Extension du codeur SPIHT au codage d'images hyperspectrales

J.-L. Gutzwiller[1], M. Hariti[2], M. Barret[1], E. Christophe[2], C. Thiebaut[2] et P. Duhamel[3]

[1]SUPELEC, Équipe IMS, 2, rue É. Belin, 57070 Metz Technopôle
[2]CNES, 18 avenue Edouard Belin, 31401 Toulouse Cedex 9
[3]CNRS, SUPELEC-LSS, 3 rue Joliot-Curie, Plateau de Moulon, 91192 Gif-sur-Yvette.
{jean-louis.gutzwiller, michel.barret}@supelec.fr, mohamed.hariti@liris.cnrs.fr,
{carole.thiebaut, emmanuel.christophe}@cnes.fr, pierre.duhamel@lss.supelec.fr

Résumé

Le codage par transformée et décomposition en ondelettes (DO) est efficace pour coder les images hyperspectrales en suivant le standard JPEG2000 (partie 2). Toutefois la complexité du codeur EBCOT et de l'algorithme d'allocation optimale entre composantes ne permet pas son utilisation à bord d'un satellite aujourd'hui. Nous présentons une extension de l'algorithme SPIHT qui permet un codage progressif en qualité d'images hyperspectrales ayant subi une DO 2D par composantes et une transformation linéaire entre elles. Les performances obtenues (PSNR versus débits) sont légèrement inférieures à celles de JPEG2000, pour une complexité significativement plus faible.

Mots clefs

Compression d'images hyperspectrales, codeur 3D par arbres, SPIHT-3D.

1 Introduction

Pour traiter et stocker le volume de données issues de capteurs d'images hyperspectrales embarqués dans les satellites de nouvelle génération, le développement d'algorithmes de compression capables d'exploiter les redondances, *spectrale et spatiale*, de ces données est nécessaire. Une image hyperspectrale $\mathbf{X}$ est constituée de N composantes $\mathbf{X}_1, \ldots, \mathbf{X}_N$. Chaque composante est une photographie 2D (ayant N_ℓ lignes et N_c colonnes) d'une même scène prise dans une bande spectrale très étroite, allant du visible au proche infrarouge en général. Les différentes composantes correspondent à différents intervalles de longueur d'ondes λ et sont rangées par λ croissant ou décroissant. Ainsi pour un point de la scène en position (i, j), la séquence $X_1(i, j), \ldots, X_N(i, j)$ correspond à un échantillonnage fin de son spectre. Dans la suite, suivant le contexte, $\mathbf{X}_i$ désignera une image 2D de dimension $N_\ell \times N_c$ ou une matrice ligne de dimension $L = N_\ell N_c$ obtenue en balayant tous les pixels de la composante dans un ordre préétabli (par exemple ligne par ligne). De même, $\mathbf{X}$ désignera une matrice de dimension $N \times L$ dont la i-ème ligne vaut $\mathbf{X}_i$ ou un tenseur d'ordre 3 d'éléments $X(i, j, k)$ avec $1 \leq i \leq N_\ell$, $1 \leq j \leq N_c$ et $1 \leq k \leq N$.

Le codage en sous-bandes et par transformée est efficace pour réduire les redondances, spatiale et spectrale, de ces données [1, 2]. Parmi les différents schémas de compression qui existent, nous étudions celui qui consiste, lors du codage, à centrer chaque composante de l'image en lui retranchant sa valeur moyenne (pour simplifier les expressions, nous supposerons ces N moyennes nulles dans la suite), puis à appliquer la même décomposition en ondelettes (DO) 2D à chaque composante $\mathbf{X}_i$, pour réduire la redondance spatiale, et une transformation linéaire $\mathbf{A}$ entre composantes, pour réduire la redondance spectrale. Le résultat de la DO appliquée à toute l'image $\mathbf{X}$ peut se mettre sous la forme $\mathbf{X}\mathbf{W}^T$, où ^{T} désigne la transposée et $\mathbf{W}$ la matrice de la DO. Les coefficients transformés sont alors les éléments de la matrice $\mathbf{Y} = \mathbf{A}\mathbf{X}\mathbf{W}^T$. Ces derniers sont ensuite quantifiés et codés. Au décodage, en notant $\widehat{\mathbf{Y}}$ la matrice des coefficients transformés quantifiés, une approximation de l'image $\mathbf{X}$ est construite en appliquant les transformations inverses : $\widehat{\mathbf{X}} = \mathbf{A}^{-1}\widehat{\mathbf{Y}}\mathbf{W}^{-T}$. Ce schéma de compression est dit *séparable* car l'ordre dans lequel la DO et la transformation A sont appliquées ne change pas le résultat. Le schéma séparable est compatible avec la partie 2 du standard JPEG2000 [3] et avec les recommandations du CCSDS pour le codage embarqué d'images satellitaires [4]. Associé au codeur EBCOT [5] et à son algorithme complexe d'allocation optimale de débits entre composantes, il donne les meilleures performances reconnues à ce jour [6, 2] à moyens et hauts débits, quand il est utilisé avec la DO 9/7 de Daubechies et la transformation de Karhunen-Loève (TKL).

Il est bien connu que généralement la TKL n'est pas optimale en codage par transformée, quand les données ne sont pas gaussiennes. Une précédente étude [7] a montré que, sans l'hypothèse de gaussianité, une transformation optimale, notée `OrthOST` pour *Orthogonal Optimal Spectral Transform* dans la suite, peut être calculée par des algorithmes d'analyse en composantes indépendantes modifiés et que, sur des images réelles, elle donne de meilleures performances que la TKL à bas, moyens et hauts débits. L'inconvénient majeur des `OrthOST` est la grande complexité des algorithmes permettant de les calculer, ce qui rend im-

possible leur utilisation dans un système de compression embarqué. Ce défaut peut-être contourné (voir [8]) en appliquant la même méthode que dans [9], qui consiste à calculer une fois pour toutes, sur une base d'apprentissage constituée d'images issues d'un même capteur, une `OrthOST`, dite *exogène*, puis de l'appliquer à d'autres images issues du même capteur. Les performances en compression avec pertes et quasi sans perte des `OrthOST` exogènes mesurées au moyen du VM9 (codeur/décodeur développé par le groupe JPEG2000 [10]) sur des images Hyperion sont très bonnes et souvent meilleures que celles de la TKL non exogène [8]. Toutefois, la complexité du codeur EBCOT associé à son algorithme d'allocation optimale de débits entre composantes le rend inutilisable aujourd'hui dans un système de compression embarqué. Il est alors justifié de rechercher un codeur entropique, de complexité réduite, permettant un codage progressif en qualité ou en résolution, qui soit bien adapté aux `OrthOST`. Pour les images en niveaux de gris, les codeurs par arbres de zéros introduits par Shapiro [11] (EZW) puis améliorés par Said et Pearlman [12] (SPIHT) offrent ces qualités. De plus, SPIHT peut être utilisé sans codeur arithmétique (et donc avec une complexité nettement plus faible) pour une baisse de performance raisonnable.

Dans ce papier nous présentons les résultats que nous avons obtenus en cherchant un codeur par arbres de zéros qui soit de complexité réduite et bien adapté au schéma séparable, dans lequel la DO utilisée est la version quasi-orthogonale de la 9/7 de Daubechies ($\mathbf{W}^T\mathbf{W} \approx \mathbf{I}$, où $\mathbf{I}$ est l'identité) et la transformation spectrale une `OrthOST`. Nous commençons par rappeler dans la section suivante le point de vue de Said et Pearlman sur le codage progressif en qualité, puis nous présentons à la section 3 une extension de SPIHT permettant un codage progressif d'images 3D et des tentatives d'amélioration. Enfin nous comparons les résultats obtenus avec ces différents codeurs et le vm9 avant de conclure.

2 Codage progressif en qualité

Rappelons [12, 11] qu'un décodeur progressif en qualité améliore la qualité de l'image reconstruite au fur et à mesure qu'il reçoit des bits transmis par le codeur. Pour cela, l'image transformée quantifiée est initialisée par des valeurs nulles, puis progressivement améliorée. La transformation $\mathbf{X} \mapsto \mathbf{Y}$ étant quasi orthogonale, la distorsion mesurée par l'erreur quadratique moyenne (eqm) vérifie $D(\mathbf{X}, \widehat{\mathbf{X}}) = \frac{1}{NL}\sum_{k=1}^{N}\sum_{n=1}^{L}|\mathbf{X}_k(n) - \widehat{\mathbf{X}}_k(n)|^2 \approx D(\mathbf{Y}, \widehat{\mathbf{Y}})$, ce qui entraîne que la diminution de distorsion $D(\mathbf{X}, \widehat{\mathbf{X}})$ due au passage de 0 à $\widehat{Y}(i,j,k)$ vaut $\frac{1}{NL}[Y(i,j,k)^2 - |Y(i,j,k) - \widehat{Y}(i,j,k)|^2]$, pour toutes coordonnées (i,j,k). Introduisons les sous-ensembles $U_0 =]-1,1[$, $U_1 = [2^0, 2^1[$, $U_2 = [2^1, 2^2[$, $U_3 = [2^2, 2^3[$, … et pour $n < 0$, $U_n = -U_{-n}$, qui forment une partition de $\mathbb{R}$. Pour alléger les expressions nous n'écrivons plus les coordonnées (i,j,k) de Y et $\widehat{Y}$. Supposons que le décodeur connaisse le sous-ensemble U_n contenant Y et que cela fixe la valeur de $\widehat{Y} = \hat{y}_n$. Alors la diminution moyenne de distorsion vaut, au facteur $\frac{1}{NL}$ près, $\mathrm{E}[Y^2 - (Y - \hat{y}_n)^2 | Y \in U_n] = \mathrm{E}[Y|Y \in U_n]^2 - (\hat{y}_n - \mathrm{E}[Y|Y \in U_n])^2$, où E désigne l'espérance mathématique, dont l'emploi est justifié par le fait que les coefficients Y peuvent être modélisés par des variables aléatoires identiquement distribuées, par sous-bande et par composante. Le gain est maximal pour $\hat{y}_n = \mathrm{E}[Y|Y \in U_n]$ et si $|\hat{y}_n - \mathrm{E}[Y|Y \in U_n]| = \delta$, le gain est réduit de δ^2 par rapport à l'optimum. Pour $\hat{y}(n) = \mathrm{median}(U_n)$, en général $\delta \ll |\mathrm{E}[Y|Y \in U_n]|$, ce qui entraîne qu'en moyenne l'information $Y \in U_n$ diminue la distorsion d'approximativement $\frac{\hat{y}_n^2}{NL}$.

Nous supposons que chaque coefficient transformé est représenté sous un format binaire à virgule fixe. Il peut donc être considéré comme un entier signé. Un coefficient est dit *significatif* pour un plan de bits n quand $2^n \le |Y(i,j,k)|$, sinon il est non significatif. Pour $x \in \mathbb{R}$, $\lfloor x \rfloor$ désigne sa partie entière. Les codeurs EZW et SPIHT procèdent par plan de bits : partant du plan de bits de poids le plus fort $n = \lfloor \max_{i,j,k}\{\log_2 |Y(i,j,k)|\} \rfloor$; 1) l'étape de tri (*sorting pass*) consiste à trier les coefficients significatifs pour n et à transmettre l'information permettant au décodeur de trouver leurs coordonnées et leur signe ; 2) l'étape d'affinement (*refinement pass*) consiste à transmettre (dans un ordre connu du décodeur) les bits de poids n des coefficients qui sont déjà significatifs depuis un ou plusieurs plans de bits ; puis retour à l'étape 1) en remplaçant n par $n-1$ jusqu'au plan de bits $n = 0$ (lsb) ou jusqu'à ce que le débit (ou la distorsion) visé soit atteint. Les décodeurs EZW et SPIHT utilisent $\hat{y}_n = \mathrm{median}(U_n)$ quand $Y \in U_n$ devient significatif, puis affinent, par dichotomie, l'intervalle I contenant Y en fonction des bits d'affinement (*refinement pass*) et posent $\widehat{Y} = \mathrm{median}(I)$. Le fait de quantifier par $\mathrm{median}(U_n)$ à la place de $\mathrm{E}[Y|Y \in U_n]$ est sous-optimal en terme de distorsion, mais a l'avantage d'être très simple. À bas et moyens débits, la partie la plus importante du flot de bits est occupée par l'étape de tri. Le problème est de sélectionner, pour $n > 0$, les coefficients tels que $|Y(i,j,k)| \in U_n$. L'algorithme de tri partitionne l'ensemble des coordonnées en sous-ensembles $\mathcal{T}_m$ et effectue les tests (dits de *significance*) : $\max_{\ell \in \mathcal{T}_m}\{|Y(\ell)|\} \ge 2^n$? Si le décodeur reçoit un "non" en réponse à cette question, il sait que tous les coefficients de $\mathcal{T}_m$ sont non significatifs. Si la réponse est "oui", la même règle est appliquée au codeur et au décodeur pour diviser $\mathcal{T}_m$ en sous-ensembles plus petits dont la signifiance est testée par la suite. Pour réduire le nombre de comparaisons (et donc le nombre de bits transmis) il faut trouver une règle de partitionnement qui donne en moyenne des sous-ensembles non significatifs ayant beaucoup de coefficients et des sous-ensembles significatifs ayant un seul élément. Pour cela, EZW et SPIHT utilisent la propriété des DO de condenser l'énergie du signal vers les basses résolutions et de laisser une redondance entre coefficients localisés dans des sous-bandes ayant la même orientation spatiale à différentes échelles. Cette redondance est exploitée en regroupant, dans des arbres dits de zéros, les coefficients statistiquement dépendants à dif-

férentes échelles (voir [11, 12] pour la description précise des arbres), car pour un plan de bit donné, un nœud non significatif a très souvent tous ses descendants également non significatifs. La construction des arbres, leur partitionnement et l'ordre dans lequel ils sont présentés au test de signifiance diffèrent entre EZW et SPIHT. Cet ordre influe grandement sur les performances du codeur : il est préférable de présenter en premier les arbres ayant le plus de chance de devenir significatif. Dans EZW [11], à chaque étape de tri l'ensemble des coordonnées est parcouru en entier, sous-bande par sous-bande, des plus basses résolutions vers les plus hautes dans l'ordre LL_d, HL_d, LH_d, HH_d, HL_{d-1}, ..., HH_1 (pour d niveaux de décomposition) et dans chaque sous-bande, le balayage se fait ligne par ligne, puis pour chaque ligne, coefficient par coefficient. Dans SPIHT [12], l'ordre dans lequel les arbres (réduits à un pixel ou non) sont présentés au test de signifiance est géré par deux listes chaînées : celle des pixels non significatifs (LIP *List of Insignificant Pixels*) et celle des sous-ensembles non significatifs (LIS *List of Insignificant Sets*). Une troisième liste chaînée gère les pixels significatifs (LSP). Les éléments de la liste LIS peuvent-être de type A, correspondant à tous les descendants d'un nœud, ou bien de type B, correspondant à tous les descendants d'un nœud excepté ses quatre enfants. Lors de l'étape de tri, le test de signifiance est d'abord appliqué à tous les éléments de la liste LIP, puis à ceux de la liste LIS, du premier jusqu'au dernier. En même temps, les listes sont modifiées de la manière suivante : si un sous-ensemble de type B devient significatif, il est partagé en quatre sous-ensembles de type A qui sont ajoutés à la fin de la liste LIS (le test de leur signifiance sera exécuté quand leur tour viendra). Si un sous-ensemble de type A (associé au nœud η) devient significatif, il est partagé en quatre pixels (les enfants de η), le test de signifiance est appliqué à chacun d'eux avant de les ajouter à la fin des listes LIP ou LSP, et, si les enfants admettent des descendants, deux cas se présentent : soit l'ensemble de type B associé au nœud η est non significatif, il est alors ajouté à la fin de la liste LIS, soit il est significatif, les quatre sous-ensembles de type A associés aux enfants de η sont alors ajoutés à la fin de la liste LIS.

3 Extension de SPIHT au 3D

3.1 Codeur arithmétique d'ordre K

Dans nos tests, les codeurs décrits ci-dessous peuvent être associés à un codeur arithmétique, adaptatif, d'ordre variable et contextuel. Nous utilisons les notations suivantes : tout codeur est noté par son nom suivi d'un paramètre K entre crochets, par exemple `spiht` devient `spiht[`K`]`. Pour $K > 0$, un codeur arithmétique adaptatif estimant l'entropie d'ordre $K-1$ des symboles est appliqué et pour $K = 0$, aucun codeur arithmétique n'est utilisé.

3.2 SPIHT et allocation optimale

Il est bien connu que l'allocation de débits entre composantes est cruciale pour les performances de codeurs d'images hyperspectrales basés sur le schéma de compression séparable [1]. Pour avoir un élément de comparaison, nous avons construit un codeur 3D en appliquant l'algorithme SPIHT à chaque composante transformée $\mathbf{Y}_1$, ..., $\mathbf{Y}_N$ et en utilisant l'algorithme de Shoham et Gersho [13] pour réaliser l'allocation de débits entre composantes. Ce codeur, qui n'est pas progressif en qualité, sera désigné par `spiht+AO` dans la suite. Nous verrons à la section 4 qu'à bas et moyens débits, il dépasse légèrement les performances du VM9 quand il est associé à un codeur arithmétique adaptatif d'ordre 4. Ces résultats montrent l'efficacité des structures d'arbres de SPIHT appliquées aux composantes de l'image transformée quand l'allocation entre composantes est optimale. Pour éviter un calcul toujours coûteux d'allocation optimale de débits, nous avons cherché à étendre l'algorithme SPIHT aux images 3D en utilisant une seule liste LIS, une seule liste LIP et une seule liste LSP obtenues en mélangeant efficacement les listes LIS, LIP et LSP des SPIHT appliqués aux N composantes.

3.3 Tri des composantes de $\mathbf{Y}$

Pour cela, nous avons naturellement ordonné les composantes de l'image transformée par norme décroissante. Parmi les trois normes 1, 2 et ∞ que nous avons testées, la norme 1 donne les meilleurs résultats à bas et moyens débits. Le fait de trier les composantes avant ou après l'application des DO ne modifie pas les performances de façon appréciable (même pour la norme 1).

3.4 Nouveau SPIHT_2D

Pour éviter la gestion de listes chaînées (qui peuvent être très longues avec des images hyperspectrales), nous avons modifié l'algorithme SPIHT de la façon suivante. Pour chaque coordonnée (i, j, k), un booléen indique si $Y(i, j, k)$ est dans LIP, un autre s'il est dans LSP et une variable pouvant prendre trois états indique si (i, j, k) est un nœud associé à un sous-ensemble de type A, de type B ou à aucun élément de la liste LIS. L'ordre dans lequel les éléments des listes LIS et LIP sont présentés au test de signifiance n'est plus fixé par le chaînage des listes, mais par un balayage de toutes les coordonnées. Ce dernier s'effectue composante par composante, puis dans chaque composante comme le balayage d'EZW. Dans la suite nous appellerons `spiht_2D` ce codeur, car il n'utilise que des structures d'arbres spatiaux. Appliqué à des images 2D (cas $N = 1$), seul l'ordre dans lequel les éléments des listes LIP et LIS sont présentés au test de signifiance et, par conséquent, celui de la liste LSP diffèrent du SPIHT de Said et Pearlman et nous avons constaté expérimentalement que cette modification ne change pas les courbes débit versus distorsion (à $0,05$ dB près), des moyens aux très bas débits, avec ou sans codeur arithmétique. Notons que si l'allocation entre composantes n'est pas optimale à tous débits, nous avons la certitude, à la fin de chaque plan de bits, d'être sur la courbe d'allocation optimale. La figure 4.1 compare les performances de `spiht_2D` avec ou sans codeur arithmétique au `vm9` et montre également les pertes de `spiht_2D`

par rapport à spiht+AO.
L'ordre dans lequel les sous-ensembles sont présentés au test de signifiance influe grandement sur les performances du codeur. Nous avons testé différents balayages de l'ensemble des coordonnées et celui décrit ci-dessus, quand les composantes sont triées par norme 1 décroissante, donne les meilleures performances en général sur des images hyperspectrales.

3.5 Un arbre pour la redondance spectrale ?

Arbre de zéros 3D. Nous avons cherché des structures d'arbres 3D qui regrouperaient beaucoup de coefficients non significatifs. Dans SPIHT, les coefficients de la sous-bande LL sont regroupés par paquets de 2×2 avec tous leurs descendants spatiaux. Nous appelons *super-bloc* un tel paquet, qui peut être considéré comme tous les descendants d'un nœud fictif que nous appelons *super-père*. Les sous-ensembles 3D sont construits en regroupant tous les super-blocs de même position spatiale. Il faut en plus que la position spatiale soit un super-père. Autrement dit, les ensembles 3D non significatifs ne sont définis que quand leur racine est un super-père. Cette dernière condition est imposée pour éviter l'existence de plusieurs chemins reliant un nœud à un de ses descendants éloignés situé dans une autre composante. Notre structure d'arbre 3D diffère ainsi légèrement de celle appliquée dans [14] au schéma séparable avec la TKL comme transformation spectrale.
Le déroulement de l'algorithme 3D fournit les mêmes courbes (PSNR en fonction du débit) au centième de dB près que l'algorithme spiht_2D décrit ci-dessus. Cela s'explique par le fait que les arbres 3D ont pour seul but d'éviter que des super-pères "parlent" pour dire que leur sous-ensemble est non significatif. Dans cette situation, le bit transmis par un super-père bloque un nombre très élevé de pixels. Donc, la contribution de ce bit exprimée en bits/pixel dans le flux binaire est très faible. Le gain à espérer en évitant la transmission de ces bits est finalement négligeable. Lorsque le sous-ensemble associé à un super-père devient significatif, la structure d'arbre de zéros 3D est détruite et le codeur se comporte alors comme le codeur 2D ci-dessus. Pour améliorer ce dernier, il faudrait bloquer de manière efficace les zéros isolés (i.e. les éléments de LIP) issus d'arbres 2D cassés. En effet, une étude statistique, faite par plans de bits sur le flot binaire généré par spiht_2D[0], a montré que pour la plupart d'entre eux, environ 30% du débit est occupé par les bits de signifiance des zéros isolés.

Arbres de zéros 1D et 2D. Nous avons alors cherché une éventuelle redondance spectrale résiduelle apparaissant dans la liste LIP. Pour cela nous avons construit un nouveau codeur 3D qui utilise un codeur arithmétique adaptatif et contextuel pour coder les bits de signifiance de la liste LIP. Nous considérons *a priori* $2^p + 1$ contextes. Pour chaque élément de la liste LIP de coordonnées (i, j, k), on recherche p coefficients à la même position spatiale et sur des composantes k' plus énergétiques ($k' < k$) qui viennent de transmettre leur bit de signifiance pour le même plan de bits. Se présentent deux cas : 1) on les trouve : ils définissent alors le contexte (un parmi 2^p) ; 2) on n'en trouve pas autant : on se rabat alors sur un contexte par défaut. Nous appellerons dans la suite de ce paragraphe *codeur 3D* ce nouveau codeur (noté également spiht_3D_p). Remarquons que quand $p = 0$, par exemple dans spiht_3D_0, il n'y a plus qu'un seul contexte, ce qui supprime la dépendance spectrale et donne ainsi un codeur 2D. Mais il ne faut pas confondre le codeur spiht_3D_0[0] avec le codeur spiht_2D[1], pour lequel un codeur arithmétique d'ordre 0 est appliqué partout (bits de signes, de signifiance, etc.). Toutefois, les codeurs spiht_3D_0[K] et spiht_2D[K] sont identiques pour $K > 0$. Nous verrons à la section 4 qu'en exploitant ainsi la redondance spectrale résiduelle, nous pouvons gagner de $0,2$ à $0,5$ dB aux bas et moyens débits par rapport au codeur spiht_2D[0], mais que ce gain ne correspond qu'à la moitié de celui obtenu avec un codeur arithmétique d'ordre 4 (i.e. avec spiht_2D[5]).

4 Résultats expérimentaux

Nos tests ont été réalisés sur trois images AVIRIS[1] dont les dimensions sont $N_\ell = N_c = 512$ et $N = 224$.

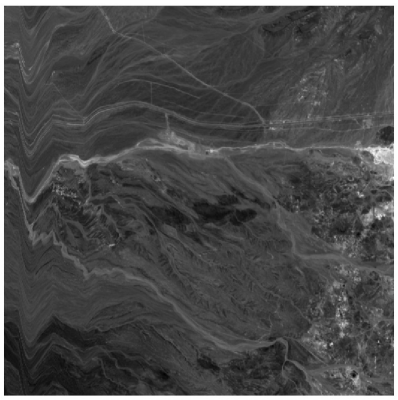

FIG. 1 – *Images AVIRIS :* Cuprite.

La DO utilisée est la 9/7 de Daubechies avec 5 niveaux de décomposition. La transformation spectrale (matrice $\mathbf{A}$) est une TKL, ou une OrthOST, dont les lignes ont été permutées pour que les composantes de l'image transformée soient ordonnées par norme 1 décroissante.

4.1 Avec une OrthOST

La figure 2 compare les distorsions (PSNR) des codeurs spiht+AO, spiht_2D, avec ou sans codeur arithmétique et le vm9. Nous voyons que l'écart entre spiht+AO, avec codeur arithmétique, et le vm9 est faible et que la baisse de performance de spiht_2D[K] par rapport à spiht[K]+AO vaut environ $0,3$ dB.

[1] Les images ont été téléchargées depuis le site de la NASA http ://aviris.jpl.nasa.gov/.

La figure 3 compare les performances de `spiht_3D_`p (pour $p = 0$ et $p = 5$), avec et sans codeur arithmétique, du vm9 et de `spiht_2D[0]` (i.e. sans codeur arithmétique). Il apparaît qu'ajouter un codeur arithmétique adaptatif sur les bits de signifiance des zéros isolés augmente le PSNR de plusieurs dixièmes de dB, ceci quel que soit l'ordre dans lequel sont parcourus ces symboles pour en extraire la redondance. L'écart entre `spiht_3D_5[5]` et `spiht_3D_0[5]` est faible et il est plus dû à l'augmentation du nombre de contextes du codeur arithmétique qu'à la capture d'une redondance spectrale résiduelle.

4.2 Avec une TKL

Pour étudier l'influence de la transformation, nous avons réalisé les mêmes opérations de codage et décodage que ci-dessus en remplaçant la transformation `OrthOST` par une TKL. Nous obtenons les mêmes allures de courbes. Toutefois nous avons constaté un écart plus important entre `spiht_3D_5[5]` et `spiht_3D_0[5]`, qui est probablement dû à une plus grande redondance spectrale qu'avec la transformation `OrthOST`.

5 Conclusion

Nous avons étendu l'algorithme SPIHT [12] pour qu'il permette un codage progressif en qualité d'images hyperspectrales. Notre extension mélange les listes LIS et LIP, construites par l'algorithme SPIHT appliqué à chaque composante, en parcourant l'ensemble des coordonnées composante par composante (en suivant la décroissance de la norme 1), puis comme dans EZW. Ce procédé fait perdre environ $0{,}3\,\mathrm{dB}$ par rapport au codeur SPIHT [12] appliqué à chaque composante transformée et une allocation optimale entre composantes. La perte en performance est compensée par le gain en rapidité de codage (i.e. une complexité plus faible). Par ailleurs, nous avons constaté que pour la plupart des plans de bits, environ 30% du flot de bits (sans codeur arithmétique) est dû aux bits de signifiance des zéros isolés. Nos tentatives pour réduire le nombre des zéros isolés en construisant des arbres dans la dimension spectrale ont échoué dans le sens où le coût de la gestion de la structure ajoutée était supérieur au gain apporté par la réduction du nombre de zéros isolés. En appliquant un codeur arithmétique adaptatif contextuel de différents ordres sur les bits de signifiance des zéros isolés nous n'avons constaté, avec les `OrthOST`, aucun gain appréciable dû à une redondance spectrale résiduelle. C'est moins vrai pour la TKL, mais pour cette dernière aussi, la redondance spatiale exploitable par un codeur arithmétique adaptatif contextuel reste supérieure à la redondance spectrale. La transformation `OrthOST` est calculée pour réduire au maximum la redondance spectrale et il est donc logique que nos tentatives pour réduire encore plus cette dernière aient échouées. Une étude en cours d'approfondissement montre que ces conclusions sont encore valides pour des transformations `OrthOST` exogènes.

Références

[1] J. T. Rucker, J. E. Fowler et N. H. Younan, "JPEG-2000 coding strategies for hyperspectral data", *Actes de International Geoscience and Remote Sensing Symposium*, vol. 1, 2005.

[2] B. Penna, T. Tillo, E. Magli et G. Olmo, "Transform coding techniques for lossy hyperspectral data compression", *IEEE Trans. Geoscience and Remote Sensing*, vol. 45, no. 5, 2007.

[3] D. S. Taubman et M. W. Marcellin, *JPEG2000 : Image compression fundamentals, standards and practice*, Kluwer Academic, 2002.

[4] P. S. Yeh, P. Armbruster, A. Kiely, B. Masschelein, G. Moury et C. Schafer, "The new CCSCS image compression recommendation", *Actes de IEEE Aerospace Conference*, Big Sky, 2005.

[5] D. S. Taubman, "High performance scalable compression with EBCOT", *IEEE Trans. on Image Processing*, vol. 9, no. 7, 2000.

[6] Q. Du et J. E. Fowler, "Hyperspectral image compression using JPEG2000 and principal component analysis", *IEEE Geoscience and Remote Sensing Letters*, vol. 4, 2007.

[7] I. P. Akam Bita, M. Barret et D. T. Pham, "Transformations linéaires optimales à hauts débits pour la compression d'images multi-composantes selon la norme JPEG2000", *Actes du XXIème GRETSI*, Troyes, 2007.

[8] M. Barret, I. P. Akam Bita, J.-L. Gutzwiller et F. Dalla Vedova, "Lossy hyperspectral images coding with exogenous quasi optimal transforms", *Actes de Data Compression Conference (DCC09)*, Snowbird, 2009.

[9] C. Thiebaut, E. Christophe, D. Lebedeff et C. Latry, "CNES studies of on-board compression for multispectral and hyperspectral images", *Actes de Satellite Data Compression, Communications, and Archiving III, SPIE*, vol. 6683, Edited by R. Heymann, B Huang and I. Gladkova, Sep. 2007.

[10] *JPEG2000 Verification Model 9.1*, ISO/IEC JTC 1/SC 29/WG 1 WG1 N2165, 2001.

[11] J. M. Shapiro, "Embedded image coding using zerotrees of wavelet coefficients", *IEEE Trans. on Signal Processing*, vol. 41, no. 12, 1993.

[12] A. Said et W. A. Pearlman, "A new, fast and efficient image codec based on Set Partitioning In Hierarchical Trees", *IEEE Trans. Circuits and Systems for Video Technology*, vol. 6, no. 3, 1996.

[13] Y. Shoham et A. Gersho, "Efficient bit allocation for an arbitrary set of quantizers", *IEEE Trans. Acoustics, Speech, and Signal Processing*, vol. 36, no. 9, 1988.

[14] P. L. Dragotti, G. Poggi et A.R.P. Ragozini, "Compression of multispectral images by three-dimensional SPIHT algorithm", *IEEE Trans. on Geoscience and Remote Sensing*, vol. 38, no. 1, 2000.

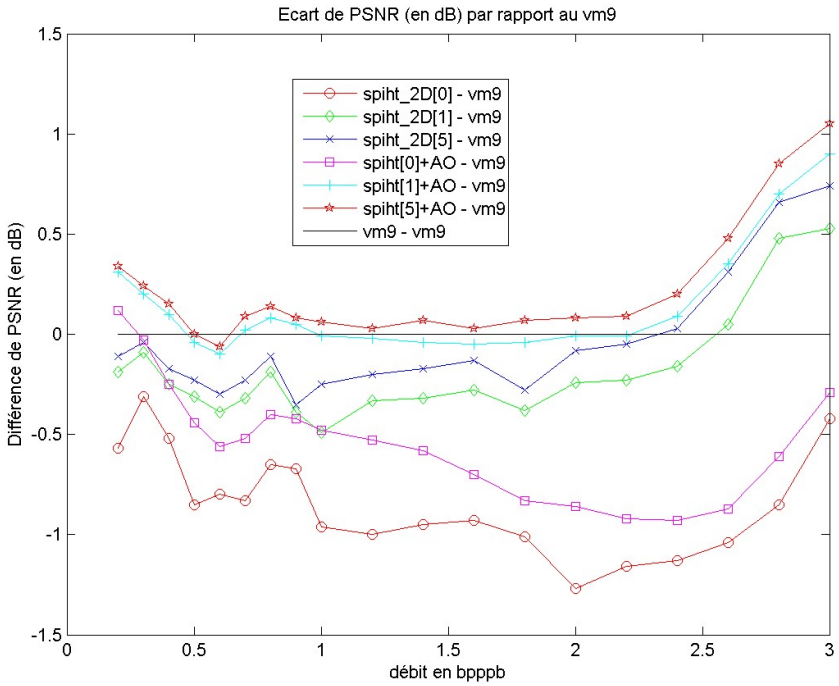

FIG. 2 – *Gain des codeurs* `spiht+AO` *et* `spiht_2D`*, avec et sans codeur arithmétique, par rapport au* `vm9`*.*

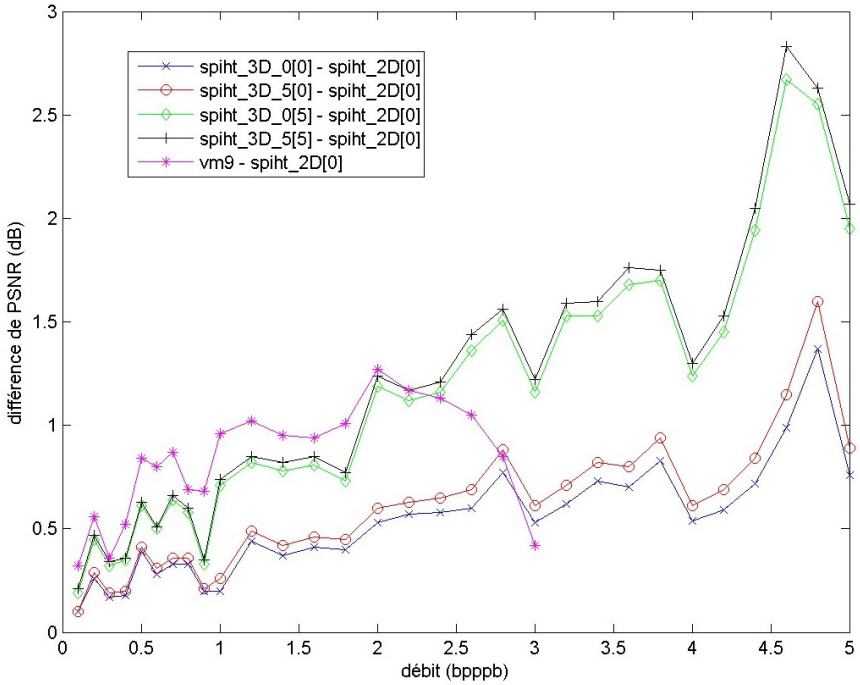

FIG. 3 – *Gains des codeurs* `spiht_3D_p` *avec et sans codeur arithmétique par rapport au* `spiht_2D[0]`*.*