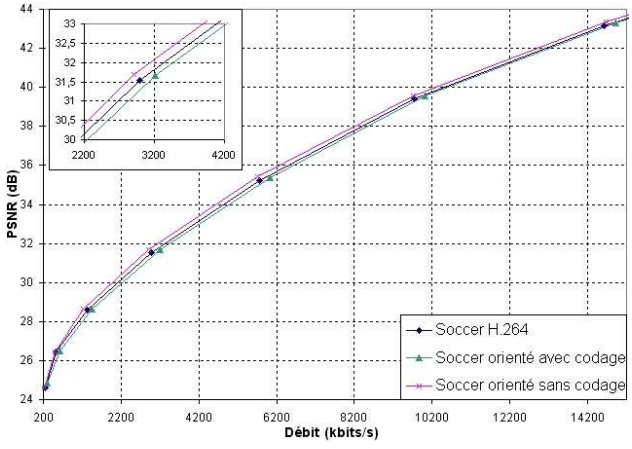

Figure 8 – *Résultats intra pour Soccer (SD 60Hz)*

Des résultats similaires ont été obtenus avec un grand nombre de séquences telles que Akiyo, Foreman, City

comme indiqué dans le Tableau 1. La mesure de Björntegaard [6] utilisée ici nous donne le gain en PSNR et le pourcentage de débit conservé par notre méthode par rapport à AVC sur une plage de 4 valeurs de QP (ici 20-25-30-35). Par exemple, la séquence Tempete génère un gain de 0.43dB et diminue de 4.60% du débit par rapport à AVC sans le codage de l'orientation, et avec le codage de l'information perd 0.17dB et 1.81% du débit par rapport à AVC.

Séquence	Sans le codage		Avec le codage	
CIF	PSNR	Débit	PSNR	Débit
Akiyo	+0.32	-4.75	-0.38	+5.82
Bus	+0.37	-4.03	-0.19	+2.09
Container	+0.29	-4.01	-0.21	+3.00
Flower	+0.52	-4.10	-0.03	+0.31
Foreman	+0.45	-7.65	-0.23	+4.23
Mobile	+0.80	-6.21	-0.12	+0.96
Tempete	+0.43	-4.60	-0.17	+1.81
SD	PSNR	Débit	PSNR	Débit
City	+0.26	-3.46	-0.16	+2.19
Crew	+0.24	-3.50	-0.13	+2.27
Soccer	+0.24	-3.27	-0.14	+1.95

Tableau 1 – *Les résultats intra pour d'autres séquences*

5.2 Résultats inter

Les tests inter ont été réalisés en fixant les valeurs de QP des slices inter à la valeur du QP des slices intra plus 1 (standard conformance tests) et en faisant varier le QP des slices intra (QPI). Comme précédemment, nous avons réalisé un test sans le codage de l'information de permutation et un second avec cette information.
Les résultats de la séquence Tempete en CIF à 15Hz et ce pour la totalité de la séquence IPPP sont présentés sur la Figure 9.

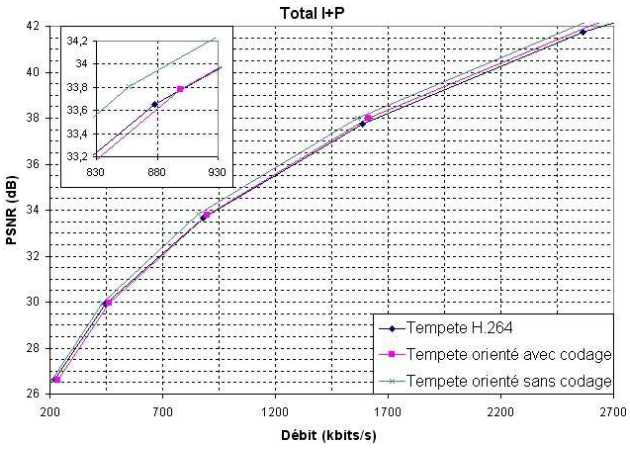

Figure 9 – *Résultats globaux pour Tempete (CIF)*

Cette Figure 9 indique que l'inter permet d'améliorer les performances générales de notre méthode. En effet, les gains de l'inter s'ajoutent à ceux de l'intra ainsi nous améliorons le codage AVC à partir d'un certain débit (ici 2200kbits/s), débit moins élevé qu'avec uniquement les intra (autour de 2800kbits/s pour cette séquence). Ce gain en inter est pour cette séquence d'environ 1.1% de débit à PSNR constant (+0.01dB pour notre méthode).
Des résultats similaires ont été observés sur de nombreuses autres séquences telles que Akiyo, Foreman, City...

6 Conclusions et perspectives

Nous avons présenté un pré-traitement pour le codeur AVC qui tient compte de l'orientation des blocs pour en améliorer les performances. Les blocs sont adaptés à la transformée sans qu'elle ne soit modifiée. Pour cela, ils sont orientés en appliquant de simples permutations circulaires au niveau pixel. Le codeur doit ensuite encoder et transmettre ces informations de permutation. Ce pré-traitement semble prometteur puisqu'il permet d'améliorer les performances d'un codeur AVC, mais le coût de codage des informations complémentaires ne permet pas de surpasser AVC.
Nos futurs travaux s'attacheront à améliorer ce prétraitement. En particulier, nous travaillerons à réduire l'impact de l'information additionnelle d'orientation qui doit être transmise au décodeur (par l'utilisation de CABAC par exemple). Nous avons aussi l'intention d'utiliser les vecteurs de mouvement et les images de références inter afin de prédire les orientations des partitions courantes et d'éviter ainsi leur transmission.

Références

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

[2] 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.

[3] 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.

[4] 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.

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

[6] G. Bjöntegaard. Calculation of Average PSNR Differences between RD-curves. *VCEG Contribution VCEG-M33*, avril 2001. Austin, Texas.