Vers un schéma temps réel de compression multi-vues sans perte

B. Battin[1] et J. Lehuraux[1] et P. Vautrot[2] et L. Lucas[2]

[1]OPEXMedia
[2]Université de Reims Champagne-Ardenne

Résumé

Cet article s'intéresse au problème de la compression multi-vues en environnement virtualisé. Nous présentons notamment un nouveau schéma de compression multi-vues sans perte destiné à des scènes virtuelles et basé sur l'algorithme LOCO-I. Notre algorithme exploite la double redondance (spatiale et temporelle) spécifique à ce type de média en adaptant les étapes de prédiction et de modélisation de contexte à la matrice d'images. Les premiers tests effectués avec notre approche montrent que celle-ci propose de bons ratios de compression pour une complexité algorithmique moindre vis-à-vis des méthodes de l'état de l'art.

This paper investigates the problem of multiview video compression in virtualized environments. In particular, we present a new lossless compression scheme adapted to virtual scenes and based on the LOCO-I algorithm. This algorithm exploits the double data redundancy inherent to such media by extending the prediction and the context modeling step along the view and time axes. The preliminary results show that our approach produces good compression ratios and outperforms state-of-the-art lossless methods on this kind of data.

Mots clé : Multi-vues, 3DTV, compression 3D, auto-stéréoscopie, compression sans perte

1. Introduction

Durant la dernière décennie, l'accélération spectaculaire des évolutions technologiques a fondamentalement changé notre relation aux médias. Dans le domaine du multimédia, de nouvelles technologies, telles que la télévision 3D (3DTV) ou la "Free Viewpoint Television" (FTV), disposent maintenant d'une qualité d'image ainsi que d'un confort visuel au moins comparable à la télévision 2D. Dans un contexte différent, la virtualisation d'applications professionnelles graphiques 3D n'était pas considérée jusqu'à maintenant comme viable compte tenu des limitations en terme de technologie et de coût. Mais grâce aux dernières avancées en terme d'accélération GPU, plusieurs acteurs ont commencé à déployer leurs applications graphiques hautes performances sur des systèmes virtualisés. Ces récents progrès soulèvent toutefois certaines questions : comment ces deux domaines peuvent-ils co-exister ? Comment assurer une expérience immersive de qualité pour l'utilisateur qui souhaite évoluer et interagir librement dans le monde virtuel 3D comme si celui-ci était physiquement présent ? Délivrer du contenu multi-vues en temps réel sur le réseau peut être très coûteux en terme de bande passante et de délais. Les protocoles de streaming multi-vues interactifs deviennent un challenge technique substanciel. En particulier, ils doivent trouver un compromis entre efficacité de codage et flexibilité de navigation afin de délivrer les images 3D dans de bonnes conditions (sans perte de qualité et avec une bonne restitution 3D). Les images (et vidéos) multi-vues sont généralement produites à l'aide d'un ensemble de n caméras synchronisées, réelles ou virtuelles, qui capturent une même scène depuis différents points de vue. Ce type de système génère une quantité de données conséquente (typiquement n fois la taille d'un flux vidéo standard), qui nécessite d'être compressée efficacement pour pouvoir être stockée ou transmise sur le réseau. Pour qu'un schéma de compression multi-vues soit considéré comme pertinent, celui-ci doit prendre en compte la double redondance spatiale et temporelle spécifique aux séquences multi-vues. Parmi les différentes approches possibles, on distingue deux familles majeures : la première utilise directement les données multi-vues comme H.264/MVC [CWU*09, MSMW07] ou le prochain standard 3D-HEVC [Ohm13,MJCPP13]. La deuxième approche consiste à essayer d'atteindre un codage optimal en convertissant les données multi-vues dans un format 3D spécifique (MVD, LDI [JMG09] ou DES [SMM*09]), puis en codant cette nouvelle donnée avec les outils adaptés. L'utilisation de ces formats 3D permet d'exploiter la corrélation spatiale entre les vues et de réduire de manière significative la quantité d'informations à coder en aval. Bien que le nombre d'algorithmes de compression multi-vues devienne

conséquent, la compression sans perte de tels média est rarement évoquée. Pourtant, la compression multi-vues sans perte peut etre considérée comme nécessaire dans certains domaines applicatifs tels que les données médicales ou les rushs cinéma. Dans ce papier, nous présentons notre schéma de compression multi-vues sans perte : Multi-LS. Notre méthode, basée sur l'algorithme LOCO-I [WSS96,WSS00], exploite la double redondance en adaptant les étapes de prédiction fixe et de modélisation de contexte de l'algorithme à la matrice d'image. Nous avons choisi de baser notre approche sur l'algorithme LOCO-I car celui-ci a l'avantage de proposer un coût algorithmique très faible et d'être facilement parallélisé sur le serveur afin de régler les problèmes de délais évoqués dans le précédent paragraphe. Afin d'évaluer les performances de notre algorithme, en terme de ratio de compression, nous comparons nos résultats à plusieurs standards de compression sans perte issus de l'état de l'art. Cet article est organisé de la manière suivante : dans la section 2, nous dressons l'état de l'art actuel des techniques de compression vidéo sans perte. La section 3 présente l'algorithme de base puis met en avant les principaux concepts de chaque étape. Dans la section 4, nous décrivons et validons l'adaptation des étapes de prédiction fixe et de modélisation de contexte dans le cadre des séquences multi-vues. La section 5 est, quant à elle, dédiée à la présentation des résultats obtenus par Multi-LS comparés aux méthodes de l'état de l'art. Nous concluons enfin cet article en section 6.

2. Etat de l'art

Avec l'émergence, ces dernières années, de la télévision 3D, on a pu observer l'apparition de nouveaux algorithmes de compression dédiés à ce nouveau type de média très coûteux en terme de stockage ou de débit. Parmis ceux-ci, on compte notamment le standard H.264/MVC ainsi que son récent successeur 3D-HEVC (basé sur H.265). Toutefois, on constate que la question de la compression multi-vues sans perte n'est pas abordée dans la littérature et les deux standards précédemment cités n'offrent pas la possibilité de compresser le signal d'entrée sans distortion. Dans certains secteurs d'activité (comme la production cinématographique ou l'industrie médicale), il est cependant préférable de conserver ou de transmettre les données sans apporter de distortion au signal d'entrée afin de faciliter le travail de post-production dans le cas du cinéma et d'éviter les erreurs d'interprétation sur les données patient dans le domaine médical.

La solution généralement adoptée consiste alors à utiliser des outils standards de compression vidéo sans perte pour encoder séparement chaque flux vidéo correspondant aux N vues de la séquence auto-stéréoscopique. Cette aproche n'est cependant pas considérée comme optimale d'un point de vue efficacité de compression car celle-ci permet uniquement de prendre en compte la redondance temporelle au sein des images appartenant au flux associé à une vue donnée, alors que la redondance spatiale présente entre chaque vue n'est pas exploitée.

Parmi les outils disponibles pour compresser un signal vidéo sans perte, on trouve tout d'abord le nouveau standard H.265/HEVC [SOHW12, OSS*12] (ou MPEG-H partie 2) qui permet d'obtenir de meilleurs taux de compression que son prédécesseur H.264 [WSBL03] (ou MPEG-4 AVC). Ce gain de performance en terme de compression est notamment dû à l'utilisation d'algorithmes similaires à H.264 mais plus complexes sans pour autant augmenter de manière réellement significative les temps d'encodage. On considère généralement qu'à qualité égale entre H.264 et H.265/HEVC, le débit de la vidéo H.265/HEVC est deux fois moindre que celle compressée à l'aide d'H.264. Ces deux algorithmes offrent la possibilité de compresser sans perte le signal d'entrée en désactivant certaines étapes du pipeline d'encodage (voir la figure 1) tout en gardant les mécanismes de base de la compression vidéo à savoir l'estimation/compensation de mouvement ainsi que l'intra-prediction. L'inconvénient majeur de ces deux algorithmes provient de ces deux mécanismes, et plus particulièrement de l'estimation/compensation de mouvement, qui requiert beaucoup de temps de calcul comme nous le constaterons dans la section 5.

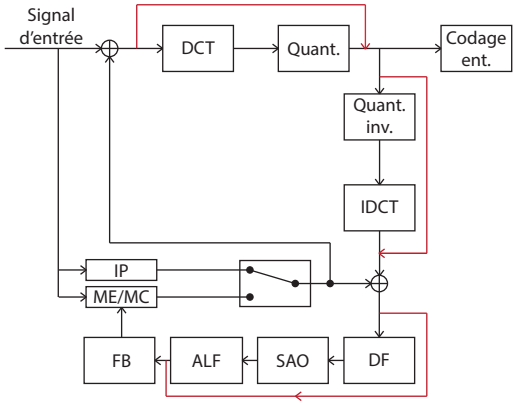

Figure 1: *Pipeline de codage sans perte pour HEVC (IP : Intra-Prediction, ME : Motion Estimation, MC : Motion Compensation, DF : Deblocking Filter, SAO : Sample Adaptive Offset, ALF : Adaptive Loop Filtering, FB : Frame Buffer).*

En marge de ces deux standards largement utilisés, on trouve plusieurs algorithmes fonctionnant tous de manière relativement similaire notamment HuffYUV [RG00], LOCO-I [WSS96] ou encore Lagarith [Gre04]. Ces trois algorithmes fonctionnent tous conformément au pipeline décrit par la figure 2 à savoir :

- Une étape de conversion colorimétrique, facultative suivant la prise en charge de l'algorithme, permettant une décorrélation préalable des données au niveau de l'espace de couleur.
- Une étape de prédiction qui va générer à partir de pixels voisins du pixel en cours de codage une valeur de prédiction à l'aide d'une fonction de prédiction spécifique. La différence entre la valeur de prédiction et la valeur du réel constitue alors l'erreur de prédiction.
- Enfin une étape de codage entropique de l'erreur de prédiction.

La table 1 permet d'identifier les différences notables entre ces trois algorithmes, relatives par exemple, à la fonc-

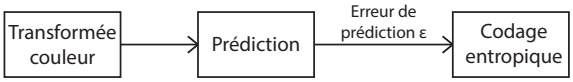

Figure 2: *Pipeline général des approches basées prédiction.*

	HuffYUV	Lagarith	LOCO-I
Espaces couleurs	RGB YUY2	RGB YUY2 YV12	RGB YUV
Prédiction	Gauche Gradient Médian	Médian	Médian
Codage entropique	Huffman	Arithmétique	Rice-Golomb
Inter prédiction	Non	NULL frames	Non
Run length	Non	Oui	Oui

Table 1: *Comparaison de codecs à base de prédicteurs*

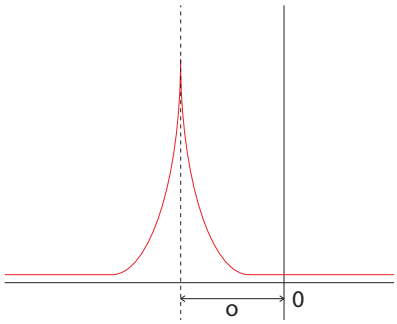

Figure 3: *Distribution des erreurs de prédiction fixe de la MED.*

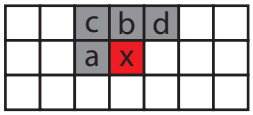

Figure 4: *Template $\mathcal{T}$ du pixel x.*

tion de prédiction utilisée, au type de codage entropique ou même à la présence ou non de mécanismes de prédiction inter-images. Dans ce dernier cas, on constate que Lagarith supporte également les “null frames”, ce qui lui permet d’éviter d’encoder l’image courante et d’utiliser l’image précédente lors du décodage dans le cas où ces dernières seraient mathématiquement identiques.

Toutes les approches précédemment citées dans cette section permettent d’exploiter la corrélation temporelle présente au sein de plusieurs frames consécutives et ainsi, de réduire de manière conséquente le volume initial de données. Dans le cas de la compression de vidéos multi-vues, l’algorithme doit aussi prendre en compte la corrélation spatiale entre les images issues de deux vues adjacentes au même temps t. Cette corrélation peut être réduite par des méthodes basées sur la compensation de disparité via l’utilisation de cartes de profondeur, comme c’est le cas pour le futur standard 3D-HEVC [MJCPP13]. Toutefois, ce type de processus rajoute un volume supplémentaire de données à compresser (sans perte) et peut être incompatible avec des applications de type rendu volumique pour lesquelles il est difficile de décider quelle profondeur utiliser pour un pixel donné. Pour ces deux raisons, nous avons opté pour une approche alternative qui sera présentée en section 4.

3. Les fondements de MULTI-LS

Multi-LS dérive de l’algorithme JPEG-LS qui constitue le standard ISO pour la compression d’image sans perte. Cette norme est basée sur l’algorithme LOCO-I (LOw COmplexity LOssless COmpression for Images) mis au point par [WSS96]. LOCO-I repose sur le concept de la modélisation de contexte et permet d’obtenir des ratios de compression similaires à l’algorithme CALIC [Wu95] tout en proposant une complexité algorithmique moindre. L’algorithme travaille de manière séquentielle sur chaque pixel de l’image, et pour chacun d’eux, procède aux cinq étapes suivantes.

3.1. Prédiction

Tout d’abord, l’algorithme doit générer une valeur de prédiction notée $\hat{x}_{MED}$ pour le pixel courant x à partir d’un voisinage constitué des pixels a, b, c et d (*cf.* figure 4). Cette opération est réalisée à l’aide d’une fonction de prédiction fixe appelée “median edge detector” (MED) et définie par l’équation 1. Celle-ci permet la détection d’éventuels bords (horizontaux ou verticaux) autour de x et ainsi d’adapter la valeur de prédiction en conséquence.

$$\hat{x}_{MED} = \begin{cases} \min(a,b) & \text{si} \quad c \geq \max(a,b) \\ \max(a,b) & \text{si} \quad c \leq \min(a,b) \\ a+b-c & \text{sinon} \end{cases} \tag{1}$$

A partir de $\hat{x}_{MED}$ et de x, on est maintenant en mesure de calculer l’erreur de prédiction fixe, notée ε_{MED}, qui est la différence entre $\hat{x}_{MED}$ et x. L’état de l’art, et tout particulièrement [NL80], nous indique que la distribution des erreurs de prédiction fixe générées par des fonctions de prédiction fixe (telle que MED) est modélisable par une distribution géométrique. Dans la section 3.3, nous verrons comment ce décalage est compensé grâce aux diverses informations associées au contexte courant afin d’assurer un codage entropique optimal.

3.2. Modélisation du contexte

Durant cette étape, l’algorithme va utiliser le pixel courant x ainsi que son template $\mathcal{T}$ afin de lui associer un contexte $\mathcal{C}_x$. Ce template est constitué des quatre pixels a, b, c et d appartenant au voisinage de x (*cf.* figure 4). L’algorithme calcule ensuite trois gradients à partir des valeurs des pixels appartenant à $\mathcal{T}$, conformément à l’équation 2.

$$\begin{aligned} g_1 &= d-b \\ g_2 &= b-c \\ g_3 &= c-a \end{aligned} \tag{2}$$

où la valeur de chaque gradient $g_i (1 \leq i \leq 3)$ est comprise dans l’intervalle $[-255;255]$.

Le triplet $\{g_1, g_2, g_3\}$ va nous permettre de caractériser l'activité autour du pixel x et de définir un modèle probabiliste pour le contexte C_x qui lui est associé. Dans [FWA04], l'auteur évoque un des problèmes récurrents dans les approches basées sur la modélisation de contexte : la dilution de contexte. Ce phénomène émerge généralement lorsqu'un trop grand nombre de contextes est utilisé, rendant insuffisante la quantité de statistiques pour chaque contexte et ayant un impact sur les performances en terme de compression. Afin de pallier cela, le nombre total de contextes est réduit par quantification des gradients g_i en q_i où $q_i \in [-4;4]$ et en fusionnant les contextes de signes opposés (l'information de signe est toutefois préservée dans une variable *SIGN* et utilisée dans la section 3.3). Ainsi, le nombre de contextes est réduit de 511^3 à $(9^3+1)/2 = 365$. Si $q_1 = q_2 = q_3 = 0$, l'algorithme considère que le pixel x appartient à une zone stationnaire de l'image. Dans ce cas, on comptabilise le nombre de pixels ayant la même valeur que x et cette chaîne est ensuite codée à l'aide de l'algorithme "Block-MELCODE" [OUO77].
Chaque contexte contient 4 compteurs *A, B, C* et *N* utilisés dans les deux prochaines sections :

- *A* accumule la valeur absolue des erreurs de prédiction (notée $|\varepsilon|$) et permet le calcul du paramètre *k* pour le codage entropique (*cf.* section 3.4) à l'aide du codeur de Rice-Golomb [Gol66]. *A* est initialisé à 4.
- *B* contient l'information nécessaire au calcul de *C* durant l'étape de correction adaptative (*cf.* section 3.3). Sa valeur initiale est 0.
- *C* contient la valeur de correction associée au contexte courant. Comme *B*, celui-ci est initialisé à 0.
- *N* contient le nombre d'occurences pour le contexte et est initialisé à 1.

3.3. Correction adaptative

Dans la section 3.1, nous avons évoqué la nécessité de compenser le décalage o entre le centre de la distribution géométrique associée au contexte courant C_x et l'origine. o est une valeur réelle pouvant s'exprimer sous la forme suivante : $o = R - s$ où R est entier et $s \in [0;1[$. L'étape de correction adaptative va stocker une approximation de R dans C et l'appliquer à la valeur $\hat{x}_{MED}$ précédemment calculée. Ainsi, l'amplitude des erreurs de prédiction corrigées ε se situe dans l'intervalle $]-1;0]$. Cette approximation peut être calculée de manière efficace en utilisant l'équation 3 où D représente la somme de toutes les erreurs de prédiction ε pour le contexte courant.

$$C = \lceil D/N \rceil \tag{3}$$

Toutefois, LOCO-I n'utilise pas cette manière de faire pour deux raisons majeures. Tout d'abord, l'équation 3 introduit une division qui n'est pas compatible avec une faible complexité algorithmique. Ensuite, la présence ponctuelle d'erreurs à forte amplitude affecterait de façon critique les valeurs suivantes de C impactant en aval le codage entropique. LOCO-I propose donc une méthode appelée "Adaptive bias cancellation" [WSS96] afin de contourner les deux problèmes précédemment evoqués : la valeur de l'erreur de prédiction corrigée peut être calculée à partir des compteurs B et N en utilisant l'équation 4.

$$\varepsilon = \hat{x}_{MED} + SIGN * C \tag{4}$$

Dans la section suivante, nous décrivons le processus de codage entropique pour la valeur de prédiction corrigée ε.

3.4. Codage entropique

L'étape de codage entropique de l'erreur de prédiction corrigée ε est réalisée grâce à un codage de Rice-Golomb. La littérature [GV75] s'accorde sur le fait que les codages de Rice-Golomb sont optimaux pour les distributions géométriques "one-sided" d'entiers non négatifs. En notant que la valeur de ε appartient à l'intervalle $[-255;255]$, nous devons dans un premier temps réduire ce dernier à $[0;255]$. La première opération consiste à fusionner l'extrêmité négative (respectivement extrêmité positive) de la TSGD ("Two-Sided Geometric Distribution") avec la partie centrale négative (respectivement positive), réduisant ainsi les valeurs de l'intervalle à $[-128;127]$. Cette manipulation n'affecte pas particulièrement la TSGD puisque le nombre d'occurences dans les parties extrêmes est négligeable vis-à-vis des parties centrales. Ensuite, l'intervalle $[-128;127]$ est porté à $[0;255]$ en entrelaçant les valeurs positives et négatives à partir de 0. Nous devons maintenant calculer le paramètre k qui régit la longueur du code binaire produit correspondant à ε. Suivant [GV75], une bonne estimation de la valeur optimale de k peut être obtenue à partir de l'équation 5 (E représente l'espérance mathématique).

$$k = \lceil \log_2 E\,[|\varepsilon|] \rceil \tag{5}$$

En utilisant les compteurs A et N de C_x, nous pouvons calculer $E\,[|\varepsilon|]$ par le quotient A/N. Ainsi, la valeur de k peut être déterminée par l'équation 6 et ε est codé entropiquement.

$$k = \min_{k'} \{k' | 2^{k'} N \geq A\} \tag{6}$$

3.5. Mise à jour du contexte

L'étape finale consiste à mettre à jour le contexte courant C_x pour prendre en compte les statistiques associées au pixel précédemment codé x. L'erreur de prédiction corrigée ε est ajoutée à B tandis que $|\varepsilon|$ est ajouté à A et N est incrémenté. LOCO-I effectue également une procédure de "réinitialisation" : si N est plus grand qu'un nombre prédéfini (entre 32 et 256), les compteurs *A, B,* et *N* sont divisés par 2, incrémentant de manière significative le poids des récentes statistiques pour ce contexte. Enfin, C est mis à jour par la réalisation de la procédure de compensation du biais.

4. Description de l'algorithme MULTI-LS

Dans cette section, nous présentons dans un premier temps notre adaptation de LOCO-I pour les séquences d'images multi-vues puis nous démontrons la pertinence d'une telle approche.

4.1. Adaptation aux séquences multi-vues

Les modifications principales apportées à l'algorithme LOCO-I résident dans les étapes de prédiction et de modélisation de contexte. Dans la section 3.1, nous avons pu

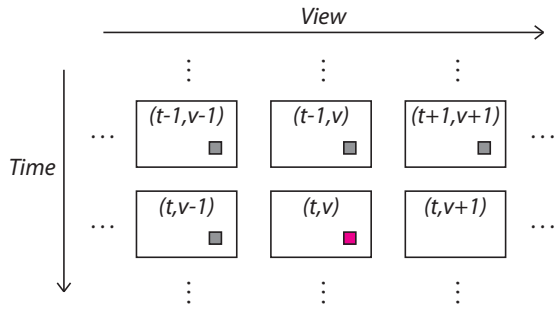

Figure 5: *Template Multi-LS.*

voir que LOCO-I utilisait 4 pixels a, b, c et d (*cf.* figure 4) pour produire une valeur de prédiction fixe $\hat{x}_{MED}$ et pour associer un contexte $\mathcal{C}_x$ au pixel courant x. Notons (i, j) la position du pixel courant x (où i correspond à l'indice des lignes tandis que j correspond à l'indice des colonnes), les pixels $(i-1, j-1)$, $(i-1, j)$,$(i-1, j+1)$ et $(i, j-1)$ constituent le "template" $\mathcal{T}$ associé à x. Notre adaptation s'attache à exploiter le modèle probabiliste utilisé dans LOCO-I sur l'ensemble de la matrice d'image spécifique aux séquences multi-vues, plutôt que sur une seule image. De ce fait, le "template" $\mathcal{T}$ associé au pixel courant x (à la position (i, j)) sera constitué de pixels situés à la même position mais appartenant aux images voisines. Ainsi, notons $I_{t,v}(i, j)$ le pixel situé à la position (i, j) dans la vue v au temps t, l'étape de prédiction fixe considérera les trois pixels $a_m = I_{t,v-1}(i, j)$, $b_m = I_{t-1,v}(i, j)$ et $c_m = I_{t-1,v-1}(i, j)$. De la même façon, l'étape de modélisation du contexte considérera a_m, b_m, c_m et $d_m = I_{t-1,v+1}(i, j)$ en accord avec la figure 5. Tandis que LOCO-I exploite les redondances spatiales au sein d'une même image, notre schéma s'appuie sur les redondances inter-vues et temporelles : les gradients calculés permettent la caractérisation d'une activité temporelle (produite par un mouvement entre le temps t et le temps $t+1$) grâce à g_3 ainsi que d'une activité inter-vues (parallaxe entre la vue courante v et ses vues voisines $v-1$ et $v+1$) grâce à g_1 et g_3. Le mode "run-length" est toujours utilisé pour encoder les zones uniformes d'une vue. En effet, avant de calculer les gradients multi-vues, notre schéma considère les gradients locaux (ceux du standard LOCO-I). Si $g_1 = g_2 = g_3 = 0$, alors l'algorithme utilise le mode "run-length" pour encoder de manière efficace la zone uniforme, sinon, les pixels multi-vues sont pris en compte.
La prochaine section s'attache à confirmer la pertinence de notre adaptation multi-vues.

4.2. Validation du "template" multi-vues

L'efficacité de l'algorithme LOCO-I provient de son mécanisme de prédiction adaptatif (prédiction fixe et compensation adaptative du biais) permettant de produire des résidus de prédiction ε distribués selon une TSGD centrée en zéro. Cette distribution est ensuite entièrement exploitée par le codage Rice-Golomb pour produire des mots binaires de longueurs optimales. En conséquence, nous devons veiller à ce que notre adaptation ne perturbe pas ce comportement. La figure 6 de même que la figure 7 démontrent la distribution des résidus de prédiction ε pour deux de nos quatre ensembles de données multi-vues ("Cartes" et "Pinceau"). En analysant la figure 5, nous pouvons noter que la distribution des résidus de prédiction conserve le comportement attendu et l'utilisation du codage de Rice-Golomb pour l'étape de codage entropique est totalement justifié.

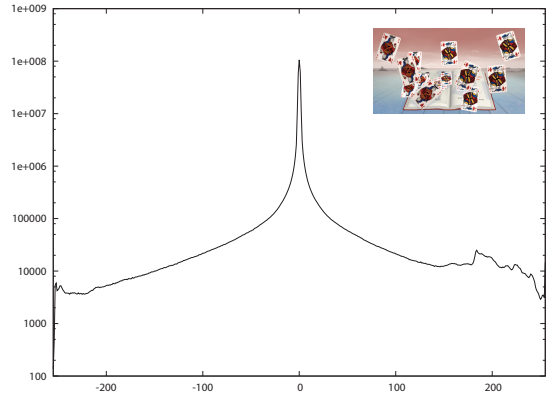

Figure 6: *Répartition des résidus de prédiction corrigés sur la séquence "Cartes".*

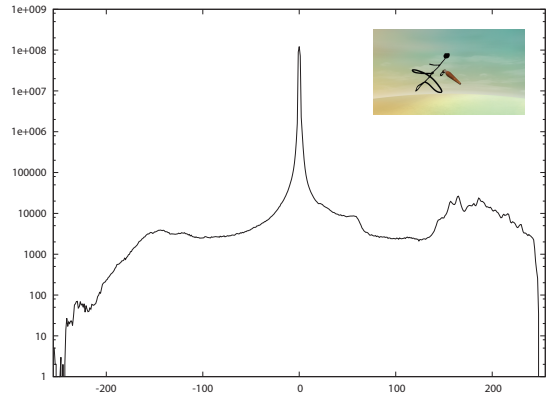

Figure 7: *Répartition des résidus de prédiction corrigés sur la séquence "Pinceau".*

5. Résultats

Afin d'évaluer au mieux l'efficacité de notre méthode, nous proposons dans cette section de comparer nos résultats avec ceux obtenus à l'aide d'algorithmes de compression sans perte, issus de l'état de l'art. Pour cela, nous avons choisi d'inclure dans nos tests des méthodes de compression sans perte d'images (CALIC [Wu95] et JPEG-LS), des méthodes de compression vidéo sans perte (Huffyuv et Lagarith) ainsi que les standards actuels de compression video (HEVC et H.264) en utilisant un profil d'encodage sans perte. Nous ajoutons enfin une méthode de compression de données basée sur l'algorithme LZMA.

Les tests ont été réalisés sur 3 séquences de données multi-vues nommées "Cartes", "Pinceau" et "Rose", chacune constituée de 8 vues sur 25 images temporelles. Pour CALIC, JPEG-LS et LZMA, les tests ont été menés en com-

	Cartes	Pinceau	Rose
CALIC	0.39	0.67	0.45
JPEG-LS	0.38	0.66	0.45
LZMA	0.59	0.97	0.46
HEVC intra	0.58	0.77	0.58
HEVC	0.78	0.98	0.66
H.264 intra	0.58	0.76	0.55
Lagarith	0.56	0.75	0.58
Huffyuv	0.48	0.71	0.48
Multi-LS	**0.65**	**0.75**	**0.46**

Table 2: *Résultats obtenus (ratio de compression)*

	Temps d'encodage
CALIC	+
JPEG-LS	+ +
LZMA	+
HEVC intra	- -
HEVC	- - - -
H.264 intra	- -
Lagarith	+ + +
Huffyuv	+ +
Multi-LS	+ +

Table 3: *Comparaison subjective du temps moyen d'encodage pour chaque séquence*

pressant chaque image de manière indépendante, alors que pour les autres algorithmes (HEVC, H.264, Huffyuv, Lagarith) un encodage simulcast est réalisé sur chaque séquence. On peut noter la présence de deux profils d'encodage pour HEVC : un profil "intra" où chaque frame temporelle est une frame *I* et un profil plus standard, utilisant le mécanisme d'estimation/compensation de mouvement.

La table 2 présente les résultats obtenus en terme de ratio de compression et la table 3 propose une comparaison subjective des temps d'encodage pour chaque algorithme. La figure 8 reprend, quant à elle, les résultats précédemment obtenus afin de les présenter sous forme d'un nuage de points.

D'après la table 2, on constate logiquement que le profil standard de HEVC (avec estimation/compensation) de mouvement propose les meilleurs performances en terme de ratio de compression, alors que les profils intra HEVC et H.264 obtiennent des résultats relativement similaires. Toutefois, la table 3 nous indique que pour ces trois approches, le temps d'encodage moyen sur les différentes séquences est très élevé (770 secondes pour HEVC/intra, 4840 secondes pour HEVC et 900 secondes pour H.264). Rappelons que notre objectif était de proposer un algorithme de compression multi-vues sans perte temps réel. On ne peut donc pas considérer ces trois algorithmes comme viables dans le contexte qui est le notre.

On peut également remarquer, toujours d'après la table 2, que notre approche MULTI-LS permet d'obtenir des taux de compression équivalents ou supérieurs par rapport à Huffyuv, CALIC, JPEG-LS ainsi que Lagarith (sauf pour la séquence "rose" pour laquelle Lagarith est plus performant).

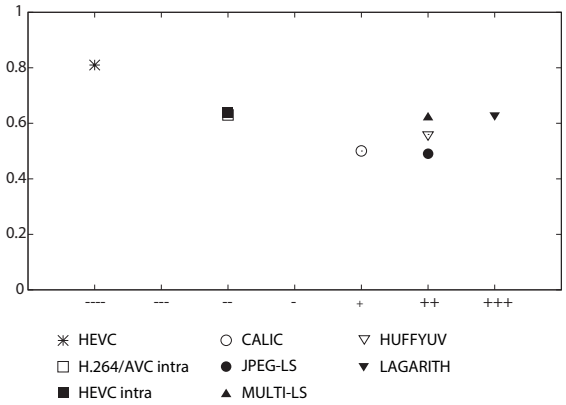

Figure 8: *Résultats obtenus.*

Au niveau des temps d'encodage, les algorithmes Huffyuv, JPEG-LS, Lagarith et notre approche Multi-LS permettent de compresser un flux vidéo monovue en temps réel. Cependant, il est important de noter que dans le cas de Huffyuv et de Lagarith, nous avons basé nos résultats sur des implémentations optimisées (SSE3, code ASM) alors que nous en sommes seulement à notre première implémentation en C de notre approche et qu'aucune optimisation n'a encore été réalisée (notamment une prochaine implémentation GPU sur CUDA). Nous espérons, par ce biais, proposer des temps d'encodage nettement inférieurs à ces standards, mais aussi permettre la compression de flux vidéos multi-vues en temps réel.

Enfin, concernant LZMA, on s'apperçoit que celui-ci propose de meilleurs taux de compression pour la séquence "Pinceau" uniquement. Cela peut être expliqué par le fait que la séquence "Pinceau" contient de larges zones uniformes où des méthodes basées sur un dictionnaire fournissent logiquement de meilleurs résultats.

6. Conclusion

Dans cet article, nous avons présenté un schéma de compression sans perte temps destiné à l'encodage temps réel de séquences multi-vues et basé sur LOCO-I. Les étapes de prédiction fixe et de modélisation de contexte sont adaptées à la matrice d'images multi-vues, exploitant ainsi les corrélations inter-vues et temporelle. Les résultats expérimentaux démontrent que le schéma proposé est capable de produire de meilleurs résultats que la plupart des algorithmes. Nous devons maintenant porter nos travaux sur la réalisation d'une implémentation GPU de Multi-LS, ce qui nous permettra de décharger le système dans les environnements virtualisés de sorte que les centres de traitement de données puissent proposer des expériences graphiques enrichies.

Références

[CWU*09] CHEN Y., WANG Y.-K., UGUR K., HANNUKSELA M., LAINEMA J., GABBOUJ M. : The emerging mvc standard for 3d video services. *EURASIP Journal on Advances in Signal Processing* (2009).

[FWA04] FORCHHAMMER S., WU X., ANDERSEN J. D. : Optimal context quantization in lossless compression of image data sequences. *IEEE Transactions on Image Processing* (2004).

[Gol66] GOLOMB S. W. : Run-length encodings. *IEEE Transactions on Information Theory. Vol. 12* (August 1966), 399–401.

[Gre04] GREENWOOD B. : Lagarith Lossless Video Codec. `http://lags.leetcode.net/codec.html`, 2004.

[GV75] GALLAGER R., VOORHIS D. V. : Optimal source codes for geometrically distributed integer alphabets. *IEEE Transactions on Information Theory* (1975).

[JMG09] JANTET V., MORIN L., GUILLEMOT C. : Incremental-ldi for multi-view coding. In *3DTV Conference : The True Vision - Capture, Transmission and Display of 3D Video* (2009).

[MJCPP13] MORA E., JUNG J., CAGNAZZO M., PESQUET-POPESCU B. : Initialization, limitation and predictive coding of the depth and texture quadtree in 3d-hevc video coding. *IEEE Transactions on Circuits and Systems for Video Technology* (2013).

[MSMW07] MERKLE P., SMOLIC A., MUELLER K., WIEGAND T. : Efficient prediction structures for multiview video coding. *IEEE Transactions on Circuits and Systems for Video Technology* (2007).

[NL80] NETRAVALI A., LIMB J. O. : Picture coding : A review. In *Proceedings of the IEEE* (1980).

[Ohm13] OHM J.-R. : Overview of 3d video coding standardization. In *Proceedings of 3DSA2013, Keynote speech 2* (2013).

[OSS*12] OHM J.-R., SULLIVAN G. J., SCHWARZ H., KAN T. K., WIEGAND T. : Comparison of the coding efficiency of video coding standards - including high efficiency video coding (hevc). *IEEE Transactions on Circuits and Systems for Video Technology. Vol. 22*, Num. 12 (2012), 1669–1684.

[OUO77] OHNISHI R., UENO Y., ONO F. : The efficient coding scheme for binary sources. *IECE of Japan* (1977).

[RG00] RUDIAK-GOULD B. : HuffYUV Lossless Codec. `http://neuron2.net/www.math.berkeley.edu/benrg/huffyuv.html`, 2000.

[SMM*09] SMOLIC A., MUELLER K., MERKLE P., KAUFF P., WIEGAND T. : An overview of available and emerging 3d video formats and depth enhanced stereo as efficient generic solution. In *Picture Coding Symposium* (2009).

[SOHW12] SULLIVAN G. J., OHM J.-R., HAN W.-J., WIEGAND T. : Overview of the high efficiency video coding (hevc) standard. *IEEE Transactions on Circuits and Systems for Video Technology. Vol. 22*, Num. 12 (2012), 1649–1668.

[WSBL03] WIEGAND T., SULLIVAN G. J., BJONTEGAARD G., LUTHRA A. : Overview of the h.264/avc video coding standard. *IEEE Transactions on Circuits and Systems for Video Technology. Vol. 13*, Num. 7 (2003), 560–576.

[WSS96] WEINBERGER M. J., SEROUSSI G., SAPIRO G. : Loco-i : A low complexity, context-based, lossless image compression algorithm. In *Data Compression Conference* (1996).

[WSS00] WEINBERGER M. J., SEROUSSI G., SAPIRO G. : The loco-i lossless image compression algorithm : principles and standardization into jpeg-ls. *IEEE Transactions on Image Processing. Vol. 9* (August 2000), 1309–1324.

[Wu95] WU X. : Context selection and quantization for lossless image coding. In *DCC 95, Data Compression Conference* (1995).